PMS
2020 2021
17th. International Workshop on
Project Management and Scheduling
April 21-23, 2021, Toulouse

https://pms2020.sciencesconf.org/

Book of Extended Abstracts

Table of contents

Foreword **9**

Committees **11**

Best student paper award **12**

Conference Program **13**

Plenary Talks **18**

Robustness in Scheduling
Marjan van den Akker 18

Industrial project and machine scheduling with Constraint Programming
Philippe Laborie 19

Data driven Project Management
Mario Vanhoucke 20

Modeling and solving complex job-shop scheduling problems
Stéphane *Dauzère-Pérès* 21

Extended Abstracts **22**

Heuristics for Scheduling Pipe-laying Support Vessels: An Identical Parallel Machine Scheduling Approach
Victor Abu-Marrul, Davi Mecler, Rafael Martinelli, Silvio Hamacher, Irina Gribkovskaia 22

Replication and sequencing of unreliable jobs
Alessandro Agnetis, Paolo Detti, Ben Hermans, Marco Pranzo 26

A Discrete Time Markov Decision Process to support the scheduling of re-manufacturing activities
Alessio Angius, Massimo Manzini, Marcello Urgo 30

A constraint programming approach for planning items transportation in a workshop context
Valentin Antuori, Emmanuel Hebrard, Marie-José Huguet, Siham Essodaigui, Alain Nguyen 34

Index merge in application to multi-skill project scheduling
Dmitry Arkhipov, Olga Battaia 38

Adapting the RCPSP framework to Evacuation Problems
Christian Artigues, Emmanuel Hébrard, Alain Quilliot, Peter Stuckey, Hélène Toussaint 42

Structural and Experimental Comparisons of Formulations for a Multi-Skill Project Scheduling Problem with Partial Preemption
Christian Artigues, Pierre Lopez, Oliver Polo Mejía 46

A Serial Schedule Generation Scheme for Project Scheduling in Disaster Management
Niels-Fabian Baur, Julia Rieck 50

Scheduling to minimize maximum lateness in tree data gathering networks
Joanna Berlińska 54

On the Activity Criticality in Project Scheduling with Generalized Precedence Relationships
Lucio Bianco, Massimiliano Caramia, Stefano Giordani 58

A Novel Matheuristic for the Multi-Site Resource-Constrained Project Scheduling Problem
Tamara Bigler, Mario Gnägi, Norbert Trautmann 62

Solution Repair by Inequality Network Propagation in LocalSolver
Léa Blaise, Christian Artigues, Thierry Benoist 66

The generalised resource-constrained project scheduling problem with flexible resource profiles
Matthew Bold, Burak Boyaci, Marc Goerigk, Chris Kirkbride 70

Why and how to evaluate the task threatness
Helena Brozova, Tomas Subrt, Jan Rydval, Petra Pavlickova 74

Local Search Algorithm to Solve a Scheduling Problem in Healthcare Training Center
Simon Caillard, Laure Brisoux Devendeville, Corinne Lucet 78

Computing lower bounds for the cumulative scheduling problem
Jacques Carlier, Antoine Jouglet, Abderrahim Sahli 82

Ultimate Instance Reduction for the Routing Open Shop
Ilya Chernykh 86

Optima Localization for the Routing Open Shop: Computer-aided Proof
Ilya Chernykh, Olga Krivonogova 90

A new tool for analysing and reporting solutions for the RCPSP and MMRCPSP
José Coelho, Mario Vanhoucke, Ricardo Amaro 94

Adaptive Robust Parallel Machine Scheduling
Izack Cohen, Krzysztof Postek, Shimrit Shtern 98

The Resource-Constrained Project Scheduling Problem: New Benchmark Results
Stefan Creemers 101

Scheduling and Routing Workers Teams for Ground Handling at Airports with Column Generation
Giacomo Dall'Olio, Rainer Kolisch 107

Robust scheduling for target tracking with wireless sensor network considering spatial uncertainty
Florian Delavernhe, André Rossi, Marc Sevaux 111

Exact solution of the two-machine flow shop problem with 3 operations
Federico Della Croce, Fabio Salassa, Vincent T'kindt 115

Adversarial bilevel scheduling on a single machine
Federico Della Croce, Vincent T'kindt 119

A Conjunctive-disjunctive Graph Modeling Approach for Job-Shop Scheduling Problem with Changing Modes
Xavier Delorme, Gérard Fleury, Philippe Lacomme, Damien Lamy 123

Near-Linear Approximation Algorithms for Scheduling Problems with Setup Times
Max Deppert, Klaus Jansen 127

Efficiency and Equity in the Multiple Organization Scheduling Problem
Martin Durand, Fanny Pascual 130

On the complexity of the crossdock truck-scheduling problem
Quentin Fabry, Alessandro Agnetis, Lotte Berghman, Cyril Briand 134

Linear inequalities for neighborhood based dominance properties for the common due-date scheduling problem
Anne-Elisabeth Falq, Pierre Fouilhoux, Safia Kedad-Sidhoum 138

An acceleration procedure for several objective functions in the permutation flow shop scheduling problem
Victor Fernandez-Viagas, José M. Molina-Pariente, Carla Talens, José M. Framiñan 142

Scheduling problems with processing time dependent profit: applications and a nice polynomial case
Florian Fontan, Nadia Brauner, Pierre Lemaire 146

Planning problem in Healthcare domain
Olivier Gérard, Laure Brisoux Devendeville, Corinne Lucet 151

Solving the Multi-mode Resource Investment Problem with Constraint Programming
Patrick Gerhards 155

Multi-Scenario Scheduling with Rejection Option to Minimize the Makespan Criterion
Miri Gilenson, Dvir Shabtay 159

A Continuous-Time Model for the Multi-Site Resource-Constrained Project Scheduling Problem
Mario Gnägi, Norbert Trautmann 163

Non-dominated sorting genetic algorithm for a bi-objective flexible flow shop problem. A Case Study
Ibeth Grattz Rodríguez, Jose-Fernando Jimenez, Eliana María González-Neira, Nicolás Eduardo Puerto Ordóñez, Yenny Alexandra Paredes Astudillo, Juan Pablo Caballero-Villalobos 167

An analytical model for budget allocation in risk prevention and risk protection
Xin Guan, Mario Vanhoucke 171

Embedded vision systems buffer minimization with energy consumption constraint
Khadija Hadj salem, Tifenn Rault, Alexis Robbes 175

Multi-project scheduling problems with shared multi-skill resource constraints
Meya Haroune, Cheikh Dhib, Emmanuel Néron, Ameur Soukhal, Hafedh Mohamed Babou, Mohamedade Nanne 179

Solving large, long-horizon resource constrained multi project scheduling problems with genetic algorithms
Brendan Hill, Adam Scholz, Lachlan Brown, Ana Novak 183

A mixed integer programming approach for scheduling aircraft arrivals at terminal airspace fixes and runway threshold
*Sana Ikli, Catherine Mancel, Marcel Mongeau, Xavier Olive, Emmanuel Racheslon*187

Optimization of order for containers placement schedule in rail terminal operations
Nadiia Kalaida, Rémy Dupas, Igor Grebennik 191

A Generation Scheme for the Resource-Constrained Project Scheduling Problem with Partially Renewable Resources and Time Windows
Mareike Karnebogen, Jürgen Zimmermann 195

On a Polynomial Solvability of the Routing Open Shop with a Variable Depot
Antonina Khramova, Ilya Chernykh 199

A Stochastic Programming Model to Schedule Projects under Cash Flow Uncertainty
Berfin Kutlag, Nazli Kalkan, Serhat Gul, Oncu Hazir 203

Multi-Objective Robotic Assembly Line Balancing Problem: A NSGA-II Approach Using Multi-Objective Shortest Path Decoders
Youssef Lahrichi, Laurent Deroussi, Nathalie Grangeon, Sylvie Norre 207

The Group Shop Scheduling Problem with power requirements
Damien Lamy, Simon Thevenin 211

A two-stage robust approach for minimizing the weighted number of tardy jobs with profit uncertainty
Henri Lefebvre, François Clautiaux, Boris Detienne 216

Scheduling of battery charging tasks with limited common power source
Tomasz Lemanski, Rafal Rozycki, Grzegorz Waligora, Jan Węglarz 220

Computational Experiments for the Heuristic Solutions of the Two-Stage Chain Reentrant Hybrid Flow Shop and Model Extensions
Lowell Lorenzo 224

Minimizing Flow Time on a Single Machine with Job Families and Setup Times
Arnaud Malapert, Margaux Nattaf 233

Using exponential smoothing to integrate the impact of corrective actions on project time forecasting
Annelies Martens, Mario Vanhoucke 237

An Experimental Investigation on the Performance of Priority Rules for the Dynamic Stochastic Resource Constrained Multi-Project Scheduling Problem
Philipp Melchiors, Rainer Kolisch, John Jack Kanet 241

Conditional Value-at-Risk of the Completion Time in Fuzzy Activity Networks
Carlo Meloni, Marco Pranzo, Marcella Samà 245

A column generation algorithm for the single machine parallel batch scheduling problem
Onur Ozturk 249

Mixed-Integer Programming Formulations for the Anchor-Robust Project Scheduling Problem
Adèle Pass-Lanneau, Pascale Bendotti, Philippe Chrétienne, Pierre Fouilhoux . . 253

Search space reduction in MILP approaches for the robust balancing of transfer lines
Aleksandr Pirogov, André Rossi, Evgeny Gurevsky, Alexandre Dolgui 257

An Inclusion-Exclusion based algorithm for the permutation flowshop scheduling problem
Olivier Ploton, Vincent T'kindt 261

Decision trees for robust scheduling
Tom Portoleau, Christian Artigues, Romain Guillaume 265

A Benders decomposition for the flexible cyclic jobshop problem
Félix Quinton, Idir Hamaz, Laurent Houssin 269

Exact and heuristic methods for characterizing optimal solutions for the 1——Lmax
Tifenn Rault, Ronan Bocquillon, Jean-Charles Billaut 273

A Comparison of two MILP formulations for the resource renting problem
Max Reinke, Jürgen Zimmermann 277

Minimizing the costs induced by perishable resource waste in a chemotherapy production unit
Alexis Robbes, Yannick Kergosien, Virginie André, Jean-Charles Billaut 281

A comparison of proactive and reactive scheduling approaches for the RCPSP with uncertain activity durations
Pedram Saeedi, Erik Demeulemeester 285

Towards the Optimisation of the Dynamic and Stochastic Resource-Constrained Multi-Project Scheduling Problem
Ugur Satic, Peter Jacko, Christopher Kirkbride 288

An FPTAS for Scheduling with Piecewise-Linear Nonmonotonic Convex Time-Dependent Processing Times and Job-Specific Agreeable Slopes
Helmut A. Sedding 292

An analysis of critical alternatives in the RCPSP-AS
Tom Servranckx, Mario Vanhoucke 296

Reference Class Forecasting to improve time and cost forecasts: Empirical and statistical analysis
Tom Servranckx, Mario Vanhoucke, Tarik Aouam 301

Buffer Sizing in Critical Chain Project Management by Network Decomposition
Bingling She, Bo Chen, Nicholas Hall 305

A new solution procedure for multi-skilled resources in resource-constrained project scheduling
Jakob Snauwaert, Mario Vanhoucke 309

The impact of limited budget on the corrective action taking process
Jie Song, Annelies Martens, Mario Vanhoucke 313

A New Lower Bound Approach for the Multi-mode Resource Constrained Project Scheduling Problem
Christian Stürck 317

How to find Critical Mass of Task Threatening the Projects
Tomas Subrt, Helena Brozova 321

Solving the stochastic multimode resource-constrained project scheduling problem
Claudio Szwarcfiter, Avraham Shtub, Yale T. Herer 325

Maximizing value-Modeling and solving lean project management
Claudio Szwarcfiter, Avraham Shtub, Yale T. Herer 329

Generating instances for the two-stage multi-machine assembly scheduling problem
Carla Talens, Victor Fernandez-Viagas, Paz Perez-Gonzalez 333

Metric Estimations for a Resource Leveling Problem With Variable Job Duration
Ilia Tarasov, Alain Haït, Olga Battaïa, Alexander Lazarev 337

Open shop problem with agreement graph: new results
Nour Elhouda Tellache, Mourad Boudhar, Farouk Yalaoui 341

Scheduling loads injection during flows merging in a collector
Blandine Vacher, Antoine Jouglet, Dritan Nace, Stéphane Pietrowicz, Marwane Bouznif 345

New benchmark datasets for the RCMPSP
Rob Van Eynde, Mario Vanhoucke 349

Minimizing Delays in Aircraft-Landing Scheduling
Marie-Sklaerder Vié, Nicolas Zufferey, Roel Leus 354

Evaluation of Scheduling Policies for the SRCPSP in a Dynamic Multi-Project Environment
Hendrik Weber, Rainer Kolisch 358

Modular equipment optimization in the design of multi-product reconfigurable manufacturing systems
Abdelkrim Yelles-Chaouche, Evgeny Gurevsky, Nadjib Brahimi, Alexandre Dolgui 362

Decomposition approach for fixed jobs multi-agent scheduling problem on parallel machines with renewable resources
Boukhalfa Zahout, Ameur Soukhal, Patrick Martineau 366

List of participants **370**

List of sponsors **377**

Author Index **379**

Foreword

Toulouse January 22, 2022.

PMS is an international workshop series devoted to Project Management and Scheduling. It was inaugurated by the European Working Group on Project Management and Scheduling (EURO - EWG Project management and scheduling), originally coordinated by Prof. Jan Węglarz from the Poznań University of Technology (Poland) and now coordinated by Prof. Erik Demeulemeester (KU Leuven, Belgium) and Prof. Joanna Jozefowska (Poznań University of Technology, Poland).

The EWG decided to organize a workshop every two years. The workshops provide an ideal opportunity to discuss recent and important issues in the field of project management (planning, scheduling, control) and machine scheduling (single and parallel machine problems, flow shop, job shop, etc.).

The first workshop was held in Lisbon in July 1988. The successor workshops were held in Compiègne (1990), Como (1992), Leuven (1994), Poznań (1996), Istanbul (1998), Osnabrück (2000), Valencia (2002), Nancy (2004), Poznań (2006), Istanbul (2008), Tours (2010), Leuven (2012), Munich (2014), Valencia (2016), Rome (2018).

The PMS 2020 workshop should have been held in Toulouse Business School, located in the heart of Toulouse, France, famous for its unique architecture which earned it the nickname la Ville Rose (”the Pink City”) and counts two UNESCO World heritage sites: the Canal du Midi and and the Basilica of St. Sernin, the largest remaining Romanesque building in Europe (https://www.toulouse-visit.com/), not to mention its famous South West French cooking and vibrant nightlife.

Due to the COVID-19 coronavirus pandemic, it was impossible to organize the event in 2020. It was postponed to April 21–23 2021, and then named PMS 2020/2021. Unfortunately the pandemic persistence forced us to turn the workshop in a full online event, the first online PMS !

The PMS 2020/2021 workshop is co-organised by researchers from LAAS-CNRS (https://www.laas.fr/), Toulouse Business School (https://www.tbs-education.fr) and ISAE-Supaero (https://www.isae-supaero.fr/) under the scientific supervision of the EURO EWG-PMS International committee.

The Workshop covers the following but non exhaustive list of project management and scheduling areas:

- Project Management: Network modeling, Project scheduling, Resource management, Due-date management, Project risk management, Project scheduling under uncertainty, Proactive/reactive project scheduling, Multicriteria project scheduling, Applications, Software
- Machine Scheduling: Shop scheduling, Scheduling with additional constraints, Machine assignment and scheduling, Flexible/robust scheduling, Grid scheduling, Multicriteria

scheduling, Applications, Software.

Methodological/theoretical papers related to Operational Research, Artificial Intelligence/Machine Learning models, exact and heuristic algorithms for scheduling problems were presented, as well as papers dealing with data-driven approaches, practical applications and industrial case studies.

Overall, 101 extended abstracts were received and 85 extended abstracts were accepted after a peer-review process. More than 23 nationalities are represented among the authors, as displayed below.

France	41	Canada	1
Belgium	11	Portugal	1
Germany	11	Turkey	1
Italy	7	Netherlands	1
United-Kingdom	5	Philippines	1
Russian federation	5	Brazil	1
Switzerland	4	Norway	1
Israel	4	Czech Republic	1
Australia	2	Colombia	1
Spain	2	Algeria	1
USA	2	Ukraine	1
Poland	2	TOTAL	107

As a new record for PMS, 216 participants registered to the workshop !

Not less than 23 extended abstracts applied to the Best Student Paper Award and 6 finalists were selected to present their work in 2 dedicated sessions. Prizes were awarded by EURO and Springer. Congratulations to Adèle Pass-Lanneau (First Prize), Quentin Fabry (2nd Prize) and Miri Gilenson (3rd Prize).

Last but not least, we had the pleasure to listen to four plenary talks by Marjan van den Akker, Philippe Laborie, Mario Vanhoucke and Stéphane Dauzère-Pérès.

You will find in these proceedings :

- The member list of the Organization Committee,
- The member list of the Program Committee,
- The finalists and the winners of the Best Student Paper Award,
- The conference program,
- The plenary talk abstracts and a short bio of each plenary speaker,
- The extended abstracts sorted by alphabetical order of the first author,
- The list of participants,
- The list of sponsors,
- The author index.

We warmly thank all the participants and the international program committee who greatly contributed to the large success of the workshop!

The organizing Committee.

Organizing Committee

pms2020@sciencesconf.org

Christian Artigues	LAAS-CNRS (Workshop chair)
Lotte Berghman	Toulouse Business School
Cyril Briand	Université Toulouse III-Paul Sabatier, LAAS-CNRS
Brigitte Ducrocq	LAAS-CNRS
Quentin Fabry	LAAS-CNRS
Sylvain Fayeulle	Toulouse Business School
Alain Haït	ISAE-SUPAERO
Laurent Houssin	Université Toulouse III-Paul Sabatier, LAAS-CNRS
Carla Juvin	LAAS-CNRS
Pierre Lopez	LAAS-CNRS
Tom Portoleau	LAAS-CNRS
Louis Rivière	LAAS-CNRS

International Program Committee

Alessandro Agnetis.	Università di Siena (Italy)
Ali Allahverdi.	Kuwait University (Kuwait)
Christian Artigues.	LAAS-CNRS (France)
Francisco Ballestín.	Universitat de València (Spain)
Jacek Błażewicz.	Poznań University of Technology (Poland)
Fayez Fouad Boctor.	Université Laval (Canada)
Massimiliano Caramia.	Università degli Studi di Roma "Tor Vergata" (Italy)
Jacques Carlier.	Université de Technologie de Compiègne (France)
Erik Demeulemeester.	Katholieke Universiteit Leuven (Belgium)
Joanna Józefowska.	Poznań University of Technology (Poland)
Sigrid Knust.	Universität Osnabrück (Germany)
Rainer Kolisch.	Technische Universität München (Germany)
Mikhail Kovalyov.	National Academy of Sciences of Belarus (Belarus)
Wieslaw Kubiak.	Memorial University (Canada)
Linet Özdamar.	Yeditepe Üniversitesi (Turkey)
Erwin Pesch.	Universität Siegen (Germany)
Chris Potts.	University of Southampton (United Kingdom)
Rubén Ruiz.	Universitat Politècnica de València (Spain)
Funda Sivrikaya-Şerifoğlu.	Istanbul Bilgi Üniversitesi (Turkey)
Avraham Shtub.	Technion - Israel Institute of Technology (Israel)
Vincent T'kindt.	Université François Rabelais Tours (France)
Norbert Trautmann.	Universität Bern (Switzerland)
Mario Vanhoucke.	Ghent University (Belgium)
Jan Węglarz.	Poznań University of Technology (Poland)
Jürgen Zimmermann.	Technische Universität Clausthal (Germany)

Best student paper award

The following extended abstracts were finalists of the best student paper award.

- **Quentin Fabry**, Alessandro Agnetis, Lotte Berghman, Cyril Briand
  On the complexity of the crossdock truck-scheduling problem
- **Miri Gilenson**, Dvir Shabtay
  Multi-Scenario Scheduling with Rejection Option to Minimize the Makespan Criterion
- **Mareike Karnebogen**, Jürgen Zimmermann
  A Generation Scheme for the Resource-Constrained Project Scheduling Problem with Partially Renewable Resources and Time Windows
- **Adèle Pass-Lanneau**, Pascale Bendotti, Philippe Chrétienne, Pierre Fouilhoux
  Mixed-Integer Programming Formulations for the Anchor-Robust Project Scheduling Problem
- **Alexis Robbes**, Yannick Kergosien, Virginie André, Jean-Charles Billaut
  Minimizing the costs induced by perishable resource waste in a chemotherapy production unit
- **Claudio Szwarcfiter**, Avraham Shtub, Yale T. Herer
  Maximizing value and Minimizing Waste: Modeling and solving lean project management

The jury is made of the whole International Program Committee, except those involved in the phD thesis.

- The first prize (500€) was awarded by EURO to **Adèle Pass-Lanneau** (Sorbonne Université and EDF R&D, France)
- The second prize (300€) was awarded by EURO to **Quentin Fabry** (LAAS-CNRS and Toulouse Business School, France)
- The third prize was awarded to (200€) was awarded by EURO to **Miri Gilenson** (Ben-Gurion university of the Negev, Israel)

Books were also offered to the winners by Springer.

Wednesday 21th of April 2021

9 AM – 9:20 AM: Welcome session

9:20 AM – 10:20AM: Plenary session 1

Robustness in Scheduling by Marjan van den Akker

chair: Erwin Pesch

10:20 AM – 10:40 AM: Coffee break

10:40 AM – 12:20 PM: Parallel session W1

Room 1: RCPSP 1

chair: Stefan Creemers

A New Lower Bound Approach for the Multi-mode Resource Constrained Project Scheduling Problem
Christian Stuerck

The Resource-Constrained Project Scheduling Problem: New Benchmark Results
Stefan Creemers

A new solution procedure for multi-skilled resources in resource-constrained project scheduling
Jakob Snauwaert and Mario Vanhoucke

Multi-project scheduling problems with shared multi-skill resource constraints
Meya Haroune, Cheikh Dhib, Emmanuel Néron, Ameur Soukhal, Hafedh

Solving large, long-horizon resource constrained multi project scheduling problems with genetic algorithms
Brendan Hill, Adam Scholz, Lachlan Brown and Ana Novak

Room 2: Shop scheduling

chair: Ilya Chernykh

A Conjunctive-disjunctive Graph Modeling Approach for Job-Shop Scheduling Problem with Changing Modes
Xavier Delorme, Gérard Fleury, Philippe Lacomme and Damien Lamy

Generating instances for the two-stage multi-machine assembly scheduling problem
Carla Talens, Victor Fernandez-Viagas and Paz Perez-Gonzalez

The Group Shop Scheduling
Damien Lamy and Simon Thevenin

Optima Localization for the Routing Open Shop: Computer-aided Proof
Ilya Chernykh and Olga Krivonogova

Ultimate Instance Reduction for the Routing Open Shop
Ilya Chernykh

12:20 PM – 1:30 PM: Lunch break

1:30 PM – 3:10 PM: Parallel session W2

Room 1: Risk management

chair: Mario Vanhoucke

An analytical model for budget allocation in risk prevention and risk protection
Xin Guan and Mario Vanhoucke

Conditional Value-at-Risk of the Completion Time in Fuzzy Activity Networks
Carlo Meloni, Marco Pranzo and Marcella Sama

Reference Class Forecasting to improve time and cost forecasts: Empirical and statistical analysis
Tom Servranckx, Mario Vanhoucke and Tarik Aouam

The impact of limited budget on the corrective action taking process
Jie Song, Annelies Martens and Mario Vanhoucke

Using exponential smoothing to integrate the impact of corrective actions on project time forecasting
Annelies Martens and Mario Vanhoucke

Room 2: Best student paper 1 (30 min slots)

chair: Joanna Jozefowska

Mixed-Integer Programming Formulations for the Anchor-Robust Project Scheduling Problem
Adèle Pass-Lanneau, Pascale Bendotti, Philippe Chrétienne and Pierre Fouilhoux

Multi-Scenario Scheduling with Rejection Option to Minimize the Makespan Criterion
Miri Gilenson and Dvir Shabtay

On the complexity of the crossdock truck-scheduling problem
Quentin Fabry, Alessandro Agnetis, Lotte Berghman and Cyril Briand

3:10 PM – 3:30 PM: Coffee break

3:30 PM – 4:30 PM: Industrial plenary talk

Industrial project and machine scheduling with Constraint Programming by Philippe Laborie

chair: Pierre Lopez

4:30 PM – 5:50 PM: Parallel session W3

Room 1: Single machine scheduling
chair: Vincent T'kindt

A column generation algorithm for the single machine parallel batch scheduling problem
Onur Ozturk

Adversarial bilevel scheduling on a single machine
Federico Della Croce and Vincent T'kindt

Exact and heuristic methods for characterizing optimal solutions for the 1||Lmax
Tifenn Rault, Ronan Bocquillon and Jean-Charles Billaut

Minimizing Flow Time on a Single Machine with Job Families and Setup Times
Arnaud Malapert and Margaux Nattaf

Room 2: Robust scheduling 1
chair: Marcello Urgo

Robust scheduling for target tracking with wireless sensor network considering spatial uncertainty
Florian Delavernhe, André Rossi and Marc Sevaux

A two-stage robust approach for minimizing the weighted number of tardy jobs with profit uncertainty
Henri Lefebvre, François Clautiaux and Boris Détienne

A Discrete Time Markov Decision Process to support the scheduling of re-manufacturing activities
Alessio Angius, Massimo Lanzini and Marcello Urgo

Buffer Sizing in Critical Chain Project Management by Network Decomposition
Bingling She, Bo Chen and Nicholas Hall

Thursday the 22th of April 2021

9 AM – 10:40 AM: Parallel session T1

Room 1: Constraint programming
chair: Christian Artigues

A constraint programming approach for planning items transportation in a workshop context
Valentin Antuori, Emmanuel Hébrard, Marie-José Huguet, Siham Essodaigui and Alain Nguyen

Embedded vision systems buffer minimization with energy consumption constraint
Khadija Hadj Salem, Tifenn Rault and Alexis Robbes

Solution Repair by Inequality Network Propagation in LocalSolver
Léa Blaise, Christian Artigues and Thierry Benoist

Solving the Multi-mode Resource Investment Problem with Constraint Programming
Patrick Gerhards

Structural and Experimental Comparisons of Formulations for a Multi-Skill Project Scheduling Problem with Partial Preemption
Christian Artigues, Pierre Lopez and Oliver Polo

Room 2: Best student paper 2 (30 min slots)
chair: Erik Demeulemeester

A Generation Scheme for the Resource-Constrained Project Scheduling Problem with Partially Renewable Resources and Time Windows
Mareike Karnebogen and Jurgen Zimmermann

Maximizing value: Modeling and solving lean project management
Claudio Szwarcfiter, Avraham Shtub and Yale T. Herer

Minimizing the costs induced by perishable resource waste in a chemotherapy production unit
Alexis Robbes, Yannick Kergosien, Virginie André and Jean-Charles Billaut

10:40 AM – 11:00 AM: Coffee break

11:00 AM – 12:20 PM: Parallel session T2

Room 1: Resource-unit focused project scheduling
chair: Norbert Trautmann

Index merge in application to multi-skill project scheduling
Dimitry Arkhipov and Olga Battaia

Metric Estimations for a Resource Leveling Problem With Variable Job Duration
Ilia Tarasov, Alain Haït, Olga Battaia and Alexander Lazarev

A Continuous-Time Model for the Multi-Site Resource-Constrained Project Scheduling Problem
Mario Gnaegi and Norbert Trautmann

A Novel Matheuristic for the Multi-Site Resource-Constrained Project Scheduling Problem
Tamara Bigler, Mario Gnaegi and Norbert Trautmann

Room 2: Complexity results and Approximation algorithms
chair: Alessandro Agnetis

An FPTAS for Scheduling with Piecewise-Linear Nonmonotonic Convex Time-Dependent Processing Times and Job-Specific Agreeable Slopes
Helmut A. Sedding

Duplication and sequencing of unreliable jobs
Alessandro Agnetis, Paolo Detti, Ben Hermans and Marco Pranzo

On a Polynomial Solvability of the Routing Open Shop with a Variable Depot
Antonia Khramova and Ilya Chernykh

Near-Linear Approximation Algorithms for Scheduling Problems with Setup Times
Max Deppert and Klaus Jansen

12:20 PM – 1:30 PM: Lunch break

1:30 PM – 3:10 PM: Parallel session T3

Room 1: Flexible scheduling chair: Nadia Brauner	Room 2: Applications: Logistics chair: Roel Leus
A Benders decomposition for the flexible cyclic jobshop problem *Félix Quinton, Idir Hamaz and Laurent Houssin*	**A mixed integer programming approach for scheduling aircraft arrivals at terminal airspace fixes and runway threshold** *Sana Ikli, Catherine Mancel, Marcel Mongeau, Xavier Olive and Emmanuel Rachelson*
A Serial Schedule Generation Scheme for Project Scheduling in Disaster Management *Niels-Fabian Baur and Julia Rieck*	**Minimizing Delays in Aircraft-Landing Scheduling** *Marie-Sklaerder Vie, Nicolas Zufferey and Roel Leus*
Computational Experiments for the Heuristic Solutions of the Two-Stage Chain Reentrant Hybrid Flow Shop and Model Extensions *Lowell Lorenzo*	**Scheduling and Routing Workers Teams for Ground Handling at Airports with Column Generation** *Giacomo Dall'olio and Rainer Kolisch*
Scheduling problems with processing time dependent profit: applications and a nice polynomial case *Florian Fontan, Nadia Brauner and Pierre Lemaire*	**Heuristics for Scheduling Pipe-laying Support Vessels: An Identical Parallel Machine Scheduling Approach** *Victor Abu-Marrul, Davi Mecler, Rafael Martinelli, Silvio Hamacher and Irina Gribkovskaia*
The generalised resource-constrained project scheduling problem with flexible resource profiles *Matthew Bold, Burak Boyaci, Marc Goerigk and Chris Kirkbride*	**A Comparison of two MILP formulations for the resource renting problem** *Max Reinke and Jurgen Zimmermann*

3:10 PM – 3:30 PM: Coffee break

3:30 PM – 4:30 PM: Plenary session 3

Data driven Project Management by Mario Vanhoucke

chair: Sigrid Knust

4:30 PM – 5:30 PM: Parallel session T4

Room 1: Applications: Manufacturing chair: Jan Węglarz	Room 2: Lower bounds, dominance and formulations chair: Safia Kedad-Sidhoum
Modular equipment optimization in the design of multi-product reconfigurable manufacturing systems *Abdelkrim R. Yelles-Chaouche, Evgeny Gurevsky, Nadjib Brahimi, Alexandre Dolgui*	**Computing lower bounds for the cumulative scheduling problem** *Jacques Carlier, Antoine Jouglet and Abderrahim Sahli*
Scheduling loads injection during flows merging in a collector *Blandine Vacher, Antoine Jouglet, Dritan Nace, Stephane Pietrowicz and Marwane Bouznif*	**Linear inequalities for neighborhood based dominance properties for the common due-date scheduling problem** *Anne-Elisabeth Falq, Safia Kedad-Sidhoum and Pierre Fouilhoux*
Scheduling of battery charging tasks with limited common power source *Tomasz Lemanski, Rafal Rozycki, Grzegorz Waligóra and Jan Węglarz*	**Open shop problem with agreement graph: new results** *Nour Elhouda Tellache, Mourad Boudhar and Farouk Yalaoui*

5:30 PM – 5:50 PM: Best student paper award

Friday the 23th of April 2021

9 AM – 10:40 AM: Parallel session F1

Room 1: Robust RCPSP chair: Rainer Kolisch	Room 2: Multicriteria scheduling chair: Helena Brozova
A comparison of proactive and reactive scheduling approaches for the RCPSP with uncertain activity durations *Pedram Saeedi and Erik Demeulemeester*	**Decomposition approach for fixed jobs multi-agent scheduling problem on parallel machines with renewable resources** *Zahout Boukhalfa, Ameur Soukhal and Patrick Martineau*
An Experimental Investigation on the Performance of Priority Rules for the Dynamic Stochastic Resource Constrained Multi-Project Scheduling Problem *Philipp Melchiors, Rainer Kolisch and John Jack Kanet*	**Efficiency and Equity in the Multiple Organization Scheduling Problem** *Martin Durand and Fanny Pascual*
Evaluation of Scheduling Policies for the SRCPSP in a Dynamic Multi-Project Environment *Hendrik Weber and Rainer Kolisch*	**Why and how to evaluate the task threatness** *Helena Brozova, Tomas Subrt, Jan Rydval and Petra Pavlickova*
Solving the stochastic multimode resource-constrained project scheduling problem *Claudio Szwarcfiter, Avraham Shtub and Yale T. Herer*	**How to find Critical Mass of Task Threatening the Projects** *Tomas Subrt and Helena Brozova*
Towards the Optimisation of the Dynamic and Stochastic Resource-Constrained Multi-Project Scheduling Problem *Ugur Satic, Peter Jacko and Christopher Kirkbride*	**Multi-Objective Robotic Assembly Line Balancing Problem: A NSGA-II Approach Using Multi-Objective Shortest Path Decoders** *Youssef Lahrichi, Laurent Deroussi, Nathalie Grangeon and Sylvie Norre*

10:40 AM – 11:00 AM: Coffee break

11:00 AM – 12:20 PM: Parallel session F2

Room 1: Robust scheduling 2 chair: Izack Cohen	Room 2: Applications: health care and external resources chair: Jurgen Zimmermann
Adaptive Robust Parallel Machine Scheduling *Izack Cohen, Krzystof Postek and Shimrit Shtern*	**Local Search Algorithm to Solve a Scheduling Problem in Healthcare Training Center** *Simon Caillard, Laure Brisoux Devendeville and Corinne Lucet*
Search space reduction in MILP approaches for the robust balancing of transfer lines *Aleksandr Pirogov, André Rossi, Evgeny Gurevsky and Alexandre Dolgui*	**Planning problem in Healthcare domain** *Olivier Gérard, Laure Brisoux Devendeville and Corinne Lucet*
Decision trees for robust scheduling *Tom Portoleau, Christian Artigues and Romain Guillaume*	**Optimization of order for containers placement schedule in rail terminal operations** *Nadiia Kalaida, Remy Dupas and Igor Grebennik*
A Stochastic Programming Model to Schedule Projects under Cash Flow Uncertainty *Berfin Kutlag, Nazli Kalkan Nazli, Serhat Gul and Oncu Hazir*	

12:20 PM – 1:30 PM: Lunch break

1:30 PM – 3:10 PM: Parallel session F3

Room 1: Flowshop scheduling chair: Federico Della Croce	Room 2: RCPSP 2 chair: Massimiliano Caramia
An acceleration procedure for several objective functions in the permutation flow shop scheduling problem *Victor Fernandez-Viagas, José M. Molina-Pariente, Carla Talens and José M. Framiñán*	**New benchmark datasets for the RCMPSP** *Rob Van Eynde and Mario Vanhoucke*
An Inclusion-Exclusion based algorithm for the permutation flowshop scheduling problem *Olivier Ploton and Vincent T'kindt*	**A new tool for analysing and reporting solutions for the RCPSP and MMRCPSP** *José Coelho, Mario Vanhoucke and Ricardo Amaro*
Exact solution of the two-machine flow shop problem with 3 operations *Federico Della Croce, Fabio Salassa and Vincent T'kindt*	**An analysis of critical alternatives in the RCPSP-AS** *Tom Servranckx and Mario Vanhoucke*
Non-dominated sorting genetic algorithm for a bi-objective flexible flow shop problem. A Case Study *Ibeth Rodriguez Grattz, José-Fernando Jiménez, Eliana Marvé Gonzalez, Eduardo Puerto, Yenny Paredes and Juan Caballero*	**Adapting the RCPSP framework to Evacuation Problems** *Christian Artigues, Alain Quilliot, Hélène Toussaint and Peter Stuckey*
Scheduling to minimize maximum lateness in tree data gathering *Joanna Berlinska*	**On the Activity Criticality in Project Scheduling with Generalized Precedence Relationships** *Lucio Bianco, Massimiliano Caramia and Stefano Giordani*

3:10 PM – 3:30 PM: Coffee break

3:30 PM – 4:30 PM: Plenary session 4

Modeling and solving complex job-shop scheduling problems by Stéphane Dauzère-Pérès

chair: Chris Potts

4:30 PM – 5:00 PM: Closing session

Plenary Talk: Robustness in Scheduling

Marjan van den Akker (Utrecht University, The Netherlands)

Wednesday, April 21, 9:20 am-10:20 am (UTC+2:00)
Chair: **Erwin Pesch** (Universität Siegen, Germany)

Abstract Traditionally, scheduling is based on the assumption that all input parameters are deterministic. Since in real-life situations uncertainty is unavoidable, recently more attention has been given to scheduling with stochastic processing times. Our goal is to obtain robust solutions for stochastic scheduling problems. To the best of our knowledge, no universal formal definition of robustness exists. An intuitive definition is: a schedule which does not significantly degrade in the face of disruption is called robust. In this presentation we discuss different models and algorithms for robustness in scheduling with an emphasis on parallel machine scheduling with precedence constraints.

About Marjan After her PhD in scheduling algorithms supervised by Jan Karel Lenstra, Marjan van den Akker has worked CORE (Louvain-la-Neuve) as postdoc and at the Netherlands Aerospace Centre NLR as expert on modelling, optimization, and simulation in Air Traffic Management and Electronic Road Pricing. Since 2000, she is at the Department of Information and Computing Sciences at Utrecht University. Her research area is advanced algorithms, robustness and simulation. In her research characteristics from practice are combined with state of the art theoretical models. When solving optimization problems, important side-constraints are included as much as possible, even though this increases the complexity: the goal is to push the boundary of what is computable as far as possible. To achieve this, her research includes the application of LP-based decomposition techniques such as column generation, as well as approaches using local search. The last few years, she has focused on planning and scheduling under uncertainty. Traditionally, scheduling and planning problems were modeled as problems with fixed (deterministic) data. Since in real-life situations disturbances occur frequently, robustness is receiving an increasing amount of attention. The purpose is to develop algorithms for finding solutions that, either remain valid in case of a disturbance, or can easily be adjusted to a feasible solution without having to solve the problem all over again. The question is "How can we capture uncertainty as well and efficient as possible in a deterministic model?" Her work includes applications in energy networks and public transportation, where she is supervising different PhD students including projects with companies. Her work is published in well-known international journals and conferences in Computer Science and Operations Research. She is a board member of the Dutch Network on the Mathematics of Operations Research (LNMB).

Industrial Plenary Talk: Industrial project and machine scheduling with Constraint Programming

Philippe Laborie (IBM France Lab, Gentilly, France)

Wednesday, April 21, 3:30 pm-4:30 pm (UTC+2:00)
Chair: **Pierre Lopez** (LAAS-CNRS, Toulouse, France)

Abstract More often than not, project and machine scheduling problems are addressed either by generic mathematical programming techniques (like MILP) or by problem-specific exact or heuristic approaches. MILP formulations are commonly used to describe the problem in mathematical terms and to provide optimal solutions or bounds to small problem instances. As they usually do not scale well, one usually resorts to using heuristics for handling large and complex industrial problems. Though constraint programming (CP) techniques represent the state of the art in several classical project and machine scheduling benchmarks and have been used for almost 30 years for solving industrial problems, they are still seldom considered as an alternative approach in the scheduling community. A possible reason is that, for years, in the absence of efficient and robust automatic search algorithms, CP techniques have been difficult to use for non-CP experts. We will explain why we think this time is over and illustrate our arguments with CP Optimizer, a generic system, largely based on CP, for modeling and solving real-world scheduling problems. CP Optimizer extends linear programming with an algebraic language using simple mathematical concepts (such as intervals, sequences and functions) to capture the temporal dimension of scheduling problems in a combinatorial optimization framework. CP Optimizer implements a model-and-run paradigm that vastly reduces the burden on the user to understand CP or scheduling algorithms: modeling is by far the most important. The automatic search combines a wide variety of techniques from Artificial Intelligence (constraint programming, temporal reasoning, learning etc.) and Operations Research (mathematical programming, graph algorithms, local search, etc.) in an exact algorithm that provides good performance out of the box and which is continuously improving.

About Philippe Philippe Laborie is a Principal Scientist at IBM. He is one of the main designers of the mathematical modeling language for scheduling problems offered in CPLEX Optimization Studio and a significant contributor to the underlying automatic search algorithm. He graduated from Telecom ParisTech in 1992, and received a PhD in Artificial Intelligence from LAAS/CNRS (Toulouse) on the integration of Artificial Intelligence Planning and Scheduling in 1995. Before joining IBM/ILOG in 1998, he worked at Electricité de France (Paris) and INRIA/IRISA (Rennes) on the Supervision and Diagnosis of complex systems (telecommunication and power distribution networks). His main scientific interests include planning, scheduling, supervision and diagnosis of complex systems and more generally, all decision problems dealing with time. He received the 2011 ICAPS Influential paper award. Philippe is member of the editorial board of the Journal of Artificial Intelligence Research and serves in the Program Committee of many conferences in AI (IJCAI, AAAI, ECAI, ICAPS, CP, CPAIOR, ...).

Plenary Talk: Data driven Project Management

Mario Vanhoucke (Ghent University, Belgium)

Thursday, April 22, 3:30 pm-4:30 pm (UTC+2:00)
Chair: **Sigrid Knust** (Universität Osnabrück, Germany)

Abstract This presentation will give an overview of the past endeavours and the recent trends in integrated project management and control, with a focus on linking scheduling to risk and control management. The presentation will show the relevance of using artificial and real project data for both research and practice. It will be shown that not many organisations have as much data as they often claim, and researchers therefore have to fall back on the use of artificial project data. Consequently, an overview of the most important artificial project datasets (each with advantages and disadvantages) will be given, and also a new set of empirical projects (freely available to researchers) will be presented. Some recent research trends will be highlighted, illustrating that the integrated use of empirical data and advanced techniques (machine learning) might lead to promising results, and should therefore define the path for future research. References to a literature overview of project control will be given to outline the future research on integrated project management and control. During my talk, I will also present my nice team of young researchers to you!

About Mario Prof Dr Mario Vanhoucke is a Full Professor at Ghent University (Belgium), Vlerick Business School (Belgium) and UCL School of Management (University College London, UK). He teaches "Project Management", "Applied Operations Research" and "Decision Making for Business". He obtained a Master's Degree in Business Engineering (1996) and a PhD in Operations Management (2001), and he was director of EVM Europe (www.evm-europe.eu) and partner at the company OR-AS (www.or-as.be) until 2018. Mario is responsible for various research projects in the field of Integrated Project Management and Control, which has led to more than 60 papers in international journals, five Project Management books published by Springer, three free online books (www.or-as.be/books), three computerised business games and an online learning platform known as PM Knowledge Center (www.pmknowledgecenter.com). His research has received multiple awards, including awards from PMI Belgium (Belgium, 2007), International Project Management Association (Italy, 2008) and the American Accounting Association (USA, 2010) and multiple awards from Belgian companies. He currently has a team of +10 enthusiastic PhD students who are jointly working on improving decision making in project management, with a strong focus on (1) developing methods to improve project scheduling, (2) analysis risk in projects and (3) validating current and newly developed statistical methods for project control.

Plenary Talk: Modeling and solving complex job-shop scheduling problems

Stéphane Dauzère-Pérès (Ecole des Mines de Saint-Etienne, France)

Friday, April 23, 3:30 pm-4:30 pm (UTC+2:00)
Chair: **Chris Potts** (University of Southampton, Great Britain)

Abstract This talk focuses on the flexible job-shop scheduling problem, first extensively studied in the 1990's, which was later extended to include additional constraints and criteria to become complex job-shop scheduling problems. The complexity related to the problems is first discussed, in particular by differentiating with the classical job-shop scheduling problem. The main characteristic of the flexible job-shop scheduling problem is that operations can be performed on several resources, i.e. that operations must be both assigned to and sequenced on resources. Modelling choices and solution methods will be surveyed, including some recent contributions related to the consideration of batching constraints, sequence-dependent setup times and multiple criteria. Some of the results are based on a long-term collaboration of more than 15 years with two manufacturing sites of the French-Italian semiconductor company STMicroelectronics.

About Stéphane Stéphane Dauzère-Pérès is Professor at Mines Saint-Etienne in its site of Gardanne, France, and Adjunct Professor at BI Norwegian Business School, Norway. He received the Ph.D. degree from Paul Sabatier University in Toulouse, France, in 1992 and the H.D.R. from Pierre and Marie Curie University, Paris, France, in 1998. He was a Postdoctoral Fellow at the Massachusetts Institute of Technology, U.S.A., in 1992 and 1993, and Research Scientist at Erasmus University Rotterdam, The Netherlands, in 1994. He has been Associate Professor and Professor from 1994 to 2004 at the Ecole des Mines de Nantes, France, where he headed the team "Production and Logistic Systems" between 1999 and 2004. He was invited Professor at the Norwegian School of Economics and Business Administration (NHH), Norway, in 1999. Since March 2004, he is Professor at Mines Saint-Etienne, where he headed the research department "Manufacturing Sciences and Logistics" from 2004 to 2013. His research interests broadly include modeling and optimization of operations at various decision levels (from real-time to strategic) in manufacturing and logistics, with a special emphasis on production planning (lot sizing) and scheduling and on semiconductor manufacturing. He has published nearly 80 papers in international journals and has contributed to more than 200 communications in national and international conferences. Stéphane Dauzère-Pérès has coordinated numerous academic and industrial research projects, including 4 European projects and 24 industrial (CIFRE) PhD theses, and also six conferences. In particular, he co-organized in 2010 the first edition of the International Workshop on Lot Sizing which was held in Gardanne, France. In 2014, he created with Bernardo Almada-Lobo (University of Porto, Portugal) the EURO Working Group on Lot-Sizing (LOT), that he coordinated until 2018. He was runner-up in 2006 of the Franz Edelman Award Competition, and won the Best Applied Paper of the Winter Simulation Conference in 2013. His h-index is 36.

Heuristics for Scheduling Pipe-laying Support Vessels: An Identical Parallel Machine Scheduling Approach

Victor Abu-Marrul[1], Davi Mecler[1], Rafael Martinelli[1], Silvio Hamacher[1] and Irina Gribkovskaia[2]

[1] Industrial Engineering Department, Pontifical Catholic University of Rio de Janeiro, Brazil
{victor.cunha,davizm}@tecgraf.puc-rio.br, {martinelli,hamacher}@puc-rio.br
[2] Molde University College - Specialized University in Logistics, Norway
irina.gribkovskaia@himolde.no

Keywords: Ship Scheduling, Offshore Logistics, Heuristic, Parallel Machine Scheduling.

1 Introduction and Problem Description

We address a problem that emerges in oil & gas offshore logistics where a company needs to schedule its fleet of pipe laying support vessels (PLSV), responsible for connecting sub-sea oil wells to production platforms. We model it as an identical parallel machine scheduling problem with non-anticipatory family setup times. Here, the vessels are machines, jobs are wells, and each job includes a number of connecting operations. The problem consists of scheduling a set $\mathcal{O}$ of operations from a set $\mathcal{F}$ of families in a set $\mathcal{M}$ of machines. Each operation $i \in \mathcal{O}$ belongs to a family (f_i), has a release date (r_i), a processing time (p_i), a load occupancy (l_i), can be assigned to an eligible subset of machines $(\mathcal{M}_i)$, and is associated with a subset of jobs $(\mathcal{N}_i)$. Each job $j \in \mathcal{N}$ has a weight (w_j) according to the related well production rate, and a subset $\mathcal{O}_j$ of the operations associated with it. Each machine $k \in \mathcal{M}$ has a release date (r_k) and a capacity (q_k). A non-anticipatory family setup time (s_f) is incurred before the first operation on each machine, whenever a machine changes the execution of operations between families, and when the capacity of a machine is reached. A set of operations that shares the same setup time is called a batch. Setups are non-anticipatory since they can only start when all operations scheduled inside a batch are released. The sum of the load occupancy of the operations assigned to a batch must respect the machine capacity. The objective is to minimize the total weighted completion time of jobs $(\sum_{j \in \mathcal{N}} w_j C_j)$, where the completion time of job j (C_j) is the maximum completion time of the associated operations $(C_j = \max_{i \in \mathcal{O}_j} C_i)$. This objective aims to complete the connections of the most productive oil wells as soon as possible. An example of a PLSV schedule is depicted in Figure 1 with 15 operations, 5 jobs, and 4 machines. The associated jobs are shown in the boxes, and their completion times are marked with dotted lines.

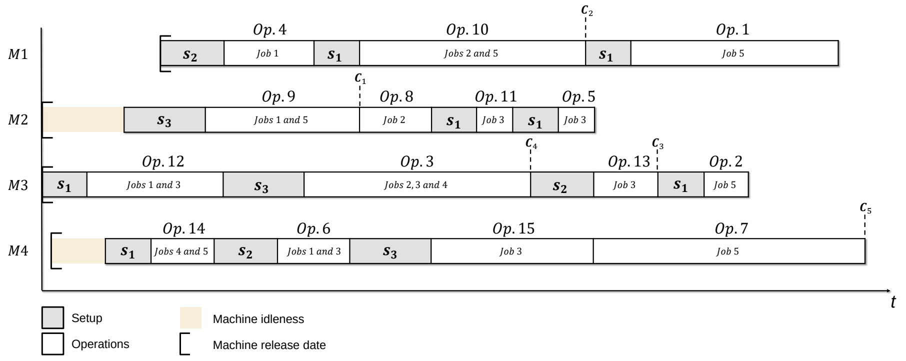


Fig. 1: PLSV Scheduling example with 15 operations, 5 Jobs and 4 machines.

2 Solution Methods

We present several constructive heuristics to solve the PLSV scheduling problem, using dispatching rules, and defining how to do the machine assignment and to construct batches. The schedule construction procedure used in all heuristics is presented in Algorithm 1. To describe the procedure, we introduce variables and sets: S_k (Start time of the current batch on machine k), L_k (Cumulative load of the current batch on machine k), F_k (Family of the current batch on machine k), C_k (Completion time of machine k), *same* (Boolean variable that defines if the chosen operation will be assigned to the last position in the current batch or to a new batch), $\mathcal{B}_k$ (Set of operations assigned to the current batch on machine k) and $\mathcal{U}$ (Set of unscheduled operations). The procedure returns a list of schedules $\sigma = (\sigma_1, \ldots, \sigma_k)$ for each machine k containing operations and families (representing the setup times).

Algorithm 1: Schedule Construction Procedure

```
 1: $C_k \leftarrow r_k, S_k \leftarrow r_k,\ L_k \leftarrow 0,\ F_k \leftarrow 0,\ \mathcal{B}_k \leftarrow \emptyset, \sigma_k \leftarrow \emptyset:\ \forall k \in \mathcal{M}$
 2: $C_i \leftarrow \infty:\ \forall i \in \mathcal{O}$
 3: $\mathcal{U} \leftarrow \mathcal{O}$
 4: while there exists operations not assigned do
 5:    Select operation $i^* \in \mathcal{U}$ and machine $k^* \in \mathcal{M}_{i^*}$ according to a chosen heuristic, defining
       same as true or false
 6:    if same = true then
 7:       $S_{k^*} \leftarrow \max(r_{i^*}, S_{k^*}),\ C_{k^*} \leftarrow C_{k^*} + \max(0, r_{i^*} - S_{k^*}) + p_{i^*}$
 8:       $L_{k^*} \leftarrow L_{k^*} + l_{i^*},\ \mathcal{B}_{k^*} \leftarrow \mathcal{B}_{k^*} \cup \{i^*\}$
 9:    else
10:       $S_{k^*} \leftarrow \max(r_{i^*}, C_{k^*}),\ C_{k^*} \leftarrow \max(r_{i^*}, C_{k^*}) + s_{f_{i^*}} + p_{i^*}$
11:       $L_{k^*} \leftarrow l_{i^*},\ \mathcal{B}_{k^*} \leftarrow \{i^*\},\ \sigma_{k^*} \leftarrow \sigma_{k^*} \cup \{f_{i^*}\}$
12:    end if
13:    $\sigma_{k^*} \leftarrow \sigma_{k^*} \cup \{i^*\},\ F_{k^*} \leftarrow f_{i^*},\ C_{i^*} \leftarrow C_{k^*},\ \mathcal{U} \leftarrow \mathcal{U} \setminus \{i^*\}$
14: end while
15: return $\sigma$
```

The main part of the method is defined at line 5, the decision of the next operation, machine, and batch composition. We consider two approaches at this step. In both, when we evaluate the insertion of an operation to an existing batch, we consider the delay on the start time of this batch that may occur due to the inserted operation release date and the non-anticipatory setup consideration. This delay changes the completion time of all operations scheduled in the batch, and it is penalized using the operations weights. We develop rules to estimate weights for the operations (w_i), since weights in our problem are related to jobs. Five rules for estimating weights are considered:

- `MAX`: Maximum weight of associated wells, computed as $w_i = \max_{j \in \mathcal{N}_i} w_j$
- `SUM`: Sum of the weights of associated wells, computed as $w_i = \sum_{j \in \mathcal{N}_i} w_j$
- `AVG`: Average weight of associated wells, computed as $w_i = \sum_{j \in \mathcal{N}_i} w_j / |\mathcal{N}_i|$
- `WAVG`: Weighted average weight of associated wells, computed as $w_i = \sum_{j \in \mathcal{N}_i} w_j / |\mathcal{O}_j|$
- `WAVGA`: `WAVG` adjusted at each iteration by the set of unscheduled operations (i.e., it replaces subsets $\mathcal{O}_j$ by subsets $\mathcal{U}_j$ of unscheduled operations associated to a job j).

In the first approach, we initially select the next operation based on a dispatching rule, and then assign an eligible machine to this operation according to the minimum weighted completion time, as described in Algorithm 2. New variables and sets are defined: Δ_{ik} (Delay at the start time of the current batch on machine k due to the insertion of operation i), C_{ik}^{CB} (Completion time of operation i if inserted in the current batch on machine k), C_{ik}^{NB} (Completion time of operation i if inserted in a new batch on machine k), $\mathcal{CB}$ (Set of feasible assignments cb_{ik} of operation i into the current batch on machine k) and $\mathcal{NB}$ (Set of feasible assignments nb_{ik} of operation i into a new batch on machine k).

Algorithm 2: Operation and Machine Disjunctive Selection Procedure

```
1: Select operation $i^* \in \mathcal{U}$ according to a chosen dispatching rule
2: $\Delta_{i^*k} \leftarrow \max(0, r_{i^*} - S_k) : \ \forall k \in \mathcal{M}_{i^*}$
3: $C^{CB}_{i^*k} \leftarrow C_k + \Delta_{i^*k} + p_{i^*} : \ \forall k \in \mathcal{M}_{i^*}$
4: $C^{NB}_{i^*k} \leftarrow \max(r_{i^*}, C_k) + s_{f_{i^*}} + p_{i^*} : \ \forall k \in \mathcal{M}_{i^*}$
5: $\mathcal{CB} \leftarrow \left\{ cb_{i^*k} = w_{i^*} C^{CB}_{i^*k} + \sum_{\hat{\imath} \in \mathcal{B}_k} w_{\hat{\imath}} \Delta_{i^*k} \mid k \in \mathcal{M}_{i^*},\ F_k = f_{i^*},\ L_k + l_{i^*} \leq q_k \right\}$
6: $\mathcal{NB} \leftarrow \left\{ nb_{i^*k} = w_{i^*} C^{NB}_{i^*k} \mid k \in \mathcal{M}_{i^*} \right\}$
7: $b_{min} \leftarrow \min\{b : b \in (\mathcal{CB} \cup \mathcal{NB})\}$
8: Select $k^*$ corresponding to $b_{min}$
9: if $b_{min} \in \mathcal{CB}$ then  $same \leftarrow$ true
10: return $i^*, k^*, same$
```

We consider six dispatching rules for this approach. The priority value π_i indicates the next operation to schedule. At each iteration, the operation i with the largest π_i value is selected (Đurasević and Jakobović 2018). We adapt some rules by adding family setup times, and assume that every operation will be assigned to a new batch. T_i is the minimum completion time among the eligible vessels for each operation i, and is computed as $T_i = min_{k \in \mathcal{M}_i} C_k$. The following dispatching rules are considered:

- `ERD`: Earliest Release Date $\quad \pi_i = 1/r_i$.
- `SPT`: Shortest Processing Time $\quad \pi_i = 1/p_i$.
- `LPT`: Longest Processing Time $\quad \pi_i = p_i$.
- `MCT`: Minimum Completion Time $\quad \pi_i = \max(T_i, r_i) + p_i + s_{f_i}$.
- `WSPT`: Weighted Shortest Processing Time $\quad \pi_i = w_i/p_i$.
- `WMCT`: Weighted Minimum Completion Time $\quad \pi_i = [\max(T_i, r_i) + p_i + s_{f_i}]/w_i$.

The second approach extends one of the heuristics from Weng et al.(2001), by considering the PLSV scheduling properties, such as the release dates of operations and machines, the family setup times and the batch composition, to decide at each iteration the next pair operation/machine simultaneously (Algorithm 3). We call it `WMCT-Pair`.

Algorithm 3: Operation and Machine Simultaneous Selection Procedure

```
1: $\Delta_{ik} \leftarrow \max(0, r_i - S_k) : \ \forall i \in \mathcal{U},\ k \in \mathcal{M}_i$
2: $C^{CB}_{ik} \leftarrow C_k + \Delta_{ik} + p_i : \ \forall i \in \mathcal{U}, k \in \mathcal{M}_i$
3: $C^{NB}_{ik} \leftarrow \max(r_i, C_k) + s_{fi} + p_i : \ \forall i \in \mathcal{U}, k \in \mathcal{M}_i$
4: $\mathcal{CB} \leftarrow \left\{ cb_{ik} = \frac{C^{CB}_{ik}}{w_i} + \sum_{\hat{\imath} \in \mathcal{B}_k} \frac{\Delta_{ik}}{w_{\hat{\imath}}} \mid i \in \mathcal{U},\ k \in \mathcal{M}_i,\ F_k = f_i,\ L_k + l_i \leq q_k \right\}$
5: $\mathcal{NB} \leftarrow \left\{ nb_{ik} = \frac{C^{NB}_{ik}}{w_i} \mid i \in \mathcal{U},\ k \in \mathcal{M}_i \right\}$
6: $b_{min} \leftarrow \min\{b : b \in (\mathcal{CB} \cup \mathcal{NB})\}$
7: Select $i^*$ and $k^*$ corresponding to $b_{min}$
8: if $b_{min} \in \mathcal{CB}$ then  $same \leftarrow$ true
9: return $i^*, k^*, same$
```

3 Computational Experiments

We introduce in total 19 heuristics, where 4 do not consider weights and 15 combine the dispatching rules and the ways of estimating the operations' weights. We tested all of them on a set of 72 PLSV instances[3] with $|\mathcal{M}| = \{2, 4\}$, and $|\mathcal{O}| = \{15, 25, 50\}$. We performed

[3] available at https://doi.org/10.17771/PUCRio.ResearchData.45799

the experiments on a computer with 64 GB of RAM and Intel Core i7-8700K CPU of 3.70GHz, using C++ for coding the heuristics and running Linux. The results of tests, in terms of the average relative deviations from the best generated solutions, are presented in Table 1. Each instance group, defined by the number of operations and machines, contains 12 instances. The relative deviation is computed as $RD^h_{inst} = TWC^h_{inst}/TWC^{best}_{inst}$, where TWC^h_{inst} is the total weighted completion time of heuristic $h \in \mathcal{H}$ applied to instance $inst \in \mathcal{I}$, and TWC^{best}_{inst} is the best solution obtained for a given instance. The best result for each instance group is shown in bold. All heuristics run in less than 0.1 seconds. Last column ($\#BKS$) accounts how many times each heuristic yields the best solution.

Table 1: Average deviations from the best solutions.

Heuristic	**Instance Group ($\|\mathcal{O}\| - \|\mathcal{M}\|$)**						**All Instances**	**#BKS**
	15-4	**15-8**	**25-4**	**25-8**	**50-4**	**50-8**		
`ERD`	1.212	1.198	1.245	1.190	1.261	1.250	1.226	2
`SPT`	1.278	1.217	1.363	1.296	1.442	1.392	1.331	1
`LPT`	1.347	1.153	1.412	1.265	1.471	1.398	1.341	0
`MCT`	1.226	1.165	1.268	1.248	1.254	1.295	1.243	0
`WSPT-MAX`	1.161	1.079	1.224	1.173	1.299	1.255	1.198	1
`WSPT-SUM`	1.156	1.066	1.181	1.141	1.280	1.241	1.178	0
`WSPT-AVG`	1.181	1.085	1.266	1.187	1.327	1.294	1.223	1
`WSPT-WAVG`	1.124	1.076	1.184	1.124	1.234	1.196	1.156	0
`WSPT-WAVGA`	1.085	1.041	1.112	1.067	1.138	1.114	1.093	3
`WMCT-MAX`	1.084	1.040	1.070	1.088	1.046	1.096	1.071	7
`WMCT-SUM`	1.063	1.046	1.093	1.082	1.158	1.132	1.096	3
`WMCT-AVG`	1.101	1.027	1.123	1.118	1.150	1.147	1.111	4
`WMCT-WAVG`	1.065	1.036	1.041	1.059	1.111	1.084	1.066	5
`WMCT-WAVGA`	**1.023**	**1.021**	**1.013**	**1.016**	1.027	**1.003**	**1.017**	34
`WMCT-Pair-MAX`	1.086	1.043	1.063	1.082	1.036	1.091	1.067	9
`WMCT-Pair-SUM`	1.063	1.043	1.087	1.071	1.151	1.124	1.090	4
`WMCT-Pair-AVG`	1.101	1.027	1.123	1.107	1.136	1.136	1.105	2
`WMCT-Pair-WAVG`	1.060	1.035	1.045	1.050	1.100	1.076	1.061	7
`WMCT-Pair-WAVGA`	1.029	1.023	1.024	1.018	**1.026**	1.005	1.021	19

Note that among the heuristics, `WCMT-WAVGA` generated the best average solutions for 5 of 6 groups with the best average deviation of 1.003, achieved on group 50-8. This heuristic also found the highest number of best solutions, on 34 of 72 instances.

4 Conclusions

We studied an identical parallel machine scheduling problem with non-anticipatory family setup times, derived from a ship scheduling problem in the offshore oil & gas logistics. Tests of the 19 heuristics presented on all instances show that the heuristic `WCMT-WAVGA` performs better, with an average deviation of 1.017 from the best solutions. For future work, local searches and meta-heuristics will be developed.

References

Đurasević, M; Jakobović, D. "A survey of dispatching rules for the dynamic unrelated machines environment", *Expert Systems with Applications*, Vol. 113, pp. 555-569, 2018.

Weng, M.; Lu, J.; Ren, H. "Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective", *International journal of production economics*, Vol. 70, pp. 215-226, 2001.

Duplication and sequencing of unreliable jobs

A. Agnetis[1], P. Detti[1], B. Hermans[2] and M. Pranzo[1]

[1] Università degli studi di Siena,
Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, Siena, Italy
{agnetis,detti,pranzo}@diism.unisi.it
[2] Katholieke Universiteit Leuven,
Research Centre for Operations Research and Business Statistics (ORSTAT), Leuven, Belgium
ben.hermans@kuleuven.be

Keywords: unreliable jobs; job duplication; parallel machines; complexity.

1 Introduction

This paper considers a scenario in which a given set of n *jobs* must be processed on a machine subject to possible *breakdowns*. If a breakdown occurs, the remaining jobs (including the job currently being processed) cannot be performed and are therefore lost. On the other hand, if job j is successfully completed, a revenue r_j is gained. We denote the success probability of job j as p_j. No rescheduling or reactions are possible, hence a *preventive disruption management* perspective is adopted (Qi *et al.* 2006), and the problem is to decide the job sequence before the beginning of the actual execution of the sequence, in such a way that the expected revenue is maximized. In what follows, we denote this problem as 1ERM (i.e., expected revenue maximization on a single machine). Given a sequence σ of jobs on the machine, and denoting the i-th scheduled job as $\sigma(i)$, the expected revenue $ER[\sigma]$ is given by

$$ER[\sigma] = p_{\sigma(1)}r_{\sigma(1)} + p_{\sigma(1)}p_{\sigma(2)}r_{\sigma(2)} + \ldots + p_{\sigma(1)} \cdots p_{\sigma(n-1)}p_{\sigma(n)}r_{\sigma(n)}. \tag{1}$$

It is known that 1ERM can be solved at optimality (Mitten 1960) by sequencing jobs from the greatest to the smallest *Z-ratio*:

$$Z_j = \frac{p_j r_j}{(1-p_j)}. \tag{2}$$

Here we address an extension of 1ERM in which there are m (identical) machines in charge of job processing. In order to hedge against unrecoverable interruptions, we adopt the technique of *duplicating* work on different machines, which is a common strategy in many computing centers (Zhou *et al.* 2016, Benoit *et al.* 2013). In particular, this means that there are *m copies* of each job, one to be executed on each machine. The revenue r_j is gained if *at least one copy* of j is successfully carried out, i.e., even if all copies are completed, the revenue is attained only once. When no job duplication is allowed, problems on parallel machines have been addressed in (Agnetis *et al.* 2017, Agnetis *et al.* 2009, Lee and Yu 2008).

The problem addressed here may arise when considering the execution of a set of tasks on a multi-processor environment, composed by different servers geographically distributed. In general we can assume that servers may fail, connections can be broken and outages may occur. A strategy to increase reliability is to duplicate the execution of the tasks on two or more independent servers possibly in different geographical locations so that in case of failure of a server the computation is still carried out on the other servers. In any case, the revenue is gained only once when one of the computations is over.

Specifically, we address the following problems.

Definition 1. *EXPECTED REVENUE MAXIMIZATION WITH TWO MACHINES (2ERM) – Given n jobs $\{1, 2, \ldots, n\}$, each having success probability p_j and revenue r_j, and two identical machines M_1 and M_2, find a sequence of the n jobs on M_1 and M_2 so that the expected revenue of attaining at least one copy of each job is maximized.*

Definition 2. *KIT AVAILABILITY MAXIMIZATION (mKAM) – Given n jobs $\{1, 2, \ldots, n\}$, each having success probability p_j, and m identical machines, find a sequence of the n jobs on each machine so that the probability of attaining at least one copy of each job is maximized.*

We discuss the problem complexity and introduce a rule based on a modified Z-ratio (2) that, given the sequence on machine M_1, allows to derive an optimal sequence for M_2. Furthermore, when the Z-ratios (2) of the jobs are all equal to 1, the rule allows to easily build an optimal solution for 2ERM.

2 The Expected Revenue Maximization Problem with two machines

We let $P = \prod_{j=1}^{n} p_j$ and assume that $p_j < 1$ for all j (if for some job $p_j = 1$, such a job is obviously processed first with no consequence on the other jobs). Let us first state a result concerning the following situation. Suppose that a job sequence $\bar{\sigma}_1$ has been fixed on M_1, and we let $\bar{p}_j$ denote the cumulative probability up to job j on M_1 in sequence $\bar{\sigma}_1$, i.e.,

$$\bar{p}_j = \prod_{k:k\prec j} p_k.$$

Moreover, we denote by Z'_j the *modified Z-ratio* of job j, defined as

$$Z'_j = Z_j(1 - \bar{p}_j). \tag{3}$$

The problem of finding the optimal sequence on M_2 given $\bar{\sigma}_1$ is solved as shown in the following lemma:

Lemma 1. *If a job sequence $\bar{\sigma}_1$ is fixed on M_1, expected revenue is maximized sequencing the jobs on M_2 by nonincreasing values of*

$$Z'_j = Z_j(1 - \bar{p}_j). \tag{4}$$

Proof. The proof uses an interchange argument. Consider a sequence σ_2 on M_2, and let P_i be the cumulative success probability of job i in σ_2, i.e., $P_i = p_i \prod_{k:k\prec i} p_k$. (Note that P_i includes the probability of job i itself.) Given a fixed sequence $\bar{\sigma}_1$ on M_1 and the associated probabilities $\bar{p}_i$ and $\bar{p}_j$, assume that in σ_2 there are two consecutive jobs j and i such that $j \prec i$ and $Z'_i > Z'_j$. Let σ'_2 be the sequence obtained swapping i and j in σ_2. The expected revenue of $(\bar{\sigma}_1, \sigma'_2)$ can be expressed as

$$ER(\bar{\sigma}_1, \sigma'_2) = A + r_i(P_i + \bar{p}_i - P_i\bar{p}_i) + r_j(P_ip_j + \bar{p}_j - P_ip_j\bar{p}_j) + B$$

while

$$ER(\bar{\sigma}_1, \sigma_2) = A + r_j(P_j + \bar{p}_j - P_j\bar{p}_j) + r_i(P_jp_i + \bar{p}_i - P_jp_i\bar{p}_i) + B,$$

where A and B denote the contribution of jobs preceding and respectively following i and j on M_2 in the two schedules. Denoting with Q the cumulative probability of jobs preceding i and j on M_2, in $(\bar{\sigma}_1, \sigma'_2)$ one has $P_i = Qp_i$ and in $(\bar{\sigma}_1, \sigma_2)$, $P_j = Qp_j$, one has that $ER(\bar{\sigma}_1, \sigma'_2) - ER(\bar{\sigma}_1, \sigma_2) > 0$ if and only if

$$r_ip_i - r_ip_i\bar{p}_i + r_jp_ip_j - r_jp_ip_j\bar{p}_j - (r_jp_j - r_jp_j\bar{p}_j + r_ip_jp_i - r_ip_jp_i\bar{p}_i) > 0,$$

i.e.,

$$r_i p_i(1-\bar{p}_i)(1-p_j) > r_j p_j(1-\bar{p}_j)(1-p_i),$$

and hence

$$Z_i(1-\bar{p}_i) > Z_j(1-\bar{p}_j),$$

which holds since $Z'_i > Z'_j$. By repeatedly applying the above argument, the thesis follows. □

A consequence of Lemma 1 is the following.

Lemma 2. *Consider an instance of 2ERM in which $Z_j = 1$ for all jobs $j = 1, \dots, n$. Then any schedule in which the jobs are reversely sequenced on the two machines is optimal.* □

Regarding the computational complexity of the problem, we recall that when duplications are not allowed, the problem with 2 machines and unreliable jobs is known to be strongly NP-hard (Agnetis *et. al.* 2009). Concerning 2ERM, it is possible to prove the following result.

Theorem 1. *2ERM is strongly NP-hard.* □

The proof consists in showing that the combinatorial problem PRODUCT PARTITION can be polynomially reduced to 2ERM. PRODUCT PARTITION was proved strongly NP-hard by (Ng *et al.* 2010).

3 The Kit Availability Maximization Problem

The following results can be established for KAM.

Theorem 2. *When there are two machines and n job types (2KAM), the problem can be solved in $O(n)$.*

The problem in which there are m machines and only two job types (1 and 2) consists in deciding the number x of machines that follow the sequence 12, so that $m-x$ will follow the sequence 21. The following result can be established.

Theorem 3. *mKAM with two job types can be solved in $O(\log m)$.* □

References

Agnetis, A., Detti, P., Martineau, P., Scheduling nonpreemptive jobs on parallel machines subject to exponential unrecoverable interruptions, *Computers and Operations Research*, 79, 109-118, 2017.

Agnetis, A., Detti, P., Pranzo, M., Sodhi, M.S., Sequencing unreliable jobs on parallel machines, *Journal of Scheduling*, 12(1), 45-54, 2009.

Benoit, A., Robert, Y., Rosenberg, A.L., Vivien, F., Static Strategies for Worksharing with Unrecoverable Interruptions, *Theory of Comput Systems*, 2013,53,386-423.

Lee, C.-Y., Yu, G., Parallel-machine scheduling under potential disruption, Optimization Letters, 2, 27-37, 2008.

Lee, C.-Y., Yu, G., Single machine scheduling under potential disruption, Operations Research Letters, 35, 541-548, 2007.

Mitten, L.G., An Analytic Solution to the Least Cost Testing Sequence Problem, J. Industrial Engineering, 11, 17, 1960.

Ng, C.T., Barketau, M.S., Cheng, T.C.E., Kovalyov, M.Y., ”Product partition” and related problems of scheduling and systems reliability: Computational complexity and approximation, *European Journal of Operational Research*, 207, 601-604, 2010.

Qi, X., Bard, J.F., Yu, G., Disruption management for machine scheduling: The case of SPT schedules, *International Journal of Production Economics*, 103, 166-184, 2006.

Zhou, A., Wang, S., Cheng, B., Zheng, Z., Yang, F., Chang, R.N., Lyu, M.R., Buyya, R., Cloud service reliability enhancement via virtual machine placement optimization, *IEEE Transactions on Services Computing*, 10 (6), 902-913, 2016.

A Discrete Time Markov Decision Process to support the scheduling of re-manufacturing activities

Alessio Angius[1], Massimo Manzini[1] and Marcello Urgo[1]

Politecnico di Milano, Mechanical Dept., Italy
`alessio.angius, massimo.manzini, marcello.urgo@polimi.it`

Keywords: Markov Decision Process, re-manufacturing, scheduling.

1 Introduction and problem statement

Rotor blades are one of the most expensive components in gas turbines for power generation due to the specific materials used and the complex manufacturing process needed. For this reason, turbine blades are one of the component whose re-manufacturing is economically viable during the maintenance of gas turbines. Nevertheless, rework activities are subject to a considerable degree of uncertainty due to the unpredictable degree of damage affecting the blades. The wear of the rotor blades could usually occur in term of lack of material or the presence of cracks whose depth is difficult to estimate in advance. The repair process consists in the removal of the hard coating and of the damaged parts, and the addition of the missing material through additive manufacturing processes. Then, the blades have to undergo a material removal process to obtain the final desired shape. This material removal process is executed through Electrical Discharge Machining technology (EDM), operating to lots of turbine blades belonging to the same stage of the gas turbine and, thus, sharing the same geometrical features. To be able to process a lot of blades, the EDM machine has to undergo a set-up to mount the right electrode. The duration of the processing of a lot also entails a certain degree of uncertainty. Some of the blades, in fact, during the rework process, results having damages that are not possible to repair and, thus, have to be discarded. For this reason, the number of blades to be manufactured in a lot cannot be known in advance. Once the lot of blades have been manufactured in the EDM shop, it undergoes further process steps, it is integrated with new blades to complement the missing ones and then shipped to the customer's premises to be made available to the turbine, thus defining a due date to be respected.

In this paper, we will focus on the EDM shop where both new and repaired blades have to be processed. New blades follow the standard manufacturing process and the associated production plans. Repaired blades arrive as soon as the repair process has been completed and compete for the same resources, i.e., an EDM machine. Thus, a proper approach is needed to schedule the processing of both the classes of blades. We model the presence of multiple EDM machines where an already defined production plan sequences the lot of new blades to be processed while a set of lots of repaired blades is known to be about to require the same machines to be processed. The scheduling approach considers the need of a set-up to be able to process a new lot of blades and aims at minimizing the tardiness of both lots of new and repaired blades.

2 The model

The model under investigation considers K machines processing two classes of production lots: *production* and *repair*. The main difference between the two classes is that production lots are immediately available and can be scheduled on machines in advance.

On the contrary, the arrival of repair lots is uncertain, thus, any decision about their processing is delayed to the moment they become available. The production of a lot is never preempted, hence, a machine must finish the production of a lot before processing a new one. A set-up is needed to move from the production of a lot to a new one.

The model considers a time interval $[0, T]$ where P production lots are produced by knowing in advance that M repair lots will require to be processed. The scheduling of the production lots is defined and conveniently described by a vector $S = |s_1, ...s_k|$ where lots between 1 and s_1 will be produced in sequence on the first machine, lots between $s_1 + 1$ and s_2 will be produced by the second machine, and so on. Each lot is associated to a due date $d_{c,i}$ and size $l_{c,i}$ where c indicates the class and i the index of the lot.

The model assumes that the M repair lots arrive together into the system with the same due date and only one arrival is possible in $[0, T]$. The state of the system is defined by a vector $|k_1, \dots, k_K, l_{p,1}, \dots, l_{p,P}, l_{m,1}, \dots, l_{m_M}|$ with $K + P + M$ entries, where $k_i \in \{R, S\}$ (*Running* and *Set-up*) refers to the state of each machine; $l_{p,i} \in [1, K] \cup \{D\}$ represents the state of the ith production lot that can be assigned to a machine or completed (*Done*). Similarly, $l_{m,i} \in [1, K] \cup \{D, NA, A\}$ represents the ith repair lot, where the two additional states NA and A (*Not Arrived* and *Arrived*) are necessary to discriminate if the lot has arrived or not. A repair lot can be assigned to a machine only if it has been arrived.

The initial state of the system consider all the machines as running and working the firsts lots assigned in the schedule S, e.g., with $K = 3$ we will have $l_{p,1} = 1$, $l_{p,s_1+1} = 2$ and $l_{p,s_2+1} = 3$, and all the repair lots marked as NA. The model divides the time in units and assumes a syncronous system, hence, in each time unit more than one change can occur in the system. Each transition is the consequence of several events due to the the change of state of the machines and the arrival of repair lots. The arrival of repair lots changes their state from NA to A. Instead, the change of state of a machine k from R to S leads to the completion of a lot. This transition will move the system in a state where the machine will be in the state S and the corresponding lot will change state in D. In this case, if exist a $l_{m,j} = k$, $1 \leq j \leq M$, then the machine was processing a repair lot, otherwise the machine was processing the first production lot not in the state D, by following the sequence in S. Vice-versa, a machine can return to the state R from the state S. Whenever this transition is performed with both production and repair lots available, a decision must be taken, i.e., the machine has to decide if it has to follow the schedule or choose a repair lot. If machine k decides to process a repair lot, then the chosen lot will move from the state A to the state k.

The probability driving the system transitions are assumed distributed according to *phase-type distributions* (PH). It is determined by a vector α, which gives the initial probabilities of the transient states and a matrix A containing the intensities of the transitions among the transient state (see (Horváth, A. 2002)). The rates toward the absorption state are collected in a firing vector $f = -A\mathbf{1}$ where $\mathbf{1}$ is a vector of ones having the same size of the matrix A. This class of distributions is able to approximate any general distribution on the positive axis with a pre-determined accuracy whilst the overall process preserves the Markovian property. This allows us to model the distribution of the lot completion time in a statistically sound manner (as described in (Angius *et. al.* 2018)). In the following, the time that machine k requires to complete the lot i of class c follows a PH distribution represented by $(\alpha_{k,c,i}, A_{k,c,i})$ and a firing vector $f_{k,c,i}$. Similarly, the set-up times and repair arrival are distributed according to a PH distribution represented by (α_s, A_s) and (α_m, A_m).

Since transitions from a state z to a state z' are always combinations of events that involve the machines and the arrival of repair lots, any transition probability is the result of a Kronecker product of the form $B_A(z, z') \otimes_{k=1}^{K} B_k(z, z')$. The function $B_A(z, z')$ is equal to A_m if the repair lots are not arrived in both z and z', it is equal to the firing vector if the repair lots arrived in z', and it is equal to 1 otherwise. Instead, the function $B_k(z', z')$ describes the dynamic of machine k in state z. If the machine is processing a lot, this function behaves as $B_A(z, z')$ by using the corresponding values $(\alpha_{k,c,i}, A_{k,c,i})$. Otherwise, if the machine is performing the set-up, the function $B_A(z, z')$ takes values from (α_s, A_s) and a behaviour similar to the previous case, but starting the execution of a new lot after the completion of the set-up. This is done by multiplying the firing vector by the initial vector of the corresponding lot to be processed. The firing of the set-up coincides with a decision every time a machine has to choose between a production and a repair lot. Because of the presence of non-deterministic decisions, the underlying process is a Discrete Time Markov Decision Process (DTMDP) which is fully characterized by a matrix P containing all the dynamics that do not depend on decisions and a set of matrices $D_1, \ldots, D_V$ describing the different strategies in selecting the next lot to be processed in each machine.

3 Problem definition

Given the DTMDP described in Section 2, the aim of this work is to analyze the tardiness of each lot as a function of a scheduler. We define a scheduler $\mathcal{W} = |w_1, \ldots, w_T|$ as a sequence of T entries $w_i \in [1, \ldots, V]$ that determine which decision matrix has to be used in each $t \in [1, T]$. Given a scheduler $\mathcal{W}$, the distribution $\pi(t)$ of the DTMDP evolves on time according to the formula $\pi(t) = \pi(t-1)(P + D_{w_{(t-1)}})$ starting from an initial vector having all the probability mass in the initial state. Let us denote the random variable describing the completion of the ith production lot as $X_{p,i}$, computed as follows:

$$Pr\{X_{p,i} \geq d_{p,i}\} = 1 - \left(\sum_{t=1}^{d_{p,i}} \pi(t-1) F^{(l_{p,i} <> D)} (P + D_{w_{(t-1)}}) F^{(l_{p,i} == D)} \right) \times \mathbf{1}$$

where $F^{<cond>}$ is a filtering matrix whose entries are equal to one on the diagonal only if the state satisfies the boolean condition $< cond >$. Filtering matrices exploit basic linear algebra to select only the transitions of interest from the matrix used to catch the moment in which a production lot completes its execution. For this reason, the filtering matrix on the *lhs* selects only the source state in which lot i is still under processing ($l_{p,i} <> D$), while the matrix on the *rhs* selects the destination states in which the lot i is completed ($l_{p,i} == D$). This guarantees that the process performs only those transitions that lead to the completion of the considered lot. Instead, the summation over t and the multiplication by $\mathbf{1}$ are used to evaluate all the time units until the due date, and to generate a scalar number from the distribution vector.

The computation of the time for the completion of a repair lot is slightly more complicated because the arrival is stochastic and, as a consequence, the due date is shifted on time. Thus, the calculations involve also the isolation of the moment in which the lot arrives into the system. By denoting the completion of the ith repair lot with $X_{m,i}$, we have that:

$$\begin{aligned} Pr\{X_{m,i} \geq d_{m,i}\} = 1 - \Big(& \textstyle\sum_{t=1}^{T} \pi(t-1) F^{(l_{m,i} <> NA)} (P + D_{w_{(t-1)}}) F^{(l_{m,i} == A)} \times \\ & \textstyle\prod_{t'=t+1}^{d_{m,i}-1} (P + D_{w_{(t-1)}}) \times F^{((l_{m,i} <> NA) \wedge (l_{m,i} <> A))} (P + D_{w_{(t-1)}}) F^{(l_{m,i} == A)} \Big) \times \mathbf{1} \end{aligned}$$

The first term isolates only those transitions starting from a state in which the repair lot is not arrived ($l_{m,i} <> NA$) and ending in a state where it is ($l_{m,i} == A$), while the second term carries on the process for $d_{m,i} - 1$ time units. The third term is used to catch only the moment in which the lot processing is completed ($(l_{m,i} <> NA) \wedge (l_{m,i} <> A)$) as already done for the production lots.

4 Numerical example

In this section we show the importance of the problem under investigation by means of a numerical example. We performed an experiment by assuming a system having $K = 3$ machines that has to produce 7 production lots and is waiting for 4 repair lots. The scheduling is such that the first three lots are scheduled at the first machine, the fourth and the fifth are scheduled at the second machine and the remaining lots are scheduled at the third machine. The sizes of production lots are equal to $|30, 20, 10, 30, 20, 30, 20|$ whereas repair lots are all composed of 30 parts each. Each part requires on average 1 time unit (TU) for being produced whereas set-ups require 1.25 TU on average. The probability that repair lots arrive in the next time unit is 0.5. In order to underline the strong impact that different policies have on the tardiness of each lot, we defined two matrices, D_1 and D_2, that represent the two extreme cases. We defined matrix D_1 in such a way that it always selects the next production lot. On the contrary, matrix D_2 gives always precedence to repair lots. We performed an experiment by four different schedulers: the first scheduler uses constantly matrix D_1; the second always uses matrix D_2; finally, the other two schedulers select randomly D_1 or D_2. We expect that the tardiness of production lots will be minimized by the first scheduler and maximized by the second. Vice versa, the second scheduler will minimize the tardiness of the repair lots and maximizes the one of the production lots. The third and fourth schedulers are expected to provide results in between the two extremes. Figure 1 shows the probability of the tardiness of the third production lot and the third repair lot as function of the due date. It is possible to notice that the results confirms the expectations. In fact, the probability to complete the third production lot on time is maximized by the first scheduler and minimized by the second. On the contrary, the second scheduler provides the best results for the repair lot whereas it is detrimental for the production lot. Furthermore, as expected, the trajectories referring to the random schedulers can be found between the trajectories of generated by the first and second scheduler.

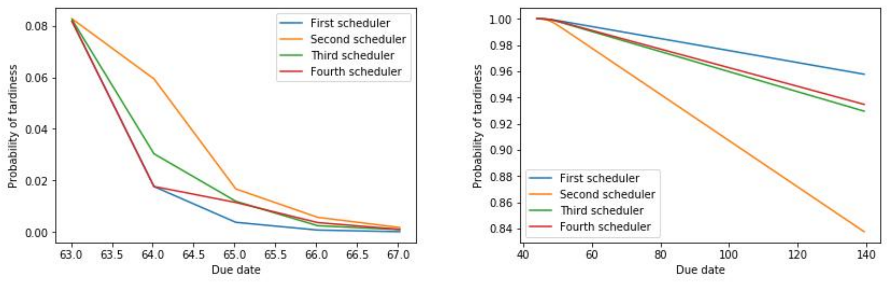


Fig. 1. Probability of the tardiness of the third production lot and the third repair lot as function of the due date.

5 Conclusive Remarks and Future Works

The paper presents a DTMDP that provides the tools for optimizing the scheduling of lots whose arrival into the system is uncertain and cannot be planned in advance. We tested two different scheduling strategies affecting the tardiness function in different ways and validating the model. Future works will regard the identification of optimal scheduling policies.

References

Horváth, A., 2011, "Approximating non-Markovian Behavior by Markovian Models", *PhD Thesis, Department of Telecommunications, Budapest University of Technology and Economics.*

Angius, A. and Colledani, M. and Yemane, A., 2018, "Impact of condition based maintenance policies on the service level of multi-stage manufacturing systems", *Control Engineering Practice*, Vol. 76, pp. 65-78.

A constraint programming approach for planning items transportation in a workshop context

Valentin Antuori[1,2], Emmanuel Hebrard[1], Marie-José Huguet[1], Siham Essodaigui[2], Alain Nguyen[2]

[1] LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France
`{vantuori,hebrard,huguet}@laas.fr`
[2] Renault, France
`{valentin.antuori, siham.essodaigui, alain.nguyen}@renault.fr`

Keywords: Constraint Programming, Single-machine Scheduling, TSP.

1 Introduction and related works

We are interested in transporting items in a workshop, from their production spots, to their consumption spots. For each type of items, there is one consumption machine, one production machine and four dedicated carriage trolleys, of limited capacity. Initially, there is one full trolley and one empty trolley in front of each machine. This problem comes from a real industrial problem, in the context of a car manufacturer workshop. When a trolley is full, an operator must transport it. He can transport several trolleys at the same time, making a train with them, but there is a limit on the maximal size on the train he can form. The production rate, and the consumption rate for a type of item is the same, therefore, when a carriage is full at a production spot, another carriage is empty at the associated consumption spot. The operator is also in charge of bringing back empty trolleys from their consumption machines, to production machines. The production rate, and the capacity of the trolley, permit to define a production cycle for each pair of machine. A production cycle is the time taken by a production (resp. consumption) machine, to fill (resp. empty) a trolley. The end time of a production cycle correspond to the beginning of the next cycle. For each pair of machines, and for each of their cycles considered until the time horizon, the goal is to deliver the trolleys which have been filled during the previous cycle to their consumption spots, and the trolleys which have been emptied during the previous cycle to their production spots.

The aim is to plan a route of the operator in order to satisfy pickup and delivery requests with time windows constraints. A particularity of this problem is the periodicity of the requests. Indeed, all the pickups and deliveries of a given item are almost completely sequenced. However, the relative order of pickups and deliveries of distinct items still need to be decided, potentially over a very long planning horizon.

This problem can be seen as a Pickup and Delivery Problem, with a single capacited vehicle and time windows. It is part of the one-to-one pickup and delivery family (which means that each pickup has a unique destination and conversely, each delivery has a unique source)(Cordeau et al. 2008). The single vehicle case comes from a Traveling Salesman Problem (TSP) in which precedences were added between clients. Since then, several additional constraints have been considered, such as time windows, limited capacity or LIFO loading (deliveries must respect a last-in-first-out rule). One can find complete survey on pickup and delivery problem in (Parragh et al. 2008) and (Berbeglia et al. 2007). There are temporal specificities in our problem. Indeed, each request has a twin request with the same time window: for each request from point A to point B to transport a full (resp. empty) trolley, there is a request from B to A for the empty (resp. full) trolley to do during the same time. Moreover, each request is repeated over time until the horizon, that is, each due date corresponds to a release date for the next request on the same pair of machine.

Routing problems can often be seen as scheduling problems with sequence-dependent setup times. In fact, time windows and precedences constraints, as well as the fact that there is no objective to optimize might suggest that the problem more adapted to scheduling technologies (Beck et al. 2003). From a scheduling point of view, it is a single resource (the driver of the train) scheduling problem. We have 4 types of activities: pickup and delivery of a full and an empty trolley which can't overlap. There is a precedence between a pickup and its delivery, and each activity has a time window. Travel times between two activities are then sequence-dependent. Finally the length of the train can be seen as a reservoir resource with limited capacity, which is filled by pickups, and emptied by deliveries.

2 Constraint Models

We propose two constraint models to deal with this problem. One TSP-oriented based on *next* variables, and one scheduling oriented. In the scheduling model, for each operation $i \in T = \{1, 2, ..., n\}$, we define the variable $start_i$ as the starting date of the operation i. Moreover we introduce for each pair of operations $i, j \in T$, a Boolean variable x_{ij} which represent the relative ordering of the two operations. The link between these two sets of variables is made with with the following constraint:

$$x_{ij} = \begin{cases} 1 \Leftrightarrow start_j \geq start_i + tt_{ij} + p_i \\ 0 \Leftrightarrow start_i \geq start_j + tt_{ji} + p_j \end{cases}$$

with tt_{ij}, the travel time between operation i and j, and p_i the processing time of i. In fact, we do not need a Boolean variable for every pair of tasks, and can easily reduce the model. For all pair of operation $i, j \in T$, such as i precede j, we avoid creating the variable x_{ij}, and directly post the following constraint : $start_j \geq start_i + tt_{ij} + p_i$.

Since production cycles are often much shorter than the planning horizon, the problem involves a lot of precedences. Therefore, we can drastically reduce the size of the model. Moreover, we observe that there are only two sensible ways to schedule the four activities of a production cycle (pickup of the full trolley, pickup of the empty trolley, and the corresponding deliveries). First, obviously pickups must precede their respective deliveries. Second, since the second pickup and the first delivery take place at the same spot, it is always preferable (w.r.t. the capacities and the time windows) to do the first delivery before the second pickup. Therefore, there are only two possible orders: the operator may either first pickup and deliver the full trolley, or first pickup and deliver the empty trolley. Hence, only one variable is needed for these four tasks. In order to deal with the constraint on the length of the train, we use the balance constraint introduced in (Laborie 2003).

The other model is inspired by the constraint model for TSP with time windows proposed in (Ducomman et al. 2016). For each operation i, there is a variable $next_i$ that indicates which operation directly follows i. Additional variables are needed in order to post redundant constraints. pos_i indicates the position of the operation i in the sequence of operations, and x_{ij} has the same semantic than in the first model. We add another variable $trainL_i$ which represent the length of the train before the operation i. Our model only differ by the adding of the following constraints :

$$trainL_{next_i} = trainL_i + l_i \qquad \forall i \in T \cup \{0\} \tag{1}$$

$$trainL_0 = 0 \tag{2}$$

$$pos_{del_i} > pos_i \qquad \forall i \in P \tag{3}$$

with l_i, the length of the trolley of the operation i (negative length for delivery), del_i is the associate delivery of the operation i, and P is the set of pickup operations. Constraint (1) and (2) represent the accumulation on the train length during the sequence of operations and use the ELEMENT constraint. The last constraint (3) ensures that each pickup precedes its delivery. In addition, we use the CIRCUIT constraint to enforce the Hamiltonian circuit.

3 Experiments

In order to compare the two models, and in addition to the industrial instances, we generated random instances [1]. We tried to generate only feasible instances, but we cannot guaranty that all instances are. There are 4 categories of instances (A, B, C, D). Instances A contain the least dense instances, i.e. with the least number of operations. They have also more pairs of machines with the same production cycle. Conversely, instances D are the most dense, and most asynchronous. We generate 10 instances of each category, and for each of them we consider 3 different temporal horizons, leading to 120 instances in total. The two models were implemented in the constraint solver Choco [2].

We observed that variable orderings following the chronological sequence tend to be more efficient. For instance, in the TSP model, building a tour by selecting the nodes from the first to be visited (pos_1) to the last to be visited (pos_n) was more effective than assigning the positions in a different order. In the case of the scheduling model, we use the following strategy: we say x_{ij} dominate x_{kl}, iff $min(start_i) \leq min(start_k)$ and $min(start_j) \leq min(start_l)$ with at least one strict inequality or $min(start_i) \leq min(start_l)$ and $min(start_j) \leq min(start_k)$ with at least one strict inequality, with $min(start_i)$ the minimum value of $start_i$. When choosing a variable to assign, we look for the boolean variables which is not dominated by any other variables, ties are broken randomly. We noticed a slight improvement when in addition we gave priority to the variables which order operations in the same pair of machines before the other ones. That is, we don't deal with a variable x_{ij} if the variable ordering the three remaining activities linked to operation i, and the variable ordering the three remaining activities linked to operation j are not instantiated. Generic heuristics such as *dom/wdeg* (Boussemart et al. 2004) were significantly less effective than these simple orderings.

Similarly, we explore first the branch that minimizes the start time of the next operation. In the TSP model, it means assigning pos_i to j such that the minimum value of $start_j$ is minimal. In the scheduling model, it means assigning x_{ij} to 1 iff minimum value of $start_i$ is lower than the minimum value of $start_j$. For both, and thanks to the propagation on the $start$'s variables, this heuristic acts more or less like a nearest neighborhood approach and takes advantage of the travel time between activities.

We ran each instance 10 times with randomized heuristics: if there are ties, they are broken randomly, and if not, one of the two best choices is chosen randomly with equal probabilities. We also used a restart strategy (the Luby sequence (Luby et al. 1993)) to improve the result. Each run has a timeout of 1 hour.

Table 1 shows the results. The first column denotes the category of the instance and second column denotes the temporal horizon. For the two models (denoted by Scheduling and TSP), 3 indicators are given, the number of solved instances (in average on 10 runs), the average time, and the numbers of fails for solved instances. We observe that the scheduling-based model can solve more instances in every category, and is faster in average. The average number of fails for the TSP model shows the relative slowness of that model. Most of the instances are unsolved.

[1] Available on https://github.com/AntuVal/SPDPTW

[2] Prud'homme, C., Fages, J.-G. & Lorca, X. (2017), Choco Documentation.

Table 1. Comparison of the two models on the generated instances

Cat.	Hor.	Scheduling			TSP			Nb Inst.
		NbSolve	Time	NbFail	NbSolve	Time	NbFail	
A	15000	9.3	44.11	174195	9.0	18.39	13081	10
	25000	8.5	52.74	143564	7.9	140.92	9445	10
	40000	8.0	36.48	29665	7.1	149.71	3803	10
B	15000	4.8	390.61	301231	2.1	943.88	18340	10
	25000	2.5	381.94	203849	0.7	1017.76	2015	10
	40000	2.0	402.88	419479	0.1	1929.71	283	10
C	15000	4.3	534.60	387635	2.0	541.68	14549	10
	25000	1.4	1264.03	352172	0.1	1693.80	1521	10
	40000	0.0	-	-	0.0	-	-	10
D	15000	2.2	495.55	112918	0.8	1271.75	9484	10
	25000	0.2	2565.44	45153	0.0	-	-	10
	40000	0.0	-	-	0.0	-	-	10

4 Conclusions

We introduced a one machine scheduling problem with several additional constraints. That problem is not new in the definition of its constraints, but the temporal structure of the instances makes it challenging. We proposed a randomly generated set of benchmark instances, whose a large number stay unsolved. We observed that a light model based on ordering pair of activities works better than a TSP-based model, which does not scale to industrial instances.

As a large number of the generated instances remains unsolved, there are still works to do. We are currently working on a Large Neighborhood Search (LNS) in order to improve these results. In this scheme, we relax the due date of each task, and instead we minimize the maximum lateness. Then during a LNS move, a subset of operations can be withdrawn from the sequence and re-inserted so as to minimize this objective. We also aim to deal with the entire problem, which is the team sizing for the whole shop. The goal is to plan a route through the pair of machines, minimizing the number of trains in the shop.

References

Beck, J. C., Prosser, P. & Selensky, E. (2003), Vehicle routing and job shop scheduling: What's the difference?, ICAPS, pp. 267–276.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I. & Laporte, G. (2007), 'Static pickup and delivery problems: a classification scheme and survey', *TOP* **15**(1), 1–31.

Boussemart, F., Hemery, F., Lecoutre, C. & Sais, L. (2004), Boosting systematic search by weighting constraints, ECAI, pp. 146–150.

Cordeau, J.-F., Laporte, G. & Røpke, S. (2008), Recent models and algorithms for one-to-one pickup and delivery problems, *in* 'The Vehicle Routing Problem', Springer, pp. 327–357.

Ducomman, S., Cambazard, H. & Penz, B. (2016), Alternative Filtering for the Weighted Circuit Constraint: Comparing Lower Bounds for the TSP and Solving TSPTW, AAAI, pp. 3390–3396.

Laborie, P. (2003), 'Algorithms for propagating resource constraints in ai planning and scheduling: Existing approaches and new results', *Artificial Intelligence* **143**(2), 151–188.

Luby, M., Sinclair, A. & Zuckerman, D. (1993), 'Optimal speedup of las vegas algorithms', *Information Processing Letters* **47**, 173–180.

Parragh, S. N., Doerner, K. F. & Hartl, R. F. (2008), 'A survey on pickup and delivery problems', *Journal für Betriebswirtschaft* **58**(2), 81–117.

Index merge in application to multi-skill project scheduling

Dmitry Arkhipov[1], Olga Battaïa[2]

[1] V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, Russian Federation
miptrafter@gmail.com
[2] KEDGE Business School, Bordeaux, France
olga.battaia@kedgebs.com

Keywords: project scheduling, multi-skill resources, workforce scheduling.

1 Introduction

In this research the index merge method is proposed to decrease the number of variables of Multi-Skilled Resource Constraint Project Scheduling Problem (MSRCPSP). This problem was formulated in (Neron E. and Baptista B. 2002) It generalizes Resource-Constraint Project Scheduling Problem (RCPSP) which is proved to be NP-hard in the strong sense (Garey M. and Johnson D. 1975). In (De Bruecker P. *et. al.* 2015) the survey of approaches to solve this problem is presented. Integer and mixed-integer statements of MSRCPSP are compared in (Almeida B.F. *et. al.* 2019).

The problem can be formulated as follows. There is a set of tasks N, set of workers W and a set of specialities S. If the worker w_i has skill s_l than $h_{il} = 1$, otherwise $h_{il} = 0$. For each task $j \in N$ the following parameters are given: p_j – processing time and a_{jk} – required number of workers with speciality s_k. Precedence relation e_{ij} can be defined for a pair of tasks, means that the task i has to be completed before the task j starts. The objective is to process all tasks without preemptions in the shortest time. For easier understanding of presented method non-human resources are not considered in this extended abstract.

The very important difference between MSRCPSP and RCPSP: *it is necessary to assign workers not only to tasks but also the speciality for which this worker is responsible for.* The following short example describes it. There is a task j which needs one worker with speciality s_1 and one with speciality s_2, a set of workers $W = \{w_1, w_2, w_3\}$ and a set of specialities $S = \{s_1, s_2, s_3\}$. Worker w_1 has specialities s_1 and s_2, w_2 – s_3, w_3 – s_1. Suppose that there is a binary variable x_{ij} which equals 1 if the worker i is assigned to task j, otherwise $x_{ij} = 0$. Suppose that speciality constraints are modelled as resources, i.e.

$$\forall l = 1,2: \quad \sum_{i=1}^{3} x_{1i} \cdot h_{il} \geq 1 \tag{1}$$

– there are enough workers for each speciality required for processing task j. The problem is that inequality (1) allows to assign workers w_1 and w_2 to the task j in feasible solution, which is not correct. In this case worker w_1 have to act as a specialist s_1 and s_2 simultaneously, which is not possible.

This means that the variable with three indices (task, worker, speciality) have to be introduced together with variables related to the start times of tasks. Such a large number of variables makes the problem very hard to solve.

2 Proposed approach

We propose a data-preprocessing method to decrease the number of variables by index merge. To solve MSRCPSP it is necessary to set task processing intervals and to assign workers to tasks. For each worker the speciality under which he operates the task has to be chosen. In the paper (Stadnicka D. *et. al.* 2017), the Hall's marriage theorem was used in integer linear programming (ILP) model to formalize operator assignments. In this work, we propose an approach based on the creation of task processing *scenarios*.

Let we call scenario z – the assignment list of workers $z = \{x_1^z, \ldots, x_{|W|}^z\}$, $x^z = 1$. If scenario z is *correct* for task $j \in N$, then it is possible to assign the set of workers $W_z = \{w_i | x_i^z = 1\}$ to all specialities required for processing j. The set of all correct scenarios for task j is denoted by Z_j. To decrease the number of scenarios in Z_j, only those with the number of workers not exceeding the required number for task j are considered.

2.1 The correctness of task scenario

The problem to verify the correctness of scenario z for task j can be formulated as follows.

Problem 1. *There is a set of workers defined by scenario z and a set of specialists required for job j. Each worker can have several specialities. How to verify that there is an assignment of workers to required specialities, such that each worker is assigned to the speciality he has and for each speciality the required number of workers are assigned?*

Let S_j – the ordered set of specialities, which includes a_{jk} elements of speciality k. Note that $|S_j| = |W_z|$. Then, problem 1 means that the set of workers W_z can be paired with the set of S_j. In terms of graph theory this problem is equivalent to the *perfect matching problem in bipartite graph.*

Problem 2. *There is a bipartite graph with two disjoint sets of vertices related to W_j and S_j. Vertex $i \in W_j$ is connected to vertex $s_j^k \in S_j$ related to speciality s if and only if $h_{is} = 1$. Is there a perfect matching for this graph?*

If there is a perfect matching, then we can assign workers to specialities they are matched with. If there is no perfect matching, then the workers cannot be assigned to specialities and task j cannot be processed under scenario z. Verification of scenarios for the example, presented in the previous section is illustrated on the Fig. 1.

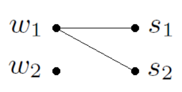
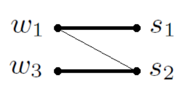
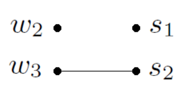
Scenarios for task j	{1, 1, 0}	{1, 0, 1}	{0, 1, 1}
Graph representation to find a perfect matching			
Is this scenario correct?	No	Yes	No

Fig. 1. Example: scenario verification.

Problem 2 can be solved by the Hopcroft–Karp algorithm (Hopcroft J. and Karp R. 1973) in $O(|W_z|^{5/2})$ operations. Therefore the complexity of the verification of the correctness of scenario z for task j is the same.

2.2 Creation of correct scenarios

To create the set of correct scenarios Z_j the following algorithm can be used.
Algotithm 1.

1. Calculate the number of workers $k_j = \sum_{s \in S} a_{js}$ required for processing task $j \in N$.
2. Generate all the scenarios of k_j workers.
3. Cycle all generated scenarios and check each scenario if it is correct for processing task j. If yes, add it to Z_j.

Number of scenarios to be verified – $C^{k_j}_{|W_z|} = \frac{|W_z|!}{k_j!(|W_z|-k_j)!}$ which is not more than $C^{\lceil |W_z|/2 \rceil}_{|W_z|}$. By the Stirling's formula this value can be asymptotically approximated by

$$C^{|W_z|}_{\lceil |W_z|/2 \rceil} \backsim \frac{2^{|W_z|+1/2}}{\sqrt{\pi |W_z|}}.$$

Then, subject to Hopcroft–Karp algorithm, the complexity of Algorithm 2 can be evaluated as $O(2^{|W_z|}|W_z|^2)$ operations. Creation of the correct scenarios for entire set of tasks N takes $O(n2^{|W|}|W|^2)$.

2.3 Using scenarios in MSRCPSP models

In case of Mixed-Integer Linear Programming models the variable with one index (task scenario) can be used instead of the variable with three indices (task, worker, speciality). In Constraint Programming models, interval variables associated with optional task scenarios can be used as follows.
Constraint programming MSRCPSP model.
Task processing optional interval variables: $\forall j \in N, z \in Z_j :$ int_z with size $|int_z| = p_j$.
Constraints:

- $\forall j \in N :$ $\sum_{z \in Z_j} presenceOf(int_z)$ – for each task only one scenario is presented in the solution;
- Let $f_i(t)$ – cumulative function defined for all $i \in W$ by the number of intervals associated with scenarios $z \in Z$ which involves the worker i ($x^z_i = 1$). Then the number of tasks processed simultaneously be the worker i can be modelled by $f_i(t) \leq 1$.
- $\forall e_{ij} \in E, z_1 \in Z_i, z_2 \in Z_j$: $endOf(int_{z_1}) \leq startOf(int_{z_2})$ – precedence relations have to be satisfied.

Objective – minimal makespan:

$$\min \max_{z \in Z} endOf(int_z).$$

3 Numerical experiments & analysis

In numerical experiments, we compared the presented model with CP model based on the IBM ILOG example: /examples/opl/sched_sequence. Both models were implemented using IBM CP Optimizer 12.6.2 and tested on Intel(R) Core(TM) i7-7700 HQ 2.8 GHz with 8 Gb RAM. We generated 800 random instances with 10, 20, 30 and 40 tasks and different number of workers, precedences and skills. Time limit for instances with 10 and 20 jobs was equal to 120 seconds and for instances with 30 and 40 jobs – 600 seconds. For 100% of generated instances proposed method gave better results. The results are presented in Table 1.

Table 1. Numerical experiments result.

	Without scenarios		With scenarios	
Tasks	Solutions found %	Optimum proved %	Solutions found %	Optimum proved %
10	53	26	100	89
20	44	19	100	63
30	18	6	100	39
40	6	0	100	0

The presented approach can be applied for other models to merge the variable indices and decrease the number of variables. Method allows to decrease the number of variables by considering constraints on the pre-processing stage and works especially efficiently if the pre-processing eliminates a large number of indices combinations as it is shown by numerical experiments.

The proposed idea has the following weaknesses.

- A larger number of constraints. In comparison with classic models, all constraints involving index i have to be applied to all merged combinations of indices including i.
- The need to store scenarios and a large number of constraints leads to the large amount of memory required.
- It is necessary to develop fast pre-processing procedures to eliminate the forbidden combinations of indices.

4 Conclusion

In this paper, an index merge method was presented and applied to MSRCPSP. The efficiency of the proposed model was evaluated theoretically and by a comparative analysis with default IBM ILOG CP model.

Acknowledgements

This research was supported by the Russian Foundation for Basic Research (grant 18-37-00295 mol_a).

References

Almeida B.F., I. Correia and F. Saldanha-da-Gama, 2019, “Modeling frameworks for the multiskill resource-constrained project scheduling problem: a theoretical and empirical comparison", *Intl Trans in Oper Res.*, Vol. 26, I. 3, pp. 946-967.

De Bruecker P., J. Van den Bergh, J. Beliën and E. Demeulemeester, 2015, “Workforce planning incorporating skills: State of the art", *Eur J Oper Res*, Vol. 243, I. 1, pp. 1-16.

Garey M., D. Johnson, 1975, “Complexity results for multiprocessor scheduling under resource constraints", *SIAM Journal on Computing*, Vol. 4, I. 4, pp. 397-411.

Hopcroft J., R. Karp, 1973, “An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs", *SIAM Journal on Computing*, Vol. 2, I. 4, pp. 225-231.

Neron E., B. Baptista, 2002, “Heuristics for multi-skill project scheduling problem", *In:International Symposium on Combinatorial Optimization.*

Stadnicka D., D. Arkhipov, O. Battaïa and R.M. Chandima Ratnayake, 2017, “Skill management in the optimization of aircraft maintenance processes", *IFAC PapersOnLine*, Vol. 50, I. 1, pp. 6912-6917.

Adapting the RCPSP framework to Evacuation Problems

Christian ARTIGUES[1], Emmanuel HEBRARD[2], Alain QUILLIOT[2], Peter STUCKEY[3], Hélène TOUSSAINT[2]

[1] LAAS Laboratory, CNRS Toulouse, France
e-mail: artigues@laas.fr

[2]LIMOS laboratory, CNRS/UCAlermont-Ferrand, France
e-mail: alain.quilliot@isima.fr

[3]Monash University, Melbourne, Australia

Keywords: Scheduling, RCPSP, Evacuation.

1. Introduction

In the context of the H2020 GEOSAFE European project [7], we have been working on the *late evacuation problem*, that means the evacuation of people and eventually critical goods facing a natural disaster (flooding, wildfire..).

We did it accordingly to the 2-step approach currently favored by practitioners [2, 4, 7]: the first step (pre-process) computes the routes that evacuees will follow; the second step, to be performed in real time, schedules the evacuation of estimated late evacuees along those routes. In practice, performing this last step requires forecasting the evolution of the disaster, rather difficult in the case of wildfires, because of their dependence to topography and meteorology [4]. But we consider here this issue as resolved and focus on the priority rules and evacuation rates which have to be imposed to evacuees [3]. Our model *non preemptive Tree evacuation planning* problem (NPETP) is equivalent to the model proposed in [1] evacuees have been clustered into groups with same original location and pre-computed route, and once a group starts moving, it keeps on at the same rate until reaching his target safe area (*non preemption*. This last hypothesis derives from practical concerns and aims at avoiding any panic effect during the evacuation process. The pre-computed evacuation routes are supposed to define a tree, with evacuee groups located at the leaves of the tree and the safe target place at its anti-root. While [1] addresses the problem through a discretization of both time and rate domains and constraint programming techniques, we make it here appear as a RCPSP: *Resource Constrained Project Scheduling Problem* variant, [5,6]), and use this RCPSP reformulation in order to get accurate optimistic bounds (lower bounds) and design an efficient network flow based heuristic.

The paper comes as follows: Section 2 provides the NPETP model. Section 3, 4 are devoted to optimistic bounds and algorithms. Section 5 proposes numerical tests.

2. The RCPSP Oriented NPETP Model

We consider here a tree *A*, oriented from its leaves towards its anti-root (*safe* target node), which is the extremity of a single arc *Root*. The leaf set is denoted by $J = \{1..N\}$: every $j \in J$ is provided with an *evacuee population* $P(j)$ which has to be brought until the safe target anti-root. We indistinctly talk about *j* as an *evacuation node* and an *evacuation job*. Arcs *e* are provided with both a capacity $CAP(e)$ and a length (or duration) $L(e)$. This induces that the path $\Gamma(j)$ which connect *j* to the anti-root has a length $\Lambda(j)$. Values $CAP(e)$ increase as long as we advance along path $\Gamma(j)$. For any arc *e*, $J(e)$ denotes the subset of *J* defined by all *j* such that $e \in \Gamma(j)$.

With every population *j* is associated a *deadline* $\Delta(j)$: evacuation of *j* must be achieved no later than time Δ(j). The evacuation time of population *j* is determined by its speed, which is supposed to be the same for all evacuees, and by its evacuation rate (number of people/time unit) v_j, which is imposed to be independent on the time. It comes that the duration of *evacuation job j* is equal to $\Lambda(j) + P(j)/v_j$. We want to schedule the evacuation process, while meeting the following requirements:

- Every evacuation job j is achieved at time T^{End}_j no later than deadline $\Delta(j)$;
- For any arc e, the sum of the *evacuation rates* v_j, taken for all j which are concurrently entering on e, does not exceed $CAP(e)$; (E1)
- The global *safety margin* $M = \text{Inf}_j (\Delta(j) - T^{End}_j)$ is the largest possible.

In order to cast our NPTEP problem into the RCPSP framework, we identify every *evacuation job j* with the *entering* process defined by the arrival of evacuees of j on the arc *Root*. Let us denote by T_j the starting time of this process and by T^*_j its ending time. Then we get a schedule if we decide, for every $j \in J$, starting time T_j, ending time T^*_j, and evacuation rate v_j, in such a way that:

- T_j is no smaller than the *release date* $R(j)$ = distance in the tree A from node j to the origin of *Root*;
- $T^*_j = T_j + P(j)/v_j \leq \Delta(j) - L(Root)$, which becomes the *deadline* $D(j)$ of job j. We deduce that v_j should be no smaller than $vmin(j) = P(j)/(D(j) - R(j))$.
- Above capacity constraints (E1) are never violated.

Because of the *Non Preemption* hypothesis, *entering* process j should take place in a continuous way between time T_j and time T^*_j, and define an interval. So we simplify the formulation of (E1) by introducing a vector $Z = (Z_{j,k},\ j,k = 1..N)\ 1..N)$ such that $Z_{j,k} = 1$ iff *j precedes k*. This allows to get our RCPSP oriented NPTEP model as follows:

NPTEP Model: {Compute vectors $T = (T_j , j = 1..N)$, $T^* = (T^*_j , j = 1..N)$, $v= (v_j , j = 1..N) \geq 0$, and {0,1}-valued vector $Z = (Z_{j,k}, j,k = 1..N)$ such that:

- $Z_{j,k} = 1$ iff *j precedes k*: we say that j and k *overlap* if $Z_{j,k} + Z_{k,j} = 0$;
- *Temporal constraints*:
  - For any j, $T_j + P(j)/v_j = T^*_j \leq D(j)$;
  - For any j, $T_j \geq R(j)$;
  - For any j, k, $Z_{j,k} = 1 \rightarrow T^*_j \leq T_k$;
- *Resource constraints*:
  - For any arc e, and any clique $C \subseteq \{1..N\}$ in the *overlap* sense,

    $\Sigma_{\ j \in J(e) \cap C}\ v_j \leq CAP(e)$. (E2)
- *Safety Margin Criterion*: Maximize $M = \text{Inf}_j (D(j) - T^*_j)$}

3. Optimistic Upper Bounds

We propose 2 upper bounds, both derived from the relaxation of *the Non Preemption constraint from the NPTEP model. We get upper bound UB-Tree* while keeping all constraints but the *Non Preemption* Constraint; we get upper bound *UB-Arc* while also relaxing all constraints (E2) but those related to the final arc *Root* and those related to the arcs $e(j)$ whose origins are the leaves $j = 1..N$ and whose capacities $CAP(e(j))$ are upper bounds values for the *evacuation rates* v_j. Computing both *UB-Arc* and *UB-Tree* follows the same algorithmic scheme:

Start from time value $t = 0$;
At any time t, consider all (*entering*) jobs j which have not been achieved yet and which are such that $t \geq R(j)$; Denote by $Q(j)$ the population which remains to enter into the arc *Root*;
Compute, for any such a job j, its current optimistic *safety margins* M_j, which means the *safety margin* $D(j) - T^*j = D(j) - Q(j)/CAP(e(j))$ which would be achieved if constant rate $v_j = CAP(e(j))$ were applied to job j from t on;
Make run jobs j with higher value M_j, which are assigned values v_j in such a way that values M_j evolve at the same pace for those jobs with highest priority;

Compute smallest time value t^* which fits some of the 3 following situations: (a) some job j gets to its end; (b) t coincides with the release date $R(j)$ of a job j which could not be started before; (c) the priority order related to safety margins M_j has been modified.
Update t: t <- t^*.

4. Algorithms

We propose here 2 algorithms. The first one is a fast insertion algorithm which relies on the Network Flow approach which was implemented in [6] in the case of RCPSP. The second one was already described in [1] and involved the use of IBM *CP Optimizer* Software.

4.1. A Network Flow Oriented Heuristic *NPETP*.

The key idea here is to consider the arcs e of the tree A as resources, likely to be exchanged by *evacuation jobs* i, j whose paths $\Gamma(i)$ and $\Gamma(j)$ share arc e. According to this purpose, we extend above NPETP model by introducing, for any pair (i,j) and any arc e in the set $Arc(i,j) = \Gamma(i) \cap \Gamma(j)$, the part $w_{i,j,e}$ of access rate to e which is given by i to j. We see that resulting vector w has to comply with the following flow constraints (E3):

- For any $j = 1..N$, e in $\Gamma(i)$: $\Sigma_{\text{i such that } e \in \text{Arc}(x,y)} w_{i,j,e} = v_i = \Sigma_{\text{i such that } e \in Arc(j,i)} w_{j,i,e}$; (E3)

We see that the main difficulty here is that we must choose between assigning high *rates* v_j to *jobs j* and let them monopolize the access to transit arcs of A, or conversely restricting v_j in order to make j share its arcs. In order to deal with it we design a 2 step *NPETP* approach:

```
NPETP Algorithmic scheme:
    First step (conservative approach):
        Starts from deadlines $D(j)$, $j = 1..N$; Not Stop;
        While Not Stop do
            Look for a feasible Schedule $(T, v, T^*)$;
            If Fail then Stop Else decrease deadlines $D(j)$, $j = 1..N$, in order to force values
            $T^*_j$ to decrease and so improve the Safety Margin criterion.
    Second step: Improve the solution by making evacuation rates $v_j$ increase (and so dates
    $T^*_j$ decrease), through resolution a specific linear program on vectors $w$ and $v$.
```

Then the core of *NPETP* Algorithm is related to the "Look for a feasible Schedule (T, v, T^*)" instruction of the "While" loop of the first step. We do it while relying on above flow vector w and providing every job with no more than what it needs in order to be achieved in time:

```
Start from some linear ordering $\sigma$ defined on $N$; Not Success; Not Failure;
While Not Success and Not Failure do
    Scan $\sigma$: $j_0$ being current job, values $v_j$, $T_j$ and $\Pi(j,e)$ = access level to arc $e$ that job $j$
    can transmit to $j_0$ have been computed for any $j$ prior to $j_0$ in s;
    Then:
        (1) : Scan path $\Gamma(j_0)$: for any $e$ in $\Gamma(j_0)$, compute flow values $w_{j,j0,e}$, $j$ prior to $j_0$
        in $\sigma$, in such a way $T^*_{j0} \le D(j_0)$; Derive $v_{j0}$ = Sup $_e$ $(\Sigma_j w_{j,j0,e})$ and related arc $e_0$;
        (2) : Increase the $w_{j,j0,e}$ for $e \neq e_0$ in order to make job $j_0$ run at the same rate for
        all arcs $e$ of $\Gamma(j_0)$.
        If Not Success then modify $\sigma$ accordingly and update Failure.
```

4.2. A Constraint Programming Approach for a Discrete Version of the NPTEP Model.

This approach associates with every variables v_j, T_j, $j = 1..N$, finite discrete domains, and apply the constraint propagation techniques which are at the core of the IBM *CP Optimizer* Software. All details are provided in [1]. Because of the rounding of values v_j, T_j, $j = 1..N$, it is also a heuristic approach.

5. Numerical Experiments

Purpose/Technical context: Algorithms were implemented C++, Windows 10, Visual Studio 2017, on PC with 16Go de RAM, Intel Core i5-8400 CPU @ 2.80GHz. Our goal was to evaluate both ability of the *NPETP* algorithm to yield good solutions and the accuracy of the optimistic upper bounds of Section III, while using results obtained in [1] through constraint programming (CPO Optimizer) as reference results.

Instances/outputs: They are as in [1]. The main characteristics of an instance is the number *N* of populations. We consider several instance packages, with, for any package, the number *S* which denotes the number of instances inside the package.

Outputs: For every instance package, we compute:

- *resCPO* = reference value through IBM-CPO in no more than 100 s (CPU).
- *optCPO* = number of instances such that IBM-CPO could achieve optimality of the discrete approximation of *NPETP*.
- *NPETP* = Value obtained through *NPETP* Algorithm; *cpuNPEP* = Related CPU time.
- *UB1* = Optimistic (upper) bound *UB-Arc*; *UB2* = Optimistic bound *UB-Tree*.

CPU times for the computation of both *UB1* and *UB2,* since they never exceed 0.1 s.

Then the following table provides a summary of our results:

N	*S*	*resCPO*	*optCPO*	*NPETP*	*cpuNPETP(s)*	*UB1*	*UB2*
10	15	104,00	12,00	97,96	0,56	112,16	107,16
15	16	69,81	12,00	65,14	0,79	78,53	73,94
20	11	40,09	8,00	42,36	1,30	51,28	43,58
25	5	8,40	0,00	43,80	1,17	70,30	49,20

Comments: The model handled by IBM-CPO is an approximation of NPETP model, and so *NPETP* algorithm obtains in some cases better results that IBM-CPO, even when IBM-CPO concludes to optimality. In any case, *NPETP*, whose computation times are very small, outperforms IBM-CPO as soon as the size of the problem increases. We also see that the *Tree Upper Bound UB2* provides us with a very efficient estimation of the optimal *NPETP* value, since the gap between *UB2* and Inf(*resCPO, NPETP*) is in average 5% (it tends to increase with the size of the instance).

Acknowledgements

This work is partially funded by the H2020-MSCA -2015 U.E project GEO-SAFE.

References

[1]. Artigues.C, Hebrard.E, Pencolé.Y, Schutt.A, Stuckey.P: "A study of evacuation planning for wildfires"; 17 th Int. Workshop on Constraint Modelling/Reformulation, Lille, France, (2018).
[2]. Bayram.V : "Optimization models for large scale network evacuation planning and management : a review "; Surveys in O.R and Management, (2016)
[3]. Even.C, Pillac.V, Van Hentenryk.P: "Convergent plans for large scale evacuation"; In Proc. 29 th AAAI Conf. On Artificial Intelligence, Austin, Texas, p 1121-1127, (2015).
[4].Intini.P, Gwynne.S.M, Ronchi.E, Pel.A : "Traffic modeling for wildland urban interface fire evacuation" ; Jour. Transportation Eng. A, 145, 3 (2019)
[5].Orji.M.J, Wei.S. "Project Scheduling Under Resource Constraints: A Recent Survey". Inter. Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 2, (2013)
[6]. Quilliot.A, Toussaint.H: "Flow Polyedra and RCPSP", RAIRO-RO, 46-64, p 379-409, (2012)
[7]. Veeraswamy.A, Galea.E, .Filippidis.L, Lawrence.P, Haasanen.S, Gazzard.R.J, TSmith.T.E: "The simulation of urban scale scenarios with application to the Swinly forest fire"; Safety Science 102, p 178-193, (2018).

Structural and Experimental Comparisons of Formulations for a Multi-Skill Project Scheduling Problem with Partial Preemption

Christian Artigues, Pierre Lopez, and Oliver Polo Mejía

LAAS-CNRS, Université de Toulouse, CNRS, France
`{christian.artigues, pierre.lopez, oliver.polo-mejia}@laas.fr`

Keywords: RCPSP, Multi-skill, Partial preemption, Mixed-integer programming, Constraint programming.

1 Problem statement

Preemptive scheduling problems assume that all resources are released during preemption periods, and that they can be used to perform other activities. However, in certain cases, constraints require that a subset of resources remains allocated to the activity when it has been interrupted, to ensure safety for example. Suppose one must execute an experimental activity that requires an inert atmosphere for its execution. In practice, one can stop this activity and allow the technicians and some of the equipment to be used in other activities. However, safety and operational constraints force us to preserve the inert atmosphere even when the activity is stopped (before its end). In other words, one cannot release the equipment that ensures the inert atmosphere during the preemption periods. Traditional preemptive schedule models cannot represent this behaviour since they assume that all resources are released during the preemption periods. Until now, the only way to model this activity, while respecting safety requirements, was to declare it as "non-preemptive". However, this decision can increase the project makespan, especially when the activities have restrictive time windows and the availability/capacity of the resources vary over time. We call *partial preemption* the possibility of only releasing a subset of resources during the preemption periods.

We are concerned here in multi-skill project scheduling problem (MSPSP) (Bellenguez and Néron 2012). We present in this extended abstract a new variant of the MSPSP that uses the concept of partial preemption. The variant of the problem under study is then called Multi-Skill Project Scheduling Problem with Partial Preemption (MSPSP-PP). To the best of our knowledge, it has not been studied yet in the scientific literature.

In the MSPSP-PP, if an activity is interrupted, we release only a subset of resources while seizing the remainder. We can then classify the set $\overline{I}$ of activities to be scheduled into three types according to the possibility of releasing the resources during the preemption periods: 1) Non-preemptive activities ($\overline{NP}$), if none of the resources can be released; 2) Partially preemptive activities ($\overline{PP}$), if a subset of resources can be released; and 3) Preemptive activities ($\overline{P}$), if all resources can be set free. In our case, the partial preemption is only related to mono-skilled resources, and we made the hypothesis that resources can always be released during preemption periods.

Our objective in the MSPSP-PP is to find a feasible schedule that minimises the total duration of the project ($C_{\max}$). Finding a solution consists in determining the periods during which each activity is executed and also which resources will execute the activity in every period; all this, while respecting the resources capacity and the activities characteristics. We must schedule these activities on renewable resources with limited capacity; they can be cumulative mono-skilled resources (machines or equipment) or disjunctive multi-skilled resources (technicians) mastering a given number of skills. Multi-skilled resources can respond to more than one skill requirement per activity and may execute it partially (except for non-preemptive activities where technicians must perform the whole activity). An activity is defined by its duration (D_i), its precedence relationships (set $\overline{E}$), its requirements of resources, its requirements of skills, the minimum number of technicians needed to perform it, and the subset of preemptive resources. Activities might or not have either a deadline or a release date. Figure 1 illustrates an example of an MSPSP-PP instance and a possible solution.

	Duration	Skill needed	Machine	Dead line	Release date	Activity type
Act 1	4	(S1,1)	(M1,1)	-	-	Preemptive
Act 2	2	(S3,1),(S4,1)	(M1,1)	5	3	Non-preemptive
Act 3	4	(S2,1)	(M1,1)	-	-	Partially preemp (M1 can not be released)

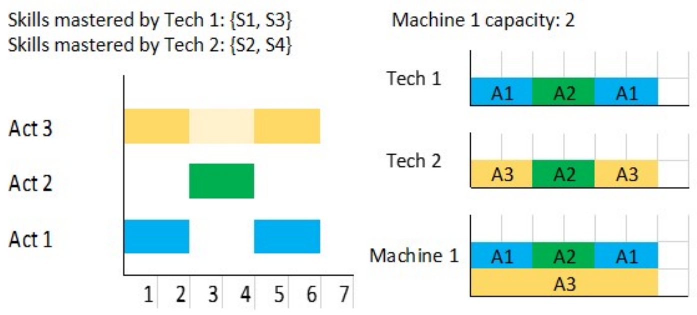


Fig. 1. Example of an MSPSP-PP instance.

The complexity of the MSPSP with partial preemption can be established using the classical RCPSP (Resource-Constrained Project Scheduling Problem) as a starting point. For each instance of the RCPSP, we can match an instance of the MSPSP with partial preemption, where all resources are mono-skilled, and none of the resources can be preempted. Thus, we can define the RCPSP as a particular case of the MSPSP with partial preemption. Since the RCPSP has been proved to be strongly NP-hard (Błazewicz *et al.* 1983), we can, therefore, infer that the MSPSP with partial preemption is also strongly NP-hard.

We propose five formulations for the MSPSP-PP using Mixed-Integer/Linear Programming (MILP) and Constraint Programming (CP).

2 MILP formulations

We present below five time-indexed formulations of the problem over a discretized horizon $\overline{H}$. These formulations generalize the ones presented in (Polo *et al.* 2018) and (Polo *et al.* 2019). All models are based on on/off binary variables $Y_{i,t}$ stating whether an activity i is in process in time period t, on/off binary variables $O_{j,i,t} = 1$ if technician j is assigned to activity i during period t, binary variable $S_{j,i} = 1$ if technician j is assigned to non-preemptive activity i (this variable is used to express that any technician assigned to a non-preemptive activity must remain assigned until its completeness). For any partially preemptive activity i, an on/off binary variable $Pp_{i,t} = 1$ if activity i is preempted in time period t. For the three first models, step binary variable $Z_{i,t} = 1$ if partially preemptive or non-preemptive activity i starts in time period t or before and step binary variable $W_{i,t} = 1$

if partially preemptive or non-preemptive activity i ends in time period t or after. We only provide a subset of the constraints of the first model (MSPP1a): precedence constraints (1) and the constraints (2–8) that link variables Y, Pp, Z, W, S, O and $C_{\max}$, the project makespan. The other constraints are standard resource constraints and operator availability constraints.

$$
\begin{aligned}
&D_i * (1 - Y_{l,t}) \geq \sum_{t'=t}^{|\overline{H}|} Y_{i,t'} && \forall (i,l) \in \overline{E}, \forall t \in \overline{H} && (1)\\
&Z_{i,t} \geq Y_{i,t'} && \forall i \notin \overline{P}, \forall t \in \overline{H}, \forall t' \leq t && (2)\\
&W_{i,t} \geq Y_{i,t'} && \forall i \notin \overline{P}, \forall t \in \overline{H}, \forall t' \geq t && (3)\\
&Pp_{i,t} = Z_{i,t} + W_{i,t} - Y_{i,t} - 1 && \forall i \in \overline{PP}, \forall t \in \overline{H} && (4)\\
&Z_{i,t} + W_{i,t} - Y_{i,t} = 1 && \forall i \in \overline{NP}, \forall t \in \overline{H} && (5)\\
&O_{j,i,t} \geq S_{j,i} + Y_{i,t} - 1 && \forall i \in \overline{NP}, \forall j \in \overline{J}, \forall t \in \overline{H} && (6)\\
&O_{j,i,t} \leq S_{j,i} && \forall i \in \overline{NP}, \forall j \in \overline{J}, \forall t \in \overline{H} && (7)\\
&C_{\max} \geq t * Y_{i,t} && \forall i \in \overline{I}, \forall t \in \overline{H} && (8)
\end{aligned}
$$

Given the variables $W_{i,t}$ and $Z_{i,t}$, we can replace the precedence constraints (1) by a disaggregated version below, yielding the second model (MSPP1b), while the third model (MSPP1c) includes both constraints (1) and (9).

$$
Z_{l,t} + W_{i,t} \leq 1 \qquad \forall (i,l) \in \overline{E}, \forall t \in \overline{H} \tag{9}
$$

We also propose two mixed continuous-time/discrete-time models (MSPP2a and MSPP2b), replacing binary variables $W_{i,t}$ and $Z_{i,t}$ by continuous time variables G_i and F_i representing the start and completion times of activity i, respectively. We replace constraints (1–5) by:

$$
\begin{aligned}
&F_i + 1 \leq G_l && \forall (i,l) \in \overline{E} && (10)\\
&Pp_{i,t} \leq 1 - Y_{i,t} && \forall i \in \overline{PP}, \forall t \in \overline{H} && (11)\\
&F_i - G_i + 1 \leq D_i + \sum_{t \in H} Pp_{i,t} && \forall i \in \overline{PP} && (12)\\
&F_i - G_i + 1 \leq D_i && \forall i \in \overline{NP} && (13)\\
&F_i \geq t * Y_{i,t} && \forall i \in \overline{I}, \forall t \in \overline{H} && (14)\\
&G_i \leq t * Y_{i,t} + (1 - Y_{i,t}) * |\overline{H}| && \forall i \in \overline{I} && (15)
\end{aligned}
$$

It remains to express the fact that partial preemption variables $Pp_{i,t}$ must be equal to 0 outside the execution interval of i. We either use the following constraints using variables Y (MSPP2a):

$$
Pp_{i,t} \leq \sum_{t'=1}^{t} Y_{i,t'} \ ; \qquad Pp_{i,t} \leq \sum_{t'=t}^{|\overline{H}|} Y_{i,t'} \qquad \forall i \in \overline{PP}, \forall t \in \overline{H} \tag{16}
$$

or the following ones using variables F and G (MSPP2b):

$$
F_i \geq t * Pp_{i,t} \ ; \qquad G_i \leq t * Pp_{i,t} - (1 - Pp_{i,t}) * |\overline{H}| \qquad \forall i \in \overline{PP}, \forall t \in \overline{H} \tag{17}
$$

In Section 3, we compare the proposed MILP formulations in terms of LP relaxation strength and we provide a computational comparison with the constraint programming formulation described in (Polo *et al.* 2019).

3 Structural and computational comparisons

3.1 Structural comparison of the MILP formulations

Using the transformation $G_i = |\overline{H}| - \sum_{t\in\overline{H}} Z_{i,t} + 1$ and $F_i = \sum_{t\in\overline{H}} W_{i,t}$ we show that the constraints in model MSPP2 involving the F_i and G_i variables are all implied by the constraints of model MSPP1 augmented with the transformation. As the computational experiments show that there are instances where MSPP1 has a strictly better LP relaxation than MSPP2, this yields the following result:

Theorem 1. *Formulation MSPP1b is tighter than formulations MSPP2a and MSPP2b.*

We could not prove a dominance relation between MSPP1a and MSPP1b, which was corroborated by the experiments fox which some bounds provided by the MSPP1a relaxation are better than those provided by the MSPP1b relaxation and vice-versa, which justifies the proposal of MSPP1c.

3.2 Computational experiments on the MILP and CP formulations

For computational tests, we use CPLEX 12.7 and CP Optimizer 12.7 for solving the MILP models and the CP models, respectively (using the default configuration and limiting the number of threads used by the solvers at 8). The computation time was limited to 10 minutes. We use the four sets of 30 activities instances of (Polo *et al.* 2019), each of them having 50 instances and a different proportion of preemption types. The activity durations are between 5 to 10 time units. There are up to 15 skills, 8 cumulative resources, 8 technicians (multi-skilled resources) divided into two teams, 20% of activities with time windows, the density of precedence relationships is low, and an average optimum $C_{\max}$ between 70 and 90 time units.

First of all, the MILP formulations are only efficient when the number of preemptive activities is high. For these instances, model MSPP1b provides the larger number of optimal solutions but model MSPP2b is faster and has a better average gap. There was no perceptible advantage for the integrated model MSPP1c. These computational results confirm one more time that a theoretically stronger formulation does not necessarily imply better practical performance. The MILP MSPP2b model outperforms the CP one when the percentage of preemptive activities is high, proving the optimality of a higher number of instances, and giving a lower average gap. CP, on the other hand, gives better results when this percentage is low. One could then say that the two methods are complementary.

Future research should be done in order to develop a hybrid method that better exploit the characteristics of each instance.

References

Bellenguez-Morineau O., E. Néron, 2008, "Multi-mode and multi-skill project scheduling problem", In *Resource-constrained project scheduling: models, algorithms, extensions and applications*, C. Artigues, S. Demassey, E. Néron (Eds.), Wiley-ISTE, pages 149-160.

Błazewicz J., J. K. Lenstra and A. H. G. Rinnooy Kan, 1983, "Scheduling subject to resource constraints: classification and complexity", *Discrete applied mathematics*, Vol. 5, pp. 11-24.

Polo-Mejía O., M.-C. Anselmet, C. Artigues and P. Lopez, 2018, "Mixed-integer and constraint programming formulations for a multi-skill project scheduling problem with partial preemption", In *12th International Conference on Modeling, Optimization and Simulation (MOSIM 2018)*, Toulouse, France, pages 367-374.

Polo-Mejía O., C. Artigues, P. Lopez and V. Basini, 2019, "Mixed-integer/linear and constraint programming approaches for activity scheduling in a nuclear research facility", *International Journal of Production Research*, In Press. DOI:10.1080/00207543.2019.1693654.

A Serial Schedule Generation Scheme for Project Scheduling in Disaster Management

Niels-Fabian Baur and Julia Rieck

University of Hildesheim, Germany
{baur,rieck}@bwl.uni-hildesheim.de

Keywords: disaster management, schedule generation scheme, flexible resource profiles.

1 Project Scheduling in Disaster Management

Due to climatic changes and a concomitant accumulation of extreme weather events, natural disasters, e.g., hurricanes and floods are a growing threat worldwide. According to the survey of Altay and Green, disasters can be described as large-scale events that pose an unusually high threat to life and health as well as to material assets. A particular challenge is the uncertainty of the events as well as the difficulty to predict a disasters impact. Four phases can be identified in the lifecycle of disaster management (i.e., mitigation, preparedness, response, and recovery). In particular, the response phase (post-disaster), where activities must be coordinated and information exchanged quickly, is considered in the literature, e.g., in the fields of infrastructure protection and medical care (Altay and Green 2006). Here, models and decision support systems can be used to directly reduce the impact of disasters. Therefore, the response phase is addressed in the following.

Using governmental emergency plans, necessary activities can be pre-defined that have to be carried out immediately after a disaster. For successful planning, it makes sense to visualize their precedence constraints by a project with a corresponding network. However, the execution of the activities requires suitable resources, the emergency forces. When responding to a disaster, it is helpful to have as many workforces as possible to carry out the necessary relief measures. Volunteers can constitute important resources and therefore be an effective complement to the professional forces in disaster relief. Hence, a successful response should integrate voluntary helpers. They must be assigned to activities and start times of activities must be determined. Consequently, a combined workforce and project scheduling problem arises. Due to the high complexity of the resulting problem, we have developed a serial schedule generation scheme (SGS) that finds feasible solutions even for large problem instances in reasonable time.

2 Problem Definition and Solution Approach

As described in Section 1, we consider a combined workforce and project scheduling problem (cf. Baur and Rieck 2019 for the mathematical model). It is assumed that projects with n real activities $i, j = \{0, \ldots, n+1\}$ and a set of reasonable precedence constraints E can be predefined. All real activities (e.g., fill sandbags and carry sandbags) require volunteers $k \in K$ to be carried out. Whether a volunteer can be assigned to an activity depends on two important aspects. Since voluntary helpers constitute partial renewable resources, each one has a defined time interval in which it is available. A resource k can be assigned to an activity at time t if $\theta_{kt} = 1$ applies. The other precondition for an assignment is that the resource is suitable for an activity. Every activity has a corresponding set of skills S_i that are needed to process it. Exemplary skills are physical fitness and driving licenses. A volunteer has to declare an associated level L_{ks} for all predefined skills $S \supseteq S_i$, which indicates to what extent the skill is mastered. According to Mansfield, the levels range from “not demonstrated” (i.e., $L_{ks} = 0$) to “outstanding” (i.e., $L_{ks} = 2$). A volunteer with a low skill level needs more time for the same workload (Mansfield 1996). Consequently, only if at least one volunteer with required skills is available, an activity can be processed. It is completed when the estimated total workload D_i is reached for every required skill.

Even if all predecessors $Pred(i)$ of activity i have been completed, a delay of the start S_i may be necessary, if there is not at least one suitable volunteer available. If an activity i already started but is not yet finished and no resource can be assigned at any time, the activity must be interrupted. Figure 1 shows exemplary interruptions of activity $i = 1$. It starts when the first resource is assigned at $t = 1$. At time $t = 2$ and $t = 4$ interruptions occur, as no suitable worker can be selected. After six time periods (i.e., $P_1 = 6$), the total processing time of $D_1 = 4$ is covered and the activity is completed.

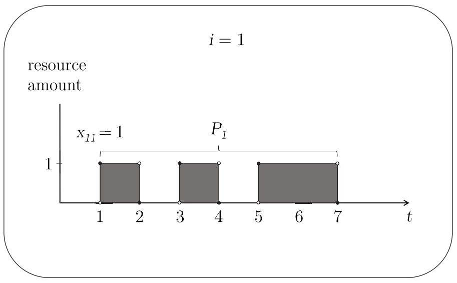


Fig. 1. Interruptions of an activity

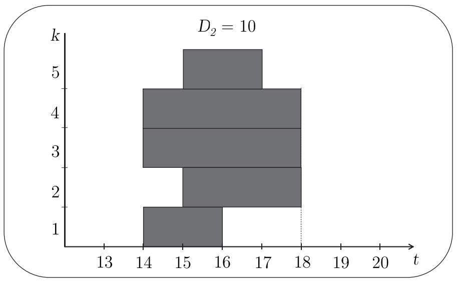


Fig. 2. Multiple assignments to an activity

If more than one resource is available at a given time, a team of several volunteers with different skill levels can be assigned to an activity. This would lead to a reduction of the expected activity duration P_i, which is variable and no deterministic parameter. The size of the assigned team is not constant during the processing time of an activity i what can be seen in Figure 2. It visualizes another example, where five resources are assigned simultaneously during the execution of activity $i = 2$. Although the estimated total processing time is $D_2 = 10$, the activity can be completed within four time units (i.e., $P_2 = 4$). Note that the number of resources assigned to $i = 2$ differs over time. While in period 14 only three volunteers are assigned, in period 15 there are five resources working in total. Therefore, the problem under consideration is a problem with flexible resource profiles (cf. (Naber and Kolisch 2014)). Besides the considered skills and the possible interruptions of an activity, the multiple resource assignments and variable activity duration are the most important characteristics of the problem. These properties make the problem more realistic, but also more difficult to solve. For this purpose, we implemented the SGS shown in Algorithm 1 to create feasible solutions even for large instances in decent time.

In the initialization step, the fictitious project start $i = 0$ is scheduled and added to the set of already completed activities $\mathcal{C}$. The schedule of all completed activities ST contains the corresponding start time $S_0 = 0$. Furthermore, the predecessors $Pred(i)$ of all nodes $i \in V$ are determined. The main step from line 3 is executed until all activities have been completed. At the beginning of his step, the eligible set $\mathcal{E}$ of activities is determined from which all predecessors have already been completed. The earliest start times ES_j of all activities $j \in \mathcal{E}$ are calculated in line 5. The activity with the highest priority is selected for the further procedure. The priority rule of the earliest start time (EST) was applied for first computational studies. For the selected activity j, the set σ of all skills for which the required working time D_j has not yet been reached is defined. T_j describes the set of time periods t at which j is actively processed by one or more resources and D_j^s represents the number of working hours remaining for each skill.

Algorithm 1 Serial Schedule Generation Scheme

```
 1: set $\mathcal{C} := \{0\}$, $ST^{\mathcal{C}} := (0)$;
 2: determine set of all predecessors $Pred(i)$ for activities $i \in V \setminus \{0\}$;
 3: **while** $\mathcal{C} \neq V$ **do**
 4:     determine $\mathcal{E} := \{i \in \bar{\mathcal{C}} | Pred(i) \subseteq \mathcal{C}\}$;
 5:     calculate earliest start $ES_j := \max_{i \in Pred(i)}(ST_i + P_i)$ of all $j \in \mathcal{E}$;
 6:     choose $j \in \mathcal{E}$ with highest priority;
 7:     determine $\sigma := \{s \in S_j\}$, $T_j := \emptyset$ and $D_j^s = D_j$ for all $s \in \sigma$;
 8:     **while** $\sigma \neq \emptyset$ **do**
 9:         **for** $t = ES_j$ to $\bar{d}$ **do**
10:             **for** $k \in K$ with $\theta_{kt} = 1 \wedge \sum_{s \in \sigma} L_{ks} > 0$ **do**
11:                 set $r_{jkt} := 1$, $\theta_{kt} := 0$ and $T_j := T_j \cup \{t\}$;
12:                 **for** $s \in \sigma$ with $L_{ks} > 0$ **do**
13:                     calculate $D_j^s := D_j^s - L_{ks}$;
14:                     **if** $D_j^s \leq 0$ **then**
15:                         set $\sigma := \sigma \setminus \{s\}$ and $P_j := t + 1 - ST_j$;
16:     set $\mathcal{C} := \mathcal{C} \cup \{j\}$ and $ST_j := \min_{t \in T_j} t$;
    **return** $ST^{\mathcal{C}}$.
```

The inner loop starting from line 8 is executed until σ is an empty set, thus the required total working hours for each skill have been met ($D_j^s \leq 0,\ \forall s \in \sigma$). From the earliest start until an activity can be completed or the maximum planning horizon is reached, all available resources k that have at least one of the required skills ($\sum_{s \in \sigma} L_{ks} > 0$) are considered one after the other. If no resource is available, the procedure continues with the next period. In line 11, the first resource found is assigned to the activity j at the current time t ($r_{ikt} = 1$ applies). Consequently, the volunteer is no longer available for other activities in this period (i.e., $\theta_{kt} := 0$). The lines 12-15 update the number of working hours still needed for each skill that is mastered by resource k and required for activity j. Once the total working time for a skill has been reached, the skill is removed from set σ and no longer needs to be considered. If σ is an empty set, activity j can be terminated and added to the set $\mathcal{C}$ in line 16. The start time of the activity j is defined as the time of the first resource assignment to j. When all activities are completed, the procedure terminates and returns the schedule of all start times.

3 Computational Results

For our computational study, we created 20 instances with $n = \{30, 60\}$ real activities on the basis of the PSPLIB benchmark (Kolisch and Sprecher 1996). The instances are supplemented by problem-specific parameters. For example, the number of considered skills is randomly set from 3 to 5. Under the assumption that the default skill level $L_{ks} = 1$ is the most common in reality, it gets the highest generation probability. The levels $L_{ks} = 0$ and $L_{ks} = 2$ are the least likely. The availability of resources is randomly determined within $\{8, 9, \ldots, 18\}$ time units without breaks. The SGS was implemented in C++ with Visual Studio 2019. The comparison results were generated with CPLEX 12.9 in GAMS 25.1 within a time limit of 7200 seconds. The tests were carried out on a server (two 2.1 GHz processors and 384 GB of RAM) using up to 16 threads.

Table 1 shows the obtained results. Instances with numbers 1 to 10 include 30 real activities. Instances 11 to 20 include 60 real activities. The SGS found feasible solutions for all instances within 100 s, which can be seen in the column "CPU", whereas CPLEX has only found a solution for five instances with 30 activities and no solution for the larger instances. The objective function values of the procedures can be taken from the columns

Table 1. Comparison of SGS and CPLEX solutions for instances with $n = 30$ and $n = 60$

	SGS		**CPLEX**		
no.	$F(x)$	CPU [s]	$F(x)$	CPU [s]	Gap [%]
1	46	42	–	7229	–
2	29	27	–	7212	–
3	33	36	–	7223	–
4	21	24	21	2890	0.0
5	38	60	38	7230	0.0
6	22	4	20	2228	10.0
7	42	7	26	7215	61.5
8	29	7	–	7228	–
9	27	26	26	3519	3.8
10	32	5	–	7247	–

	SGS		**CPLEX**		
no.	$F(x)$	CPU [s]	$F(x)$	CPU [s]	Gap [%]
11	50	30	–	7883	–
12	43	72	–	7807	–
13	51	25	–	8065	–
14	52	70	–	7876	–
15	78	22	–	7954	–
16	37	15	–	8168	–
17	81	27	–	8121	–
18	48	16	–	8097	–
19	100	76	–	8009	–
20	59	24	–	8089	–

$F(x)$. The objective is to minimize the project duration and thus to cope with the disaster as soon as possible. The column "Gap" shows the deterioration of the solution found by the SGS compared to the solution of CPLEX. For instances 4 and 5, the SGS found an equally good solution after 24 respectively 60 s, as CPLEX did after 7200 s. Only for instance 7, the SGS found a clearly worse (61.5%) solution than CPLEX.

4 Conclusion and Outlook

The abstract introduces a serial schedule generation scheme for a particular problem with skills, skill levels, possible interruptions of activities, and variable activity durations. The results of the procedure were compared to the results of CPLEX. The next step is the development of a metaheuristic, which is able to improve the found solution in reasonable time. In addition, the problem should be adapted to the dynamic and stochastic characteristics of a disaster by transforming the currently static and deterministic model into a dynamic formulation with stochastic components.

References

Altay, N., Green, W.,G., 2006, "OR/MS research in disaster operations management", *European Journal of Operational Research*, Vol. 175, pp. 475-493.

Baur, N.-F., Rieck, J., 2019, "Project Management with Scarce Resources in Disaster Response", submitted to *Operations Research Proceedings 2019*.

Kolisch, R., Sprecher, A., 1996, "PSPLIB – A project scheduling problem library", *European Journal of Operational Research*, Vol. 96, pp. 205-216.

Mansfield, R.S., 1996, "Building competency models: Approaches for HR professionals", *Human Resource Management*, Vol. 35, pp. 7-18.

Naber, A., Kolisch, R., 2014, "MIP models for resource-constrained project scheduling with flexible resource profiles", *European Journal of Operational Research*, Vol. 239(2), pp. 335-348.

Scheduling to minimize maximum lateness in tree data gathering networks

Joanna Berlińska

Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań, Poland
Joanna.Berlinska@amu.edu.pl

Keywords: scheduling, data gathering network, maximum lateness, hybrid flow shop.

1 Introduction

Scheduling for data gathering has been attracting increasing attention in recent years. Choi and Robertazzi (2008) and Moges and Robertazzi (2006) constructed algorithms for partitioning the total amount of measurements between the nodes of a wireless sensor network in order to gather the data in the shortest possible time. Algorithms minimizing schedule length were also proposed for networks with data compression (Berlińska 2015, Luo et al. 2018), and with limited memory (Berlińska 2020). Scheduling with maximum lateness criterion in star networks was studied by Berlińska (2018), Berlińska (2019).

In this paper, we analyze minimizing maximum lateness in 2-level tree networks. The data gathering application consists of three partially overlapping stages. First, each of the leaf nodes of the network has to transfer acquired data to an appropriate intermediate node. In the second stage, datasets are preprocessed by the intermediate nodes. Finally, they are transferred from the intermediate nodes to a single base station. It is assumed that an intermediate node can receive at most one dataset at a time, and it can process at most one dataset at a time. Therefore, a subnetwork consisting of an intermediate node and all leaves that communicate with it, can be seen as a sequence of two machines working in a flow shop mode. The base station can also receive at most one dataset at a time, and hence, the third stage consists in executing a sequence of jobs on a single machine. Thus, our network is a three-machine hybrid flow shop with m dedicated machines in the first stage and the second stage, and one machine in the last stage. Hybrid flow shops with dedicated machines were studied mostly in two-machine setting (see the survey Hwang and Lin (2018)). Three-machine hybrid flow shop with one machine in the first and the third stage, and two dedicated machines in the second stage, was studied by Riane et al. (1998). The case of two dedicated machines in the last stage, and one machine in the first and the second stage was analyzed by Bedhief and Dridi (2019). To our best knowledge, three-machine hybrid flow shops with multiple dedicated machines in more than one stage were not studied in the earlier literature.

2 Problem formulation and complexity

The data gathering network consists of n leaf nodes, m intermediate nodes and a single base station. Intermediate node P_j $(1 \le j \le m)$ collects data from n_j leaf nodes P_{jk}, where $k = 1, \dots, n_j$. Thus, $n_1 + \dots + n_m = n$. Leaf node P_{jk} acquires dataset D_{jk} of size α_{jk} at time r_{jk}. The due date for receiving this dataset at the base station is denoted by d_{jk}. The time necessary to transfer one unit of data is denoted by C. Thus, dataset D_{jk} is sent from P_{jk} to P_j in time $C\alpha_{jk}$. After receiving the whole dataset, node P_j has to preprocess it, which takes time $A\alpha_{jk}$. During this process, the dataset size changes to $\gamma\alpha_{jk}$, where γ is a given application parameter. Afterwards, the dataset has to be sent to the base station, which takes time $C\gamma\alpha_{jk}$. Each node can receive at most one dataset at a time. An intermediate node can simultaneously receive one dataset, preprocess another dataset, and send yet another dataset to the base station. Preemptions are allowed both in communication and computation.

Let T_{jk} be the time when dataset D_{jk} arrives at the base station. The lateness of D_{jk} is $L_{jk} = T_{jk} - d_{jk}$. Our goal is to organize dataset transfer and processing so that the maximum lateness $L_{max} = \max_{j=1}^{m} \max_{k=1}^{n_j} \{L_{jk}\}$ is minimized.

When $m = 1$ and $\gamma = 0$, our scheduling problem becomes equivalent to minimizing the maximum dataset lateness in a star network with datasets processed at the base station, which was proved to be strongly NP-hard by Berlińska (2019). Thus, the problem analyzed in this work is also strongly NP-hard.

3 Algorithms

In this section, we propose heuristic algorithms for solving our scheduling problem. Note that if a schedule for the first two stages of the application is fixed, an optimum schedule for the last stage can be easily found. Indeed, for each dataset D_{jk}, the time r'_{jk} when it becomes available for transfer to the base station is known, and it remains to solve an instance of problem $1|r_j, pmtn|L_{max}$, which can be done using the preemptive earliest due date first rule (Horn 1974). Therefore, our algorithms concentrate on building a good schedule for transferring datasets to intermediate nodes and preprocessing them, separately for each of the m subnetworks.

Firstly, we implement an exponential algorithm BB, which uses the branch-and-bound technique to obtain a schedule which minimizes the maximum dataset lateness after two stages of the application. A detailed description of this algorithm can be found in Berlińska (2019). Note that a partial schedule that minimizes the maximum lateness after the first two stages, does not necessarily result in the optimum schedule for the whole application. Thus, although algorithm BB uses an enumerative approach, it does not guarantee finding optimum solutions.

Secondly, we propose algorithms that build a communication schedule first, and after fixing it, construct a schedule for dataset preprocessing. For each of these two stages, we use one of the following rules:

- FIFO: choose datasets in the order in which they are released (no preemptions);
- EDD: select an available dataset with the smallest due date (possible preemptions);
- SRT: choose an available dataset with the shortest remaining transfer/preprocessing time (possible preemptions).

An algorithm which uses Rule1 for dataset transfer, and Rule2 for dataset preprocessing, will be called Rule1-Rule2. We study all possible combinations of the above rules, resulting in 9 different algorithms. Each of these algorithms finds a schedule for the j-th subnetwork in $O(n_j \log n_j)$ time, for $j = 1, \ldots, m$. After fixing subnetwork schedules, a schedule for sending datasets to the base station is computed in $O(n \log n)$ time. Since $n = n_1 + \ldots + n_m$, the total algorithm running time is also $O(n \log n)$.

4 Experimental results

In this section, we compare the performance of the proposed heuristics. The algorithms were implemented in C++ and run an Intel Core i5-3570K CPU @ 3.40 GHz with 8GB RAM. Due to limited space, we report here only on a subset of the obtained results. In the experiments presented here, the network consisted of $m = 5$ subnetworks, containing $n_i = 10$ leaf nodes each. Note that if $m > 1$, A is small in comparison to C, and γ is large, then the third stage of the application dominates the whole schedule, i.e. it takes a significantly longer time than each of the first two stages, and hence, it has the largest impact on the obtained maximum lateness. Since the last stage is always scheduled optimally, building good solutions is easy in this case. Therefore, in order to construct demanding instances, we used $C = 1$, $A \in \{1, 2, 3\}$ and $\gamma \in \{0.01, 0.1, 0.5\}$. Dataset release times r_{jk} were generated separately for each subnetwork j as follows. The release time of the first dataset was $r_{j1} = 0$. The remaining release times were computed from the formula $r_{jk} = r_{j,k-1} + \delta_{jk}$, with δ_{jk} chosen randomly from interval $[1, 10]$, for each j and k independently. Due dates d_{jk} were selected randomly from interval $[0, 100]$, and dataset sizes α_{jk} from interval $[1, 15]$. For each analyzed combination of parameters A and γ, 30 instances were generated and solved.

Since the optimum solutions for the test instances were not known, in order to assess the schedule quality we computed a lower bound

$$LB = \max_{j=1,\dots,m} \max_{k=1,\dots,n_j} \{r_{jk} + (C(1+\gamma) + A)\alpha_{jk} - d_{jk}\}. \tag{1}$$

Measuring schedule quality for the L_{max} criterion may be problematic, as the optimum value can be positive, zero or negative, which precludes using relative measures. Therefore, we measure solution quality by the difference between the maximum lateness delivered by a given algorithm and the lower bound LB.

Table 1. Average distance of the solutions from the lower bound.

	$\gamma = 0.01$			$\gamma = 0.1$			$\gamma = 0.5$		
Algorithm	$A = 1$	$A = 2$	$A = 3$	$A = 1$	$A = 2$	$A = 3$	$A = 1$	$A = 2$	$A = 3$
BB	2.201	42.858	119.080	1.313	39.883	124.100	49.900	59.550	124.400
FIFO-FIFO	32.784	101.835	171.999	34.127	99.437	180.550	50.450	97.783	179.500
FIFO-EDD	28.299	44.493	120.244	29.027	43.243	125.040	50.350	59.550	125.017
FIFO-SRT	31.901	98.147	173.014	31.463	95.050	172.930	51.150	95.533	177.200
EDD-FIFO	5.731	52.349	124.176	6.183	48.993	129.527	51.000	54.050	128.483
EDD-EDD	3.602	49.125	123.044	3.197	46.710	129.527	51.217	57.633	127.617
EDD-SRT	18.915	95.982	172.949	19.417	91.010	173.167	52.917	95.183	176.383
SRT-FIFO	30.856	97.945	172.479	28.157	94.933	177.153	50.883	97.250	178.517
SRT-EDD	27.891	48.163	120.943	25.143	45.720	127.367	50.633	58.750	126.633
SRT-SRT	31.324	100.948	173.217	30.500	95.207	175.857	51.467	94.933	175.633

The average quality of the obtained solutions is presented in Table 1. The distances from LB obtained by all algorithms grow with A. The main reason for this is that the distance between LB and the actual optimum increases with A. Algorithm BB delivers the best results for all settings except $A = 2$, $\gamma = 0.05$. However, BB has high computational costs. Its average running time ranged from about 7 seconds for $A = 1$, $\gamma = 0.5$ to approximately 2075 seconds for $A = 3$, $\gamma = 0.5$, while the remaining heuristics needed about 0.004 seconds in all analyzed settings. All algorithms return similar results when $A = 1$ and $\gamma = 0.5$. This illustrates the mentioned above fact that the combination of small A and big γ leads to easy instances.

Let us now compare the performance of the fast heuristics in the remaining settings. When $A = 1$ and $\gamma \in \{0.01, 0.1\}$, algorithm EDD-EDD is the winner. When $A = 3$, the best results are obtained by FIFO-EDD, for all values of γ. For tests with $A = 2$, the best results are returned by algorithm FIFO-EDD when $\gamma \in \{0.01, 0.1\}$, and by EDD-FIFO when $\gamma = 0.5$. It seems that the choice between the EDD and FIFO rules should be based on the expected durations of the three stages of our application. When $A = 2$ and $\gamma \in \{0.01, 0.1\}$, or when $A = 3$, the second stage dominates, and the best strategy is to use FIFO in the first stage. For $A = 2$ and $\gamma = 0.5$, the third stage is the longest (because $m = 5$), and FIFO should be applied in the second stage. In the remaining cases, the best strategy is to use only EDD rule. We infer that if stage i dominates the schedule ($i = 2, 3$), then the FIFO rule should be applied in stage $i - 1$ in order to pass some data to stage i as soon as possible, and EDD should be used in the remaining stages. If there is no dominating stage, algorithm EDD-EDD seems the best choice.

5 Conclusions

In this work, we analyzed minimizing maximum lateness in tree data gathering networks. As the problem is computationally hard, we proposed several heuristic algorithms. Computational experiments showed that algorithm BB usually delivers the best results, but at a high computational cost. Good schedules can be obtained in polynomial time using an adequate combination of EDD and FIFO rules. Future research may include investigating theoretical performance guarantees of the proposed algorithms.

Acknowledgements

This research was partially supported by the National Science Centre, Poland, grant 2016/23/D/ST6/00410.

References

Bedhief, A., N. Dridi, 2019, “Minimizing makespan in a three-stage hybrid flow shop with dedicated machines”, *International Journal of Industrial Engineering Computations*, Vol. 10, pp. 161-176.

Berlińska J., 2015, “Scheduling for data gathering networks with data compression”, *European Journal of Operational Research*, Vol. 246, pp. 744-749.

Berlińska J., 2018, “Scheduling Data Gathering with Maximum Lateness Objective”, In: R. Wyrzykowski et al., *Parallel Processing and Applied Mathematics: 12th International Conference PPAM 2017, Part II*, LNCS 10778, pp. 135-144, Springer, Cham.

Berlińska J., 2019, “Scheduling in a data gathering network to minimize maximum lateness”, In: B. Fortz, M. Labbé, *Operations Research Proceedings 2018*, pp. 453-458, Springer, Cham.

Berlińska J., 2020, “Heuristics for scheduling data gathering with limited base station memory”, *Annals of Operations Research*, Vol. 285, pp. 149-159.

Choi K., T.G. Robertazzi, 2008, “Divisible Load Scheduling in Wireless Sensor Networks with Information Utility”, In: *IEEE International Performance Computing and Communications Conference 2008: IPCCC 2008*, pp. 9-17.

Horn, W.A., 1974, “Some simple scheduling algorithms”, *Naval Research Logistics Quarterly*, Vol. 21, pp. 177-185.

Hwang, F.J., B.M.T. Lin, 2018, “Survey and extensions of manufacturing models in two-stage flexible flow shops with de dicated machines”, *Computers and Operations Research*, Vol. 98, pp. 103-112.

Luo, W., Y. Xu, B. Gu, W. Tong, R. Goebel, G. Lin, 2018, “Algorithms for Communication Scheduling in Data Gathering Network with Data Compression”, *Algorithmica*, Vol. 80, pp. 3158-3176.

Moges M., T.G. Robertazzi, 2006, ”Wireless Sensor Networks: Scheduling for Measurement and Data Reporting”, *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 42, pp. 327-340.

Riane, F., A. Artiba, S.E. Elmaghraby, 1998, “A hybrid three-stage flowshop problem: Efficient heuristics to minimize makespan”, *European Journal of Operational Research*, Vol. 109, pp. 321-329.

On the Activity Criticality in Project Scheduling with Generalized Precedence Relationships

Lucio Bianco, Massimiliano Caramia and Stefano Giordani

University of Rome "Tor Vergata", Italy
bianco, caramia, giordani@dii.uniroma2.it

Keywords: Critical Path, Criticality and Flexibility Analysis, Project Scheduling.

1 Introduction

It is well known that a project network with Generalized Precedence Relations (GPRs), due to the presence of maximum and minimum times lags, may contain cycles and a critical path may contain cycles of zero length (De Reyck, 1998). Consequently, it may happen that the project duration increases when the duration of a critical activity is shortened. This phenomenon was firstly studied by Wiest (1981). In another seminal paper, Elmaghraby and Kamburowski (1992) further studied this anomaly under GPRs. They introduced five different criticality types (i.e., start-critical, finish-critical, backward-critical, forward-critical, and bi-critical) and the new concept of flexibility. Later, De Reyck (1998) in his doctoral thesis, revisited these concepts adapting them to an Activity on Nodes (AON) representation of the project network and proposed a method for recognizing criticalities and inflexibilities of an activity, based on the types of its ingoing and outgoing precedence relations. Since the Ph.D. work of De Reyck appears not to be present in the open literature, the reader may find the analysis of De Reyck in the book by Demeulemeester and Herroelen (2002). To the best of our knowledge, the analysis made by De Reyck has been widely accepted and not revised by any point of view in the last twenty years.

In this work, starting from some concerns related to the criticality definitions of the activities and potential failures of De Reyck's method for the analysis of activity criticalities and flexibilities, we propose a new method. Section 2 provides some definitions and notations. In Section 3, by means of one example, we show some potential failures of the De Reyck's method, giving in Section 4 new results. Similar remarks can also be made for the method proposed by Elmaghraby and Kamburowski (1992). In Section 5 we provide a brief outline of the new method after having redefined and discussed the types of criticalities.

2 Definitions and notations

Accordingly to De Reyck (1998) and to Demeulemeester and Herroelen (2002), hereafter we assume that a project is modeled by means of an AON network $N = (V, A; d, \delta)$. V is the set of nodes, with $V = V^r \cup \{1, n\}$, where $V^r = \{2, \ldots, n-1\}$ is the set of $n-2$ real activities, d_i is the duration of activity $i \in V^r$, and nodes 1 and n are two additional dummy activities, with duration equal to zero, representing project beginning and completion, respectively; without loss of generality, we assume the real activity durations being integers and positive. A is the set of arcs representing GPRs between pairs of activities. An arc may model a start-to-start (*SS*), a start-to-finish (*SF*), a finish-to-start (*FS*) and a finish-to-finish (*FF*) precedence relation with minimum or maximum time lags for an overall number of eight relations that may be represented, i.e., $SS_{ij}^{min}(\delta)$, $SS_{ij}^{max}(\delta)$, $SF_{ij}^{min}(\delta)$, $SF_{ij}^{max}(\delta)$, $FS_{ij}^{min}(\delta)$, $FS_{ij}^{max}(\delta)$, $FF_{ij}^{min}(\delta)$, $FF_{ij}^{max}(\delta)$, where δ is the minimum or maximum time lag. It is well known (see e.g., Demeulemeester and Herroelen, 2002) that a GPR with maximum time lag is equivalent to a GPR with minimum time lag with opposite direction and opposite time lag, i.e., $SS_{ij}^{max}(\delta) \equiv SS_{ji}^{min}(-\delta)$, $SF_{ij}^{max}(\delta) \equiv FS_{ji}^{min}(-\delta)$, $FS_{ij}^{max}(\delta) \equiv SF_{ji}^{min}(-\delta)$, and $FF_{ij}^{max}(\delta) \equiv FF_{ji}^{min}(-\delta)$. Hence, we can always model the project activities and their relationships with a GPRs AON network with only minimum time lag, but the resulting network may contain cycles, due to maximum time lag. Therefore, without loss of generality, in the following we assume that the project has only GPRs with minimum time lags.

It is well known that with the transformations of Bartush *et al.* (1988) we can represent the project network in a so called *standardized* form where there are, for example, only GPRs of type $SS_{ij}^{min}(\ell)$. This standardized network allows to calculate the project duration as the length of the longest path form node 1 to node n, i.e., the length of a critical path. Moreover, it allows to determine the critical activities and the critical arcs (i.e., critical precedences among critical activities).

3 An example showing potential failures

We show a project network example for which the method proposed by De Reyck fails in determining some activities' criticalities. Analog failures can also be shown on the method by Elmaghraby and Kamburowski. Let us consider the project network with GPRs in Figure 1a, with node weights being activities' durations. The standardized network (with only $SS_{ij}^{min}(\ell)$ precedences) is shown in Figure 1b, with arcs' weights being time lags. Let us consider the critical path $(1, 2, 3, 4, 5, 4, 6, 7)$ with length equal to 20 (note that the path contains a cycle of length equal to zero). All the activities belong to the critical path. The criticalities according to De Reyck's method (see the definitions and Table 6 at page 124 of the book of Demeulemeester and Herroelen, 2002) are: activity 2, forward-critical; activity 3, finish-critical; activity 4, bi-critical; activity 5, start-critical; activity 6, finish-critical.

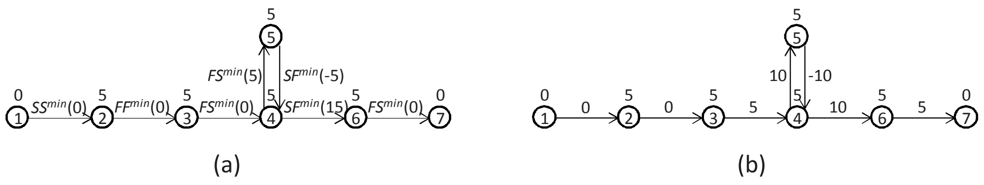


Fig. 1. The project network with GPRs of the example (a) and its standardized network (b)

Actually, the criticalities of activities 3, 4, and 5 are different, as the flexibility analysis reveals. Indeed, if it was $d_3 = 6$, it would be $\ell_{23} = -1$ and $\ell_{34} = 6$, with ℓ_{ij} being the length of arc (i, j) in the standardized network. In this case, the length of the longest path from node 1 to node 3 would be negative and hence we need to add arc $(1, 3)$ of length $\ell_{13} = 0$ to the standardized network, which corresponds to add precedence $SS_{13}^{min}(0)$ between activities 1 and 3 (and the related arc to the original network), to force activity 3 to start not before time 0. On the standardized network, the critical path would change to $(1, 3, 4, 5, 4, 6, 7)$ of length 21. If it was $d_3 = 4$, it would be $\ell_{23} = 1$, $\ell_{34} = 4$, and the critical path would not change. Therefore, activity 3 is backward-flexible and forward-inflexible, and, hence, it is also forward-critical. Similarly, if it was $d_5 = 6$, the length of the longest path from node 5 to node 7 would be less than d_5, and hence we need to add arc $(5, 7)$ of length $\ell_{57} = d_5$ to force the start time of dummy activity 7 (i.e., the project makespan) to be not less than the finish time of activity 5, which corresponds to add precedence $FS_{57}^{min}(0)$ between activities 5 and 7 (and the related arc to the original network). On the standardized network, the critical path would be $(1, 2, 3, 4, 5, 7)$ of length 21. Therefore, activity 5 is forward-inflexible, meaning that it is also forward-critical. Finally, activity 4 is both start- and forward-critical, because it is backward-flexible since we can decrease its duration without increasing the length of the critical path $(1, 2, 3, 4, 5, 4, 6, 7)$. These results show possible failures of the De Reyck's method.

4 New general results

Referring to the previous analysis, it is possible to prove that:

Proposition 1. *Given a critical activity i such that the longest path in the standardized network from node 1 to node i is equal to 0, if the critical precedence relations ingoing activity i are only of type XF_{hi}^{min} and the critical precedence relations outgoing activity i are of type FX_{ij}^{min} (SX_{ij}^{min}), then activity i is not only finish-critical (backward-critical) as induced by the De Reyck's method but also forward-critical (start-critical).*

Proposition 2. *Given a critical activity i whose duration d_i is equal to the longest path from node i to node n in the standardized network, if the critical precedence relations outgoing activity i are only of type SX_{ij}^{min} and the critical precedence relations ingoing activity i are of type XS_{hi}^{min} (XF_{hi}^{min}), then activity i is not only start-critical (backward-critical) as induced by the De Reyck's method but also forward-critical (finish-critical).*

These results suggest further corrections to the standardized network and consequently to the original network. In particular, in the previous example, if we add arc $(1, 3)$ with $\ell_{13} = 0$ and arc $(5, 7)$ with $\ell_{57} = d_5$ to the standardized network, that correspond to precedence relations $SS_{13}^{min}(0)$ and $FS_{57}^{min}(0)$, respectively, we obtain that by applying the De Reyck's method we identify the correct criticality of activities 3 and 5. However, the criticality of activity 4, involved in the cycle $(4, 5, 4)$, remains incorrect. The conclusion is that, apart from the corrections, it is necessary a new method to define on a generic project network the right criticality and flexibility of each single activity.

5 Our proposal

Before outline a new method for analyzing activity criticalities and flexibilities, we redefine activity criticalities, independently from the project network representation.

Definition 1. *An activity is* critical *if its earliest and latest start (finish) times are equal.*

Definition 2. *An activity is* start-critical *if it is critical and the project duration increases only if we delay the activity start time.*

This means that, given a start-critical activity, if we maintain fixed its start time and vary (either increase or decrease) its duration, and, hence, vary (either increase or decrease) its finish time, the project duration does not change, meaning that a start-critical activity is bi-flexible. In addition, the finish time of a start-critical activity is not constrained.

Definition 3. *An activity is* finish-critical *if it is critical and the project duration increases only if we delay the activity finish time.*

This means that, given a finish-critical activity, if we maintain fixed its finish time and vary (increase or decrease) its duration, and, hence, vary (increase or decrease) its start time, the project duration does not change, meaning that a finish-critical activity is bi-flexible. In addition, the start time of a finish-critical activity is not constrained.

Definition 4. *An activity is* forward-critical *if it is critical and the project duration increases whether we delay its start time, while maintaining fixed its duration, or we increase its duration while maintaining fixed its start time (apart from project time-infeasibility).*

Therefore, in anyone of the above two cases, also the activity finish time increases, meaning that the forward-criticality *dominates* the finish-criticality. Moreover, apart from the project time-infeasibility, an increase of the duration of a forward-critical activity increases the project duration, meaning that a forward-critical activity is forward-inflexible.

Definition 5. *An activity is* backward-critical *if it is critical and the project duration increases whether we delay its finish time, while maintaining fixed its duration, or we decrease its duration while maintaining fixed its finish time (apart from project time-infeasibility).*

Hence, in anyone of the above two cases, also the activity start time increases, that is, backward-criticality *dominates* start-criticality, and, apart from the project time-infeasibility, a decrease of the duration of a backward-critical activity increases the project duration, that is, a backward-critical activity is backward-inflexible. Our definitions differ from those by De Reyck for which "an activity is forward-critical (backward-critical) if (a) it is start-critical (finish-critical), and (b) when the project duration increases when activity's duration is increased (decreased)" (cfr. p. 124 of Demeulemeester and Herroelen, 2002).

Definition 6. *An activity is* bi-critical *if it is both forward-critical and backward-critical.*

Therefore, a bi-critical activity is bi-inflexible.

We propose the following approach for analyzing activity criticalities and flexibilities, whose correctness is formally proved:

1. Adopt the AON project network representation with minimum time lags.
2. Convert the network into the standardized network (with only $SS_{ij}^{min}(\ell)$ precedences).
3. Correct the standardized network, if necessary, with the addition of new arcs outgoing from source node 1 and/or ingoing to sink node n, also on the basis of Propositions 1 and 2. Consequently, additional precedence relations of type $SS_{1i}^{min}(0)$ outgoing from node 1 and/or of type $FS_{jn}^{min}(0)$ ingoing to node n might have to be considered.
4. Find on the (corrected) standardized network the critical subnetwork composed by all the critical nodes (activities) and all the critical arcs on the standardized network.
5. Trace back the critical nodes and the critical arcs on the original AON project network in order to consider only its critical subnetwork.
6. Determine the types of criticality of each critical activity i on the basis of the precedence types of the couples of critical ingoing and outgoing arcs of i and the existence or not of elementary critical paths passing through these arc couples.
7. Determine possible project time-infeasibility of each critical activity i on the basis of the existence or not of elementary cycles traversing node i on the critical subnetwork.
8. Analyze the flexibility of non-critical activities in order to detect possible project time-infeasibility due to duration changing for these activities.

References

Bartusch, M., R.H. Möhring R.H. and F.J. Radermacher F.J., 1988, "Scheduling project networks with resource constraints and time windows", *Ann. of Oper. Res.*, Vol. 16(1), pp. 201–240.

De Reyck, B., 1998, "Scheduling Project with Generalized Precedence Constraints - Exacts and Heuristics Approaches", Ph.D. Thesis, Dept. of Appl. Econ., Kath. Univer. Leuven, Belgium.

Demeulemeester, E., W. Herroelen, 2002, *Project scheduling: a research handbook*, Kluwer, Boston.

Elmaghraby, S.E., J. Kamburowski, 1992, "The analysis of activity networks under generalized precedence relations", *Management Sci.*, Vol. 38(9), pp. 1245–1263.

Wiest, J.D., 1981, "Precedence diagramming methods: some unusual characteristics and their implications for project managers", *J. of Oper. Management*, Vol. 1(3), pp. 121–130.

A Novel Matheuristic for the Multi-Site Resource-Constrained Project Scheduling Problem

Tamara Bigler, Mario Gnägi and Norbert Trautmann

University of Bern, Switzerland
tamara.bigler@pqm.unibe.ch, mario.gnaegi@pqm.unibe.ch,
norbert.trautmann@pqm.unibe.ch

Keywords: Multi-site resource-constrained project scheduling problem, Matheuristic.

1 Introduction

In the well-known resource-constrained project scheduling problem (RCPSP), a set of precedence-related project activities must be scheduled such that the project makespan is minimized subject to limited resource availabilities. We consider a planning problem that extends the RCPSP by involving different sites. Some of the renewable resource units are mobile and can be transferred between sites while others are permanently located at one site. It is assumed that the mobile resource units are available at the site at which they are used for the first time. Transportation times apply a) when a mobile resource unit is transferred from one site to another or b) when the output of an activity is transferred to another site where one of its successor activities will be processed. In the latter case, the successor activity can only start once the outputs of all predecessor activities have arrived at the respective site. These transfers enable the sharing of resources among sites, e.g., the sharing of medical staff among hospitals. As activities from multiple sites are scheduled simultaneously in the multi-site RCPSP, the number of activities to be scheduled is often greater than in the single-site RCPSP.

The literature comprises some exact and heuristic approaches for this planning problem. Laurent et al. (2017) introduced a discrete-time mathematical model that they applied to instances involving 5 to 30 activities. Because their model did not seem to scale well, they developed four metaheuristics, which are based on an activity list and a site list representation of the solution. One metaheuristic conducts a local search, one metaheuristic is based on simulated annealing, and two metaheuristics perform an iterated local search. They applied the four metaheuristics to instances comprising 30 to 120 activities. It turned out that the iterated local search and the simulated annealing metaheuristics perform best. Gnägi and Trautmann (2019; 2021) formulated a continuous-time mathematical model that they applied to the same instances as Laurent et al. (2017) comprising 30 activities. Their model was able to derive a large number of new best known solutions for these instances.

In this paper, we propose a novel matheuristic for the multi-site RCPSP. In the matheuristic, the activities are scheduled by performing the following two steps in an iterative manner. First, a subset of activities that will be scheduled in the next iteration is selected based on standard priority rules from the literature. Second, the selected activities are scheduled by solving a relaxation of the model of Gnägi and Trautmann (2019; 2021). The matheuristic obtains high-quality solutions for instances comprising 30 activities and 2 or 3 sites. Among the 960 tested instances from the literature, it derives new best known solutions for 164 instances.

The remainder is structured as follows. In Section 2, we illustrate the planning problem with an example. In Section 3, we outline the novel matheuristic. In Section 4, we report the computational results. In Section 5, we conclude and give an outlook on future research.

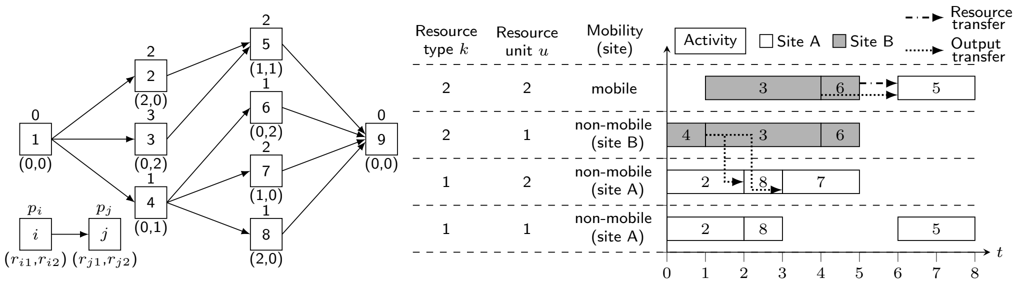


Fig. 1. Example: activity-on-node network (left) and an optimal solution (right)

2 Illustrative example

In this section, we illustrate the planning problem with an example that comprises seven real activities $\{2, \ldots, 8\}$. The fictitious activities 1 and 9 represent the project start and completion, respectively. The left part of Figure 1 shows an activity-on-node network. Each node in the network represents one activity and each arrow represents one precedence-relationship. Moreover, the example includes two sites A and B between which we assume a transportation time of one time unit. Each of the two resource types $k = 1$ and $k = 2$ comprises two resource units $u = 1$ and $u = 2$. Both resource units of resource type $k = 1$ are non-mobile and permanently located at site A. One unit ($u = 2$) of resource type $k = 2$ is mobile, and the other unit ($u = 1$) is permanently located at site B. The activity-on-node network further shows for each activity i its resource requirement r_{ik} for the two resource types and its duration p_i. The right part of Figure 1 shows an optimal solution for the illustrative example. Each line represents a resource unit u of a resource type k, and the rectangles represent the activities. Each real activity is assigned to at least one resource unit and exactly one site. The activities $\{2, 5, 7, 8\}$ are executed at site A while the activities $\{3, 4, 6\}$ are executed at site B. The resource transfers are indicated by a dash-dotted arrow, e.g., between activities 6 and 5 which take place at a different site; thus, the commonly used resource unit $u = 2$ of resource type $k = 2$ must be transferred from site B to site A. The output transfers are indicated by a dotted arrow, e.g., between activities 4 and 8, which are precedence-related and take place at a different site; thus, the output of activity 4 must be transferred from site B to site A before activity 8 can start.

3 Novel matheuristic

In this section, we describe the novel matheuristic in more detail and illustrate it with the example from Section 2. The matheuristic is based on the continuous-time sequencing/natural-date model of Gnägi and Trautmann (2019), subsequently referred to as GT19. The model involves continuous start-time variables that indicate the start time of an activity, and binary site-selection variables that represent the execution site for each activity. Moreover, it includes binary resource-assignment variables that assign the activities to the resource units, and binary sequencing variables y_{ij} that indicate the sequence between pairs of activities i and j. Hence, $y_{ij} = 1$ means that activity i is scheduled before activity j.

Our matheuristic is based on a variant of GT19, in which some sequencing variables are relaxed, i.e., they can take any fractional value between 0 and 1, and some activities are locked, i.e., the site-selection and resource-assignment variables of these activities as well as the sequencing variables between all pairs of these activities are fixed to their values in the current solution of the relaxation. Before the first iteration, the matheuristic derives promising initial values for the site-selection variables by solving a relaxation of GT19,

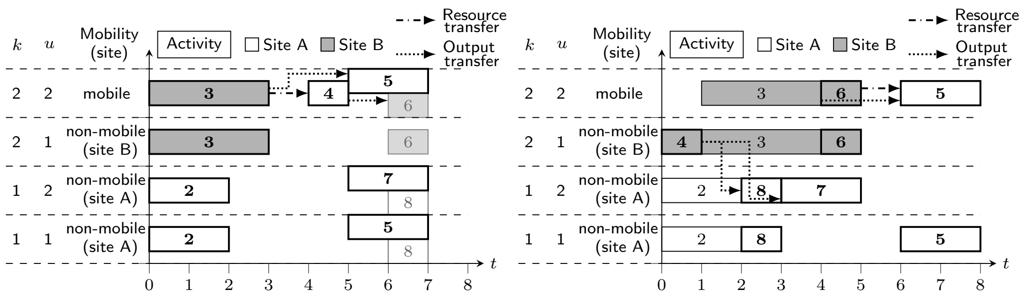


Fig. 2. Example: schedule after Iteration 1 (left) and schedule after Iteration 2 (right)

in which all sequencing variables are relaxed. Then, the matheuristic iteratively schedules changing subsets of activities in a rolling-horizon manner as follows. First, $2b$ activities are selected based on the latest starting time (LST) priority rule and the latest finishing time (LFT) priority rule as a tie breaker. Second, a relaxation of GT19 is solved, in which only the sequencing variables among and between the $2b$ selected and the locked activities are defined as binary; all remaining sequencing variables are relaxed. Moreover, the initial values for the site-selection variables of the $2b$ selected activities are provided. The solution of this relaxation provides a schedule in which the sequence of the $2b$ selected and the locked activities is determined. The remaining activities (subsequently referred to as eligible activities), however, may overlap among each other as well as with the selected and the locked activities. Finally, the initial values for the site-selection variables are updated based on the current solution of the relaxation, and the b activities with the highest priority (according to the combined LST and LFT priority rule) among the $2b$ selected activities are locked. Consequently, b activities remain selected. Then, the next iteration starts by selecting b additional activities from the eligible activities based on the combined LST and LFT priority rule. If there are no eligible activities to select in this step, the matheuristic stops.

In the illustrative example, we set $b = 3$. Figure 2 illustrates the resulting two iterations. The selected activities are marked in bold, the eligible activities are transparent, and the locked activities are not highlighted. In Iteration 1, the activities $\{1, 3, 2, 4, 5, 7\}$ are selected. Figure 2 (left) shows the schedule obtained in Iteration 1, in which some of the eligible activities $\{6, 8, 9\}$ overlap with some of the selected activities. This is feasible in this iteration because all sequencing variables between the eligible activities and the selected activities are relaxed. Next, the activities $\{1, 3, 2\}$ are locked and the activities $\{6, 8, 9\}$ are selected in addition to the already selected activities $\{4, 5, 7\}$. As all activities are either selected or locked in Iteration 2, the sequencing variables between all activities are defined as binary and the conflicts between the activities $\{5, 6\}$, $\{5, 8\}$, and $\{7, 8\}$ must be resolved. Figure 2 (right) illustrates the schedule obtained in Iteration 2. Compared to Iteration 1, activity 4 is scheduled at a different time, at a different site, and on a different resource unit. Without the site change of activity 4 in Iteration 2, an additional transportation time would apply, which would delay the project makespan by one time unit. After performing Iteration 2, there are no eligible activities and the matheuristic stops.

4 Computational results

In this section, we present the computational results. The matheuristic was tested on 960 instances comprising 30 activities and 2 or 3 sites that belong to the test set MSj30. This set has been adapted to the multi-site context by Laurent et al. (2017) from the instances of the PSPLIB by Kolisch and Sprecher (1996). We implemented the matheuristic

Table 1. Computational results

	Overall	Laurent et al. (2017)		Gnägi and Trautmann (2021)	
# Sites	# New BKS	# Better	∅ Gap [%]	# Better	∅ Gap [%]
2	63	110	0.70	94	0.38
3	101	143	0.53	144	-0.57

in Python 3.7 and used the Gurobi 9.1 solver. We prescribed a time limit of 300s to the Gurobi solver in each iteration. Moreover, we set $b = 5$.

Table 1 summarizes the results obtained. The solutions of the matheuristic are compared to the best known solutions that Laurent et al. (2017) published for their metaheuristics on their website and to the solutions Gnägi and Trautmann (2021) reported for their continuous-time mathematical model. We group the results by the number of sites (2 or 3). The column # New BKS corresponds to the number of instances for which the matheuristic found a new best known solution. The columns # Better report the number of instances for which the matheuristic obtained a better solution than the approaches of Laurent et al. (2017) or Gnägi and Trautmann (2021), respectively, and the columns ∅ Gap report the average gap of the matheuristic solutions to the solutions of the approaches of Laurent et al. (2017) or Gnägi and Trautmann (2021), respectively. Even though the average gap to the benchmark approaches is overall slightly positive, our matheuristic is able to derive 164 new best known solutions in a shorter average running time than the benchmark approaches.

5 Conclusions and outlook

In this paper, we studied a variant of the RCPSP that involves multiple sites. This extension allows for resource pooling among sites and introduces two types of transportation times that must be considered. We developed a matheuristic for this problem that derives high-quality solutions for a standard test set of instances with 30 activities and 2 or 3 sites.

In future research, the matheuristic could be extended by an LP-based improvement step involving the so-called justification technique. This technique has been shown to improve schedules considerably while running times increase only slightly (cf. Valls et al., 2005). Also, the planning problem could be extended to take into account resources that are required for the transfer of the output of an activity to another site (Krüger and Scholl, 2010).

References

Gnägi, M., and Trautmann, N., 2019, "A continuous-time mixed-binary linear programming formulation for the multi-site resource-constrained project scheduling problem.", In: Wang, M., Li, J., Tsung, F., and Yeung, A. (eds.): *Proceedings of the 2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Macau, pp. 382–365.

Gnägi, M., and Trautmann, N., 2021, "A continuous-time model for the multi-site resource-constrained project scheduling problem", In: *Proceedings of the 17th International Conference on Project Management and Scheduling (PMS)*, Toulouse, to appear.

Kolisch, R., and Sprecher, A., 1996, "PSPLIB-a project scheduling problem library.", *European Journal of Operational Research*, Vol. 96(1), pp. 205–216.

Krüger, D., and Scholl, A., 2010, "Managing and modelling general resource transfers in (multi-)project scheduling", *OR Spectrum*, Vol. 32(2), pp. 369–394.

Laurent, A., Deroussi, L., Grangeon, N., and Norre, S., 2017, "A new extension of the RCPSP in a multi-site context: Mathematical model and metaheuristics.", *Computers & Industrial Engineering*, Vol. 112, pp. 634–644.

Valls, V., Ballestin, F., and Quintanilla, S., 2005, "Justification and RCPSP: A technique that pays.", *European Journal of Operational Research*, Vol. 165(2), pp. 375–386.

Solution Repair by Inequality Network Propagation in LocalSolver

Léa Blaise[1 2], Christian Artigues[1], Thierry Benoist[2]

[1] LAAS-CNRS, Université de Toulouse, CNRS, INP, Toulouse, France
[2] LocalSolver, 36 Avenue Hoche, 75008, Paris, France
`lblaise@localsolver.com`

Keywords: constraint propagation, inequality networks, local search, solution repair, solver, disjunctive scheduling.

1 Introduction and context

This paper introduces a solution repair algorithm based on constraint propagation, designed to overcome the difficulties met by small neighborhood search on a family of tightly constrained problems.

LocalSolver is a mathematical programming solver, whose goal is to offer a model-and-run approach to optimization problems, including combinatorial, continuous, and mixed problems, and to offer high quality solutions in short running times, even on large instances. It allows OR practitioners to focus on the modeling of the problem using a simple formalism, and then to defer its actual resolution to a solver based on efficient and reliable optimization techniques, including local search (but also linear, non-linear, and constraint programming). The local search algorithms implemented in LocalSolver are described in Gardi *et. al.* (2014).

This paper focuses on problems whose constraints include a network of two-variable inequalities. More precisely, it focuses on problems whose constraints comprise either linear inequalities between two variables or disjunctions of such inequalities. Many such problems arise in the field of scheduling: for example the Job Shop Problem (Fisher and Thompson 1963) and the Bridge Building Problem (Bartusch 1983), which are both characterized by generalized precedences and disjunctive resource constraints. However, this kind of structure is also typical of packing, layout, and mining problems.

These problems are highly constrained: in a good solution of a Job Shop or Bridge instance, the precedences and disjunctive resource constraints are often very tight. Because of that, moving from a solution of makespan x to a solution of makespan $x-1$ requires a lot of small changes on many integer variables – the start times of the tasks. Being able to move from a good feasible solution to another using random small neighborhoods is then very unlikely: one would have to randomly target the right set of integer variables and to randomly shift them all by the right amount. For these reasons, the algorithms described in Gardi *et. al.* (2014) encounter serious difficulties on these problems. In the vast literature on job-shop scheduling by local search, these difficulties are overcome by exploiting higher level dedicated solution representations, such as the disjunctive graph (Vaessens *et. al.* 1994). In this work, we aim at keeping the modeling elements simple and we wish to target other problems as well. Hence, we focus on the direct integer variable representation.

2 Solution repair

The solution that we envisioned and implemented in LocalSolver to tackle this problem is a kind of constraint propagation: a promising but infeasible solution is gradually repaired one constraint at a time. This gradual procedure might remind one of the min-conflicts heuristic, used to repair an inconsistent assignment for the variables of a CSP, or of ejection chains algorithms, both recently studied in the context of scheduling problems, respectively in Ahmeti and Musliu (2018) and Ding *et. al.* (2019).

Before the search, specific constraints (such as generalized precedences and disjunctive resource constraints) are detected in the model and made into a constraint network. Those constraints will be the ones on which the repairing algorithm will apply when necessary. Let's consider any iteration of the local search, and let's assume that the current solution $\mathcal{S}_0$ is feasible. A local transformation is applied to the solution (for example, modification of a few start times), rendering it infeasible ($\mathcal{S}$). In the simplest case, the repair phase is similar to the classic constraint propagation of Constraint Programming (Rossi *et. al.* 2006), but has a few singularities. First, while constraint propagation generally aims at reducing the variables' domains, here the constraints are only propagated when they are violated and need repairing. Indeed, our only concern here is to build a feasible solution $\mathcal{S}'$ as close to $\mathcal{S}$ as possible. Then, we impose that, during the local move and along the successive constraint repairs, the variables must always be shifted in the same direction: if a variable's value was increased, it can still be increased further when repairing a constraint, but it can never be decreased, and reciprocally. This ensures that all the decisions taken during the current iteration (during the successive constraint repairs, and more importantly during the local move) are respected. This way, the successive repairs "follow the move's direction": they lead to finding a feasible solution as close as possible to $\mathcal{S}$, by amplifying the move rather than by cancelling it. At the end of the propagation, the algorithm was either able to repair all the constraints, and thus found a feasible solution $\mathcal{S}'$, or found a constraint that could not be repaired. In the latter case, all the changes are cancelled and the whole procedure starts over: the current solution is set back to $\mathcal{S}_0$, and a new move is applied.

2.1 Two-variable linear inequalities

We consider inequalities of the form

$$aX + bY \leq c$$

where X and Y are integer or floating point variables, and a, b, and c are any constants. When $a = 1$ and $b = -1$, these inequalities correspond to the generalized precedence constraints encountered in scheduling problems, but the following algorithm is not limited to this specific case.

Let's assume that the inequality $aX + bY \leq c$ is violated. Since the initial solution was feasible, at least X or Y has already been shifted in the "wrong" direction, and therefore cannot be shifted now in the repair direction. At most one variable (X by symmetry) can then be shifted in the repair direction: there exists only one way to repair the constraint, which consists in shifting X just enough ($X \leftarrow \frac{c-bY}{a}$). This is a necessary decision, as in every feasible solution following the move's choices, X is at least (resp. at most) $\frac{c-bY}{a}$.

When the only repairable constraints in the model are inequalities, the repair phase is equivalent to a particular kind of constraint propagation, which will be referred to as "half bound consistency" in the remainder of the paper. The reduction of a variable X's domain is propagated only if it excludes its current support x (only if the constraint is violated and needs repairing). After repairing the constraint, the new support of X is written x'. When propagating the reduction of X's domain, only one of its bounds is modified: either $x' > x$ (then X's value can no longer be decreased) and x' is its new lower bound, or $x' < x$ and x' is its new upper bound. Since each variable must always be shifted in the same direction, one of its bounds at most can be modified throughout the propagation.

An interesting property when the only repairable constraints in the model are inequalities is that, if there exists a feasible solution that respects the decisions of the local move, the algorithm is guaranteed to find one at the end of the propagation.

2.2 Disjunctions of two-variable linear inequalities

We consider disjunctions of inequalities of the form

$$\bigvee_i (a_i X_i + b_i Y_i \leq c_i)$$

where the X_i and Y_i are integer or floating point variables, and the a_i, b_i, and c_i are any constants. When $a_i = 1$ and $b_i = -1\ \forall i$, these disjunctions correspond to disjunctive resource constraints (disjunctions of size 2), or packing constraints in higher dimensions. However, the algorithm is not limited to this specific case here either.

Let's assume that a disjunction is violated. We will try to repair it in a non deterministic way. Since *a priori* none of the generalized precedences of the disjunction should prevail over the others, the algorithm chooses one at random and tries to repair it. If it cannot be repaired, it tries to repair the following one, and so forth. If none of them could be repaired, the propagation fails.

Let $aX + bY \leq c$ be the inequality that was randomly chosen for repair in the disjunction. If only one of its variables can be shifted in the repair direction, then the constraint is repaired as described in 2.1. It is also possible that both variables can be shifted in the repair direction, since the chosen inequality may not have been the one that was respected in the initial feasible solution $\mathcal{S}_0$. If so, the algorithm randomly chooses how to shift them. Let Δ be the distance to feasibility: $\Delta = aX + bY - c$, and let δ_X and δ_Y be the shares of the repair respectively attributed to X and Y, verifying $\delta_X + \delta_Y = \Delta$. There are four possible ways for the algorithm to repair the constraint: either X repairs it alone ($\delta_X = \Delta$), or Y repairs it alone ($\delta_Y = \Delta$), or X and Y equitably share the repair ($\delta_X = \delta_Y = \frac{\Delta}{2}$), or X and Y randomly share the repair ($\delta_X = random(1, \Delta - 1)$ and $\delta_Y = \Delta - \delta_X$).

Since the repair procedure of a disjunction is non deterministic, the previous properties, of half bound consistency and guarantee to find a feasible solution, do not hold anymore when such constraints are detected among the repairable constraints in the model. However, if there exists a solution that respects the decisions of the move, there is always a non-zero probability that the propagation leads to finding one, depending on whether the algorithm always takes the "right" random decisions when repairing a disjunction.

3 Application to scheduling problems

This method of solution repair by constraint propagation dramatically improves the performances of LocalSolver on problems with a network of two-variable linear inequalities.

3.1 Results on the Bridge Building Problem

The optimum value of the Bridge Building Problem is 104. Without our repair mechanism, within ten seconds of search, LocalSolver 9.0 is only able to find solutions of value 115 on average, and virtually never finds an optimum solution. But with the integrated repair mechanism, it always finds a solution of value 104 within four seconds of search, and very often finds one in less than one second.

3.2 Results on the Job Shop Problem

We compared the performances of LocalSolver 9.0 with and without this repair mechanism on three classic Job Shop instance classes: the FT class by Fisher and Thompson (1963), the LA class by Lawrence (1984), and the ORB class by Applegate and Cook (1991).

Table 1. Evolution of the gap between the average solution found by LocalSolver and the optimum solution, in 10 seconds and in 60 seconds, on different Job Shop instance classes

	Gap 10s		**Gap 60s**	
Instance class	**No repairs**	**With repairs**	**No repairs**	**With repairs**
FT	73%	7%	15%	3%
LA	246%	10%	91%	4%
ORB	120%	6%	22%	4%

As shown above in table 1, our repair mechanism enables LocalSolver not only to find very good solutions of many Job Shop instances, but also to find good solutions very quickly: less than 10% from the optimum within 10 seconds of search.

4 Conclusion

In this paper, we considered a family of problems, whose constraints comprise a network of two-variable linear inequalities. Although small neighborhood search algorithms may encounter serious difficulties on these problems, we introduced a solution repair algorithm based on constraint propagation, overcoming the difficulties met by small neighbourhood search. The two main specificities of our propagation algorithm are that a domain reduction is only propagated if it excludes the current support of the variable, and that each variable must always be shifted in the same direction. Its integration into LocalSolver dramatically improves its performances on the targeted problems, not only on classic scheduling problems such as the Job Shop Problem, but also on some 3D packing and mining industrial instances of our test base.

References

Ahmeti A. and N. Musliu, 2018, "Min-conflicts Heuristic for Multi-mode Resource-constrained Projects Scheduling", *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 237-244.

Applegate D., W. Cook, 1991, "A computational study of the job-shop scheduling problem", *ORSA Journal on Computing* 3, pp 149-156.

Bartusch M., 1983, "Optimierung von Netsplanen mit Anordnungsbeiziehungen bei knappen Betriebsmitteln", PhD thesis, Universitat Passau, Fakultat fur Matematik und Informatik.

Ding J., L. Shen, Z. Lü, and B. Peng, 2019, "Parallel machine scheduling with completion-time-based criteria and sequence-dependent deterioration", *Computers and Operations Research* (103), pp. 35-45.

Fisher H., G. Thompson, 1963, "Probabilistic learning combinations of local job-shop scheduling rules", *Industrial Scheduling*, Muth J., G. Thompson (Eds.), Prentice Hall, Englewood Cliffs, New Jersey, pp. 225-251.

Gardi F., T. Benoist, J. Darlay, B. Estellon, and R. Megel, 2014, "Mathematical Programming Solver Based on Local Search", Wiley.

Lawrence S., 1984, "Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement), Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburg, Pennsylvania.

Rossi F., P. Van Beek, and T. Walsh, 2006, "Handbook of Constraint Programming (Foundations of Artificial Intelligence)", Elsevier Science Inc., New York, NY, USA.

Vaessens, R. J. M., E. H. L. Aarts, and J. K. Lenstra, "Job shop scheduling by local search", *Informs Journal on computing*, 8(3), pp. 302-317.

The generalised resource-constrained project scheduling problem with flexible resource profiles

Matthew Bold[1], Burak Boyaci[2], Marc Goerigk[3] and Chris Kirkbride[2]

[1] STOR-i Centre for Doctoral Training, Lancaster University, United Kingdom
m.bold1@lancaster.ac.uk
[2] Lancaster University Management School, Lancaster University, United Kingdom
[3] Network and Data Science Management, University of Siegen, Germany

Keywords: project scheduling, flexible resource profiles, generalised precedence constraints, schedule generation scheme, genetic algorithm.

1 Introduction

The classical resource-constrained project scheduling problem (RCPSP) consists of scheduling a set of activities, subject to resource and precedence constraints, in order to minimise the project makespan. The applicability of the RCPSP, however, is limited by the following two assumptions: 1. only finish-to-start, zero time-lag precedence relationships exist between activities, and 2. the resource requirements of each activity are fixed and constant throughout its duration. In practice, rarely do these assumptions hold true.

The extension of the RCPSP to include generalised precedence relationships addresses the first limiting assumptions, and is a well studied problem for which a number of exact (Bartusch, Möhring & Radermacher 1988, De Reyck & Herroelen 1998, Schutt, Feydy, Stuckey & Wallace 2013) and heuristic (Franck, Neumann & Schwindt 2001, Ballestín, Barrios & Valls 2011) solution methods have been developed. We refer to this problem as the generalised resource-constrained project scheduling problem (GRCPSP).

More recently, the resource-constrained project scheduling problem with flexible resource profiles (FRCPSP) has been introduced to address the second limiting assumption of the RCPSP. This problem assumes only that the total amount of resource that is required to complete each activity is known, and that, as well as the start time, the resource allocation for each activity must be determined. Whilst heuristic approaches have provided the most success in solving the FRCPSP (Fündeling & Trautmann 2010, Tritschler, Naber & Kolisch 2017), a number of exact mixed-integer programming (MIP) formulations have also been developed (Naber & Kolisch 2014, Naber 2017).

Here, we introduce the generalised resource-constrained project scheduling problem with flexible resource profiles (GFRCPSP) to combine these two extensions of the RCPSP into a single model. The GFRCPSP is an NP-hard problem for which realistically sized instances cannot be solved exactly, and hence, in addition to proposing a MIP formulation for this problem, we also propose a genetic algorithm (GA) based on a non-greedy serial schedule generation scheme with an unscheduling step.

2 Problem description

A project consists of a set of non-preemptive activities $V = \{0, 1, \dots, n, n+1\}$, where 0 and $n+1$ are dummy source and sink activities. The GFRCPSP consists of determining a start time and resource profile of each activity, subject to a set of resource constraints and generalised precedence relationships, in order to minimise the project makespan.

There exist four types of generalised precedence relationship: start-to-start, start-to-finish, finish-to-start and finish-to-finish. Every generalised precedence relationship in a project has an associated minimal or maximal time-lag, which together create a feasible time-window for the processing of each activity. In the GRCPSP, since the duration of each activity is known a priori, each of the four types of relationship can be transformed into a single type (Bartusch et al. 1988). In the GFRCPSP however, since activity durations are variables, these transformations do not apply, and each relationship type remains distinct. Maximal time-lags however, can be converted into negative minimal time-lags going in the opposite direction, which allows the set of project precedence constraints to be represented on a network, such as the one shown in Figure 1.

We define resource constraints for the GFRCPSP in the same way that Naber & Kolisch (2014) define them for the FRCPSP. The total resource allocated to each task $i \in V$ over its duration must at least satisfy its total resource requirement w_i, whilst adhering to upper and lower bounds on its per period resource allocation, $\underline{q}_i$ and $\overline{q}_i$, and a so-called *minimum block length*, l^{min} (Fündeling & Trautmann 2010), that is, the minimum number of time periods for which the resource allocation to an activity must remain constant. All resources $r \in R$ are assumed to be renewable, continuous, and have limited availability R_r^{max}. It is also assumed that there are three types of resource: principle, dependent and independent. The allocation of principle resource to an activity determines the allocation of each dependent resource to that activity through a linear resource-function. The allocation of independent resources to an activity is fixed and independent of the allocation of the other resources.

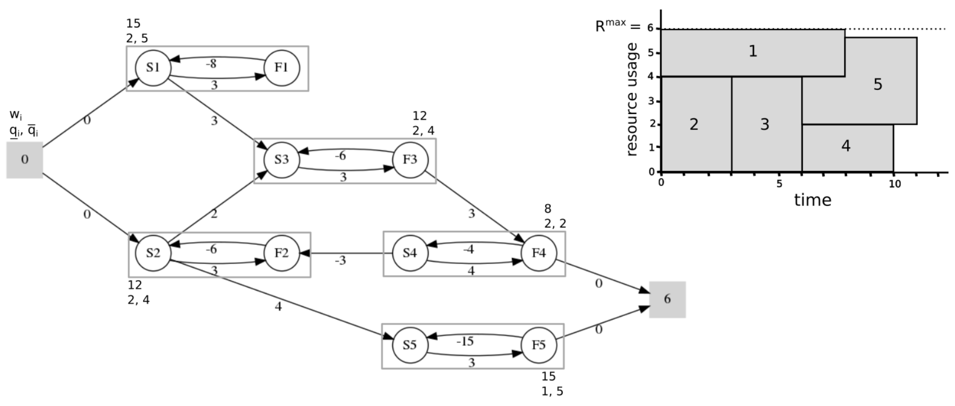


Fig. 1. An example GFRCPSP network with five non-dummy activities, a $l^{\text{min}} = 2$, and a single resource with $R^{\text{max}} = 6$. Arc labels indicate lower bounds on precedence relations. The chart shows an optimal solution to this problem.

3 Solution approaches

The GFRCPSP can be modelled by adapting the FP-DT3 model proposed by Naber & Kolisch (2014) for the FRCPSP. For brevity, we omit this formulation from this abstract. Solving this MIP model becomes intractable as instance sizes get larger, and hence we also propose a schedule generation scheme-based heuristic (Kolisch & Hartmann 1999) and GA for finding good solutions to realistically-sized instances in a reasonable time. We outline this scheduling heuristic and GA here.

The scheduling heuristic we propose is a non-greedy serial schedule generation scheme (SGS) with unscheduling step. This algorithm takes an activity list as input, and constructs a complete solution by scheduling activities one at a time in the given order. Each activity is started as early as possible and with as much resource as possible, subject to 'delay' and 'greediness' parameters (Tritschler et al. 2017), which are used to encourage non-greedy scheduling. This is desirable, since an optimal solution for a given instance cannot necessarily be found using a purely greedy SGS (Fündeling & Trautmann 2010).

If a resource constraint is violated whilst scheduling an activity, the algorithm attempts to re-start the activity at the next resource-feasible start time. If a precedence constraint is violated, an unscheduling step is invoked to reschedule the activities that cause either the missed latest start or latest finish time, and start them closer to the activity with which they have a maximum time-lag with the aim of restoring the feasibility of the schedule. If the unscheduling step fails to feasibly reschedule the activities after a given number of attempts, the schedule is completed infeasibly and the total number of time periods by which precedence constraints are missed is recorded.

A GA is used to search over individual solutions, each of which consists of an activity list, and delay and greediness parameters for each activity. An initial population of individuals is generated randomly subject to precedence feasibility. Each individual is scheduled using the above heuristic. The 'fitness' of a feasible schedule is equal to its makespan, whilst the fitness of an infeasible schedule is equal to the total number of time periods by which precedence relationships are missed, plus a fixed penalty. New individuals are obtained using the adapted two-point crossover of Franck et al. (2001), which is designed to keep activities that are related by a maximal time-lag close together in the resulting activity list, thus increasing the likelihood of finding a feasible solution. The mutation operator of Hartmann (1998) is applied to each offspring solution. Having produced enough new individuals to double the original population size, the next generation is chosen using 3-tournament selection. The next generation has the same size as the original population.

4 Results and conclusions

Table 1 compares the results of the proposed GA and scheduling heuristic with the results of solving the FP-DT3-based MIP model. The two approaches are tested over five sets containing instances with 10, 20, 30, 50 and 100 activities respectively. Each of the five sets set contains 30 instances for three different values of resource strength (Kolisch, Schwindt & Sprecher 1999), resulting in a total of 450 instances. Resource strength (RS) measures the restrictiveness of the resource availability, with a smaller RS generally indicating a more challenging problem. These instances have been created using a new GFRCPSP instance generator we have developed as an extension to the ProGen/max project generator (Kolisch et al. 1999).

Table 1 shows the average percentage gap to the critical-path based lower bound of solutions found by the two solution methods. For each instance, the GA searched 50,000 schedules, whilst a limit of 2 hours was allowed for solving the MIP. To enable a fair comparison between the two solution methods, the instances for which both approaches find a feasible solution have been presented separately from those for which only the GA finds a feasible solution. There are no instances where only the MIP finds a feasible solution. These results show the MIP performing well on instances with 10 and 20 activities, but dramatically worsening over the larger test sets, as expected. In contrast, the quality of the solutions found by the GA remain roughly constant across the five test sets.

In conclusion, the GFRCPSP has been introduced to combine two existing extensions to the RCPSP. The GFRCPSP can be solved for small instances as an MIP, whilst a new scheduling heuristic and GA has been proposed for solving larger instances. Future work will include the application of this new model to a real-world scheduling problem, as well as further improvements to the metaheuristic approach proposed here, perhaps with the introduction of a local improvement step.

Test set	RS	MIP & GA #	MIP & GA Δ_{lb}^{MIP}	MIP & GA Δ_{lb}^{GA}	Only GA #	Only GA Δ_{lb}^{GA}
	0.05	30	10.90	13.14	0	-
P10	0.15	30	2.17	3.24	0	-
	0.25	30	0.49	0.49	0	-
	0.05	30	19.37	19.65	0	-
P20	0.15	30	0.25	0.49	0	-
	0.25	30	0.00	0.00	0	-
	0.05	28	150.81	13.43	2	10.71
P30	0.15	29	0.00	0.00	1	0
	0.25	30	14.56	0.00	0	-
	0.05	0	-	-	30	25.24
P50	0.15	6	617.23	0.00	24	0.00
	0.25	29	769.84	0.00	1	0.00
	0.05	0	-	-	30	18.81
P100	0.15	3	1029.66	0.00	27	0.00
	0.25	29	1442.55	0.00	1	0.00

Table 1. Average percentage gap to the critical path-based lower bound of solutions found by the MIP and GA. These values are denoted by Δ_{lb}^{MIP} and Δ_{lb}^{GA}, respectively.

References

Ballestín, F., Barrios, A. & Valls, V. (2011), 'An evolutionary algorithm for the resource-constrained project scheduling problem with minimum and maximum time lags', *Journal of Scheduling* **14**(4), 391–406.

Bartusch, M., Möhring, R. H. & Radermacher, F. J. (1988), 'Scheduling project networks with resource constraints and time windows', *Annals of Operations Research* **16**(1), 199–240.

De Reyck, B. & Herroelen, W. (1998), 'A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations', *European Journal of Operational Research* **111**(1), 152–174.

Franck, B., Neumann, K. & Schwindt, C. (2001), 'Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling', *OR-Spektrum* **23**(3), 297–324.

Fündeling, C.-U. & Trautmann, N. (2010), 'A priority-rule method for project scheduling with work-content constraints', *European Journal of Operational Research* **203**(3), 568–574.

Hartmann, S. (1998), 'A competitive genetic algorithm for resource-constrained project scheduling', *Naval Research Logistics* **45**(7), 733–750.

Kolisch, R. & Hartmann, S. (1999), Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis, *in* J. Weglarz, ed., 'Project scheduling', Vol. 14, Springer, Boston, MA, pp. 147–178.

Kolisch, R., Schwindt, C. & Sprecher, A. (1999), Benchmark instances for project scheduling problems, *in* J. Weglarz, ed., 'Project scheduling', Vol. 14, Springer, Boston, MA, pp. 197–212.

Naber, A. (2017), 'Resource-constrained project scheduling with flexible resource profiles in continuous time', *Computers & Operations Research* **84**, 33–45.

Naber, A. & Kolisch, R. (2014), 'MIP models for resource-constrained project scheduling with flexible resource profiles', *European Journal of Operational Research* **239**(2), 335–348.

Schutt, A., Feydy, T., Stuckey, P. J. & Wallace, M. G. (2013), 'Solving rcpsp/max by lazy clause generation', *Journal of scheduling* **16**(3), 273–289.

Tritschler, M., Naber, A. & Kolisch, R. (2017), 'A hybrid metaheuristic for resource-constrained project scheduling with flexible resource profiles', *European Journal of Operational Research* **262**(1), 262–273.

Why and how to evaluate the task threatness

Brožová Helena[1], Šubrt Tomáš[2], Rydval Jan[3], Pavlíčková Petra[4]

Czech University of Life Sciences Prague, Fac. of Economics and Management,
Dept. of Systems Engineering, Kamýcká 129, 165 21 Praha 6 – Suchdol, CZ
[1]brozova@pef.czu.cz, [2]subrt@pef.czu.cz, [3]rydval@pef.czu.cz, [4]pavlickovap@pef.czu.cz,

Keywords: Project management, project success, task impact, criticalness, failureness, threatness.

1. Introduction

Nowadays, the overwhelming majority of projects fail or end up with more or less problems. It concerns over half of all large complex industrial projects (Aschman, 2018, Betz, 2018). The main factors of the projects fail are budget overspending, schedule slipping, a lot of project changes gradually requested by customers, and/or severe and continuing operational problems holding for at least one year (Aschman, 2018).

Generally only methods and process of risk analysis are in the focus of many authors which suggest new, more exact approaches to this analysis (Williams, 2017, APM, 2008). Almost no authors mention the need to analyse the project in terms of the implementation of individual tasks, of their time parameters, costs and work. Commonly, methods for creating a project schedule (based on critical path) or determining the budget and resource requirements are only used but they are not aimed at predicting and preparing for potential threats of individual task. The criticality of project tasks is often defined from a time perspective only, using stochastic approaches (Bowers, 1996, Cruz et al, 1999), fuzzy sets methods (Chen, Huang, 2007, Yakhchali, 2012) or using the findings of a network analysis (Chanas, Zielinski 2003). Gong and Rowings (1995) mention that ignoring the impact of non-critical tasks, which may easily become critical, is the most frequent criticism of project duration analysis methods. Another point of view on tasks criticalness is given by the structure of relations in the project. Bowers (1996) or Williams (1992) deal with a stochastic analysis of a project network where the criticality of tasks in the project is derived from the relation between task duration and the whole project, and on the basis of a number of resources used for a task and the whole project. Another approach to analysis of the project performance is based on multiple attribute evaluation (Koelmans, 2004, de Oliveira Moraes, Laurindo, 2013).

For these reasons we introduce the task threatness matrix, our proposed tool for analysis of the criticalness and failureness potential of the project tasks. The main advantage of this tool is its similarity to project risk matrix and relatively easily obtainable data.

2. Task criticalness, failureness and threatness concept

The concept of threatness of project task combines two views – task criticalness and failureness.

The task criticalness potential (Brozova et al. 2014, 2016) is suggested to provide the overall evaluation of the task criticalness using quantitative crisp evaluation without soft knowledge about character of the tasks (Figure 1). The task criticalness potential is based on the multiple attribute decision making method using five indicators of the criticalness which are based on objective values from project schedule and are transformed using linear utility function and fuzzified using fuzzy linguistic scale:

- Duration – longer task duration means higher value of time criticalness indicator,
- Slack – shorter task slack means higher value of slack criticalness indicator,
- Cost - higher task cost means higher value of cost criticalness indicator,
- Work - higher task work means higher value of work criticalness indicator, and
- Topology - higher probability the activity will lie on critical pass related to the project topology means higher value of topological criticalness indicator.

The task failureness potential (Brozova et al. 2016, 2019) is based on the expert estimation of the possibility of task fails from different even soft aspects considering the role of human factor which are expressed using fuzzy linguistic scale (Figure 1). The task failureness indicators are primarily derived from the project triangle criteria respecting three key parameters (and can describe also other parameters of project tasks):

- Duration – higher possibility of task duration extension means higher value of time failureness indicator,
- Cost – higher possibility of task cost increasing means higher value of cost failureness indicator,
- Quality – higher possibility of task quality deterioration means higher value of quality failureness indicator.

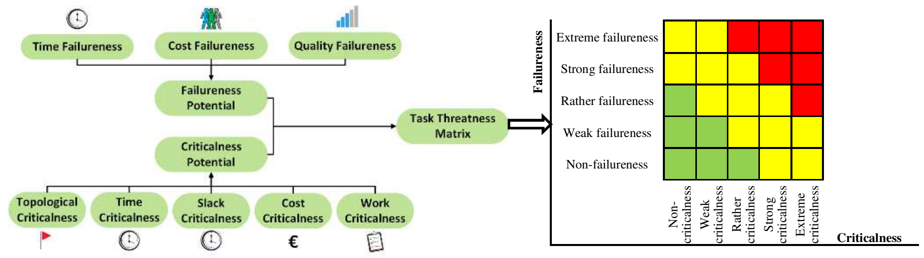


Figure 1. Factors of task threatness and task threatness matrix

The task threatness is obtain using the fuzzy linguistic evaluation of the task criticalness and failureness potentials as two-dimensional evaluation of task. The fuzzy values of criticalness and failureness potential are used for placing of the tasks into cells of task threatness matrix (Figure 1) which is inspired by Winterlink's matrix.

3. Approaches to tasks evaluation

The evaluation of all indicators can be crisp values (numbers) or fuzzy values (actually a fuzzy linguistic value). The first one is used for objective evaluation and the second one has its advantage for subjective evaluation of failureness factors and for division of tasks into five groups.

The crisp evaluation of criticalness factors is based on the objective parameters obtained from the project schedule. Each task criticalness indicator transforms the task parameter so that the best value of this parameter corresponds to the value 0 meaning the lowest criticalness and the worst parameter value corresponds to the value 1 showing the higher criticalness. Then the values of criticalness indicators are fuzzified using the six step non-uniform fuzzy scale (Table 1). The fuzzy value of each criticalness indicator is received as weighted sum of all values of linguistic variable where the weights are the membership function values of criticalness indicator. The criticalness potential of the task is then calculated as the weighted sum of individual fuzzy evaluation of indicators. The weights of all indicators are set by the experts' evaluation.

Table 1. Fuzzy linguistic terms describing intensity of task criticalness and failureness indicators

Linguistic terms		**Fuzzy number**
Not at all critical	Not at all failing	(0; 0; 0; 0.1)
Usually not critical	Usually not failing	(0; 0.1; 0.2; 0.3)
Rather not critical	Rather not failing	(0.2; 0.3; 0.4; 0.6)
Rather critical	Rather failing	(0.4; 0.6; 0.7; 0.8)
Usually critical	Usually failing	(0.7; 0.8; 0.9; 1)
Always critical	Always failing	(0.9; 0.1; 1; 1)

The final linguistic term expressing the classification of the task criticalness potential is received using suitable method of linguistic approximation into five step non-uniform fuzzy scale (Table 2).

The fuzzy evaluation of the task failureness is based on the expert evaluation of indicators using the six step non-uniform fuzzy scale (Table 1). The failureness potential is calculated as sum of the failureness indicators and using linguistic approximation is again mapped into five step non-uniform fuzzy scale (Table 2).

Table 2. Fuzzy linguistic terms describing the task criticalness and failureness potential

Linguistic terms		**Fuzzy number**
Non-criticalness	Non-failureness	(0; 0; 0.05; 0.15)
Weak criticalness	Weak failureness	(0.05; 0.15; 0.25; 0.35)
Rather criticalness	Rather failureness	(0.25; 0.35; 0.5; 0.6)
Strong criticalness	Strong failureness	(0.5; 0.6; 0.75; 0.85)
Extreme criticalness	Extreme failureness	(0.75; 0.85; 1; 1)

4. Example

The tasks threatness matrix creation is described on the following small-scale project with 7 tasks (Table 3, Figure 2). The critical path of this project consists of the tasks B, C. D, and G.

Table 3. Project example – data from project schedule

Task	Top.	Dur.	Slack	Work	Cost
A	0.5	2	9	4	5
B	0.5	3	0	3	5
C	0.25	8	0	12	15
D	0.75	4	0	8	22
E	0.25	6	6	6	18
F	0.375	1	2	5	6
G	0.625	3	0	6	4

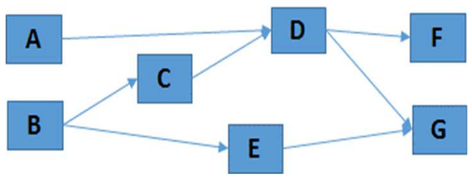


Figure 2. Project example – AON network

The Table 4 shows the initial quantitative values of criticalness factors, which are then transformed into fuzzy criticalness indicators and aggregated into the criticalness potential. The Table 5 shows the experts' evaluation of failureness indicators and their aggregation into the failureness potential.

Table 4. Project example – Task criticalness potential

Task	Criticalness factors					Criticalness potential				Linguistic approximation
	Top.	Dur.	Slack	Work	Cost					
A	0.5	0.143	0	0.111	0.056	0.057	0.141	0.215	0.324	Weak criticalness
B	0.5	0.286	1	0	0.056	0.209	0.275	0.326	0.437	Rather criticalness
C	0	1	1	1	0.611	0.586	0.696	0.725	0.772	Strong criticalness
D	1	0.429	1	0.556	1	0.665	0.785	0.824	0.883	Strong criticalness
E	0	0.714	0.333	0.333	0.778	0.327	0.428	0.51	0.645	Rather criticalness
F	0.25	0	0.778	0.222	0.111	0.111	0.197	0.281	0.396	Weak criticalness
G	0.75	0.286	1	0.333	0	0.294	0.375	0.433	0.557	Rather criticalness
Weights	0.189	0.164	0.129	0.230	0.288					

Table 5. Project example – Task failureness potential

Task	Failureness factors			Failureness potential				Linguistic approximation
	Times	Quality	Costs					
A	Not at all fail.	Rather fail.	Usually fail.	0.367	0.467	0.533	0.633	Rather failureness
B	Rather fail.	Always fail.	Always fail.	0.733	0.867	0.9	0.933	Extremely failureness
C	Not at all fail.	Rather fail.	Usually not fail.	0.133	0.233	0.3	0.4	Weakly failureness
D	Rather not fail.	Rather fail.	Rather not fail.	0.267	0.4	0.5	0.667	Rather failureness
E	Rather not fail.	Usually not fail.	Usually not fail.	0.067	0.167	0.267	0.4	Weakly failureness
F	Always fail.	Usually not fail.	Not at all fail.	0.3	0.367	0.4	0.467	Rather failureness
G	Not at all fail.	Always fail.	Rather not fail.	0.367	0.433	0.467	0.567	Rather failureness

The tasks are now placed into the task threatness matrix (Figure 3). In the red area there is the highly threatening task C requiring great attention. This task is shown as the critical task by MPM method also. In the yellow area there are all other tasks of the project. These tasks have to be controlled to ensure the successful completion of the project regardless of their criticality or non-criticality. The tasks in green area should not significantly influence the project. In this project, there is no task, so in this project all task needs more or less attention, control and care.

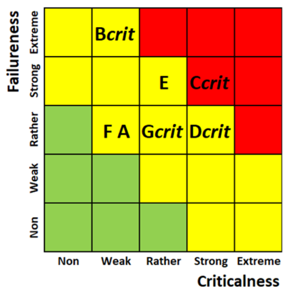


Figure 3. Project example – Task threatness matrix

5. Conclusion

The proposed task threatness matrix was presented to a number of project managers who evaluated it as an interesting, usable and useful tool to support project management. This approach is useful for tasks evaluation with respect to the project schedule, the project management triangle and possibly for other parameters of project tasks failureness or criticalness which have an impact on the project success. In the large projects, the failureness can be evaluated only for selected tasks and remaining tasks can only be arranged according to the criticalness potential in the additional row bellow the task threatness matrix. Important advantage of suggested threatness matrix is that it allows fuzzy assessments of the impact of individual tasks on project completion.

6. Acknowledgements

The research is supported by the Operational Programme Prague – Growth Pole of the Czech Republic - Implementation proof-of-concept activities CULS to promote technology transfer and knowledge into practice, No: CZ.07.1.02/0.0/0.0/17_049/0000815 - KZ10.

References

APM - Association for Project Management, 2008, "Prioritising project risks". Princess Risborough, UK: Association for Project Management.

Aschman, A., 2018, "Why Capital Project Systems Succeed or Fail", IPA, https://www.ipaglobal.com/news/article/why-capital-project-systems-succeed-or-fail-2/.

Betz, J., 2019, "27+ Impressive Project Management Statistics in 2019", https://learn.g2.com/project-management-statistics.

Bowers, J., 1996, "Identifying Critical Activities in Stochastic Resource Constrained Networks", Omega - International Journal of Management Science, Vol. 24, pp. 37–46.

Brožová, H., Bartoška, J. and Šubrt, T., 2014, "Fuzzy Approach to Risk Appetite in Project Management", In Proceedings of the 32st International conference on Mathematical Methods in Economics, pp. 61-66, Olomouc, Czech Republic.

Brožová, H., Bartoška, J., Šubrt, T. and Rydval, J., 2016, "Task Criticalness Potential: A Multiple Criteria Approach to Project Management", Kybernetika, Vol. 52, pp. 558-574.

Brožová, H., Šubrt, T. Rydval, J. and Pavlíčková, P., 2019, "Task Threatness Matrix in the Project Management", In Proceedings of the 32st International conference on Mathematical Methods in Economics, pp. 234-239, České Budějovice, Czech Republic.

Brožová, H., Šubrt, T. Rydval, J. and Pavlíčková, P., 2019, "Fuzzy Threatness Matrices in Project Management", In Proceedings of the 15thInternational Symposium on Operational Research in Slovenia, pp. 581-586. Bled, Slovenia.

Chanas, S., Zielinski, P., 2003, "On the hardness of evaluating criticality of activities in a planar network with duration intervals". Operation Research Letters, Vol. 31, pp. 53–59.

Chen, C. T. and Huang, S. F., 2007, "Applying fuzzy method for measuring criticality in project network", Information Sciences, Vol. 177, pp. 2448–2458.

Cruz, S., García, J. and Herrerías, R., 1999, "Stochastic models alternative to the classical PERT for the treatment of the risk: mesokurtic and of constant variance". Central European Journal of Operations Research, Vol. 7, pp. 159–175.

Gong, D. and Rowings, J. E., 1995, "Calculation of safe float use in risk-analysis-oriented network scheduling", International Journal of Project Management, Vol. 13, pp. 187–194.

Koelmans, R.G., 2004, "Project success and performance evaluation", In: International Platinum Conference 'Platinum Adding Value', pp. 229-236.

de Oliveira Moraes, R., Barbin Laurindo, F.J., 2013, "Performance Evaluation of IT Projects - The Shenhar and Dvir Model", J. Technol. Manag. Innov. Vol. 8, Special Issue ALTEC, pp. 15-24.

Yakhchali, S. H., 2012, "A path enumeration approach for the analysis of critical activities in fuzzy networks", Information Sciences, Vol. 204, pp. 23–35.

Williams, T., 2017, "The Nature of Risk in Complex Projects", Project Management Journal, Vol. 48, pp. 55–66.

Williams, T. M., 1992, "Criticality in stochastic networks", Journal of Operational Research Society, Vol. 43, pp. 353–357.

Local Search Algorithm to Solve a Scheduling Problem in Healthcare Training Center[⋆]

Simon Caillard[1,2], Laure Brisoux Devendeville[1] and Corinne Lucet[1]

[1] Laboratoire MIS (EA 4290), Université de Picardie Jules Verne
33 rue Saint-Leu, 80039 Amiens Cedex 1, France
{simon.caillard, laure.devendeville, corinne.lucet}@u-picardie.fr
[2] Health Simulation Center SimUSanté®
Amiens University Hospital, France
simon.caillard@chu-amiens.fr

Keywords: Scheduling, Local search, Healthcare training, Timetabling.

1 Introduction

SimUSanté, located in Amiens, France is one of the biggest healthcare training center in Europe. All kinds of health actors: professionals, patients, students use this center and can meet and train together by simulating medical acts in various fields of healthcare but also attending regular courses, for a total of more than 500 different formations. The problem faced by SimUSanté is a scheduling problem that consists in planning a set of training sessions respecting a set of time and resource constraints.

Scheduling problems are NP-Complete (Cooper, T.B. and Kingston, J.H. 1995). SimUSanté's problem belongs to this family of problems and specifically to the Curriculum-Based Courses Timetabling Problem (CB-CTT)(Di Gaspero L. *et. al.* 2007) which consists in finding the best weekly assignment for university lectures, available rooms and time periods for a set of classes under a set of hard and soft constraints. However, some features of SimUSanté's problem differ from CB-CTT ones, such as resources types, skills and precedence constraints required for activities, lunch break management, and objective function. Another way would be to consider CB-CTT as a variant of the Resource-Constrained Project Scheduling Problem (RCPSP)(Brucker, P. AND Knust, S. 2001). In this case, we need to add the followings constraints : some activities cannot be planned in parallel and each resource can have more than one type.

We present in this paper a local search algorithm *SimuLS*, based on dedicated neighborhood operators to solve SimUSanté's problem. We generated adequate instances[3] inspired by those used in CB-CTT. We then compared the results obtained by SimuLS with those worked out by the mathematical model implemented in CPLEX and a dedicated greedy algorithm *SimuG* (Caillard S. *et. al.* 2020).

The paper is organized as follows: in section 2, we briefly formalize the scheduling problem encountered by SimUSanté and describe how a solution is evaluated. In section 3 we present our local search algorithm *SimuLS* and give the different operators used in order to explore the search space. Section 4 provides computational results. Finally, section 5 concludes this paper and presents some perspectives.

2 Formalization and evaluation

The problem encountered by SimUSanté is to schedule a set of training sessions S over a determined period T. A training session $s \in S$ is composed by a set of activities A_s.

[⋆] This project is supported by region Hauts-de-France and Health Simulation Center SimUSanté
[3] SimUSanté instances available on: https://mis.u-picardie.fr/Benchmarks-GOC

$A = \bigcup_{s \in S} UA_s$ represents the set of all activities. Activity $a \in A$ has a specific duration and requires different types and quantities of resources. Activities can be linked by precedence constraints. In addition, there is a set of resources R which is composed by employees, rooms and materials. Each resource $r \in R$ is associated to one or more types of resources. For example a room can have both meeting room and classroom types.

Solution *Sol* is a set of triplets (a, t_a, R_a) where $a \in A$ is an activity, $t_a \in T$ the starting time slot of a, and $R_a \subseteq R$ the set of avalaible resources assigned to a, from t_a and for its total duration $duration_a$. The set of scheduled activities is denoted $SA = \{a | (a, t_a, R_a) \in Sol\}$, with $(SA \subseteq A)$. The set of unscheduled activities is denoted $UA = A \setminus SA$. For session $s \in S$, $Sol_s \subseteq Sol$, represents the set of triplets of the solution related to s, with $Sol_s = \{(a, t_a, R_a) \in Sol | a \in A_s\}$. $SA_s = SA \cap A_s$, is the set of scheduled activities of s, and $UA_s = A_s \setminus SA_s$, the set of unscheduled activities of s.

For a given session $s \in S$, if at least one activity has been scheduled $(SA_s \neq \emptyset)$, start date $t_{start_s} = \min\{t_a, a \in SA_s\}$, and end date $t_{end_s} = \max\{t_a + duration_a, a \in SA_s\}$ allow to compute the corresponding makespan $mk_s = t_{end_s} - t_{start_s}$ of session s. If no activity has been scheduled $(SA = \emptyset)$, then $mk_s = 0$.

The evaluation of *Sol*, denoted $Makespan(Sol)$, is the sum of the makespans of all sessions, plus the amount of unplanned activities, multiplied by penalty α (see equation 1). The objective is to find a valid solution with a minimum $Makespan$.

$$Makespan(Sol) = \sum_{s \in S} mk_s + |UA| \times \alpha \tag{1}$$

3 Local search algorithm: SimuLS

SimuLS is a local search algorithm that explores the solution space by applying neighborhood operators, starting from a solution provided by a greedy algorithm, *SimuG* (Caillard S. *et. al.* 2020). For a maximum preset *limitCounter* iterations, *SimuLS* relies on *saturator* operator to plan unscheduled activities and when it is not possible, it uses several operators: *intra*, *extra* and $extra^+$. Each of these operators checks possible movements and applies one. If the best solution ever met is not improved after a preset $noImprov$ iterations, a part of the solution is destroyed by the *destructor* operator, in order to escape from a local minimum.

A *movement* is caracterized by a couple $< (a, t_a, R_a) \; ; \; \Upsilon >$. (a, t_a, R_a) represents a triplet that will be added to the current solution, with $a \in UA$, an unscheduled activity, $t_a \in T$, a time slot from which a could be started, and R_a, the set of resources assigned to a , that exactly matches its resources requirement. In order to plan a, we need to remove a set Υ of triplets from the solution. $\Upsilon = \{(b_1, t_{b_1}, R_{b_1}), \ldots, (b_n, t_{b_n}, R_{b_n})\}$, $n \in \{1, \ldots |Sol|\}$. The set of resources R_a can be composed by resources directly available over T, plus thoses released by canceling all activities of Υ. A *movement* respects all operational rules and resources constraints.

The choice of a movement by an operator relies on criteria such as makespan mk_s of impacted sessions, global makespan $Makespan$, the number of canceled activities, etc. The different operators present in *SimuLS* are:

$saturator_s$: This operator tends to place an unscheduled activity $a \in UA_s$ without changing the current solution. It builds a set of movements M:$\{< (a, t_a^1, R_a); \emptyset >, \ldots, < (a, t_a^k, R_a); \emptyset >\}$ so that for each movement $< (a, t_a^i, R_a) \; ; \; \emptyset > \in M$, with $i \in [1; k]$, $[t_a^i; t_a^i + duration_a[\cap [t_b; t_b + duration_b[= \emptyset \quad \forall (b, t_b, R_b) \in Sol_s$.

$intra_s$: This operator removes one or more scheduled activities from session s in order to plan an unscheduled activity $a \in UA_s$. It builds a set of movements M:$\{< (a, t_a^1, R_a); \Upsilon >, \ldots, < (a, t_a^k, R_a); \Upsilon >\}$ so that for each movement $< (a, t_a^i, R_a); \Upsilon > \; \in M$, with $i \in [1; k]$ and $\Upsilon \subseteq Sol_s$, the following properties are verified:

- $[t_a^i; t_a^i + duration_a[\cap [t_b; t_b + duration_b[\neq \emptyset, \forall (b, t_b, R_b) \in \Upsilon$
- $[t_a^i; t_a^i + duration_a[\cap [t_b; t_b + duration_b[= \emptyset, \forall (b, t_b, R_b) \in \{Sol_s \setminus \Upsilon\}$

$extra_s$**:** This operator removes one or more scheduled activities from a randomly selected session $s' \neq s$, in order to plan an unscheduled activity $a \in UA_s$. It builds a set of movements M:$\{< (a, t_a^1, R_a); \Upsilon >, \ldots, < (a, t_a^k, R_a); \Upsilon >\}$ so that for each movement $< (a, t_a^i, R_a); \Upsilon > \in M$, with $i \in [1; k]$ and $\Upsilon \subseteq Sol_{s'}$, the following properties are verified:

- $[t_a^i; t_a^i + duration_a[\cap [t_b; t_b + duration_b[\neq \emptyset, \forall (b, t_b, R_b) \in \Upsilon$
- $[t_a^i; t_a^i + duration_a[\cap [t_b; t_b + duration_b[= \emptyset, \forall (b, t_b, R_b) \in Sol_s$

$extra_s^+$**:** This operator is an extension of $extra_s$. The canceled activities can belong to a set of sessions $\{s'_1, \ldots, s'_k\} \subseteq S$. For activity $a \in UA_s$, it builds a set of movements M:$\{< (a, t_a^1, R_a); \Upsilon >, \ldots, < (a, t_a^k, R_a); \Upsilon >\}$ so that for each movement $< (a, t_a^i, R_a); \Upsilon > \in M$, with $i \in [1; k]$ and $\Upsilon \subseteq Sol$, the properties below are verified :

- $[t_a^i; t_a^i + duration_a[\cap [t_b; t_b + duration_b[\neq \emptyset, \forall (b, t_b, R_b) \in \Upsilon$
- $[t_a^i; t_a^i + duration_a[\cap [t_b; t_b + duration_b[= \emptyset, \forall (b, t_b, R_b) \in \{Sol_s \setminus \Upsilon\}$

$destructor$**:** This operator destroys a part of current solution Sol. It builds and applies a set of movements M:$\{< \emptyset \; ; \; \{(a_1, t_{a_1}, R_{a_1}\} >, \ldots, < \emptyset \; ; \; \{(a_k, t_{a_k}, R_{a_k}\} >$ so that $\forall i \in [1; k]$, $(a_i, t_{a_i}, R_{a_i}) \in Sol$ represents the triplet that will be removed from the Sol.

Algorithm 1 : SimuLS

```
Input: $Sol$ (the current solution), $S$ (set of sessions), $\forall s \in S, UA_s$ (set of unscheduled activities
  for session s), $UA = \bigcup_{s \in S} UA_s$ (the set of unscheduled activities), $noImprov$, $limitCounter$
  $noBest \leftarrow 0$
  $counter \leftarrow 0$
  $Sol \leftarrow saturator(UA)$
  $bestSol \leftarrow Sol$
  while $counter < limitCounter$ do
    if ($noBest = noImprov$) then
      $Sol \leftarrow destructor(Sol)$
      $noBest \leftarrow 0$
    end if
    if $UA \neq \emptyset$ then
      $a \leftarrow random(UA)$
      $s \leftarrow (s/a \in UA_s)$
      $Sol \leftarrow selectOperator(\{intra, extra, extra^+\}, s, a)$
    end if
    $Sol \leftarrow saturator(UA)$
    if $Makespan(Sol) < Makespan(bestSol)$ then
      $noBest \leftarrow 0$
      $bestSol \leftarrow Sol$
    else
      $noBest \leftarrow noBest + 1$
    end if
    $counter \leftarrow counter + 1$
  end while
```

In order to choose which operator to apply between $intra$, $extra$, $extra^+$, $SimuLS$ uses the $SelectOperator$ function that uses two specific counters c^s_{extra} and $c^a_{extra^+}$. The first one, c^s_{extra}, counts the number of times where $intra$ have been consecutively applied to session s. The second one counts how many times activity a remained consecutively unscheduled. By default, operator $intra$ is applied, except whenever one of these counters reaches a preset limit, $selectOperator$ then activates the operator that corresponds to the counter. In case of equality between the two counters, $extra^+$ is always used first.

4 Experimental study

A mathematical model has been implemented under CPLEX. It provides optimal results for small instances with a running time of two hours or more. Table 4 presents the comparison between CPLEX, *SimuG* and *SimuLS* on SimUSanté instances. Penalty α is set to $|T|$. *SimuLS* was implemented in Java, on an Intel i7 7500U processor. The time used to find solutions is always less than 1 second for the greedy algorithm *SimuG* and less than 1 minute for *SimuLS*. The numbers in parentheses after some if the results, represent the amount of unscheduled activities.

Table 1. Results for Brazil1 and Italy1 instances

Instance Brazil1				Instance Italy1			
Instance	**cplex**	**SimuG**	**SimuLS**	**Instance**	**cplex**	**SimuG**	**SimuLS**
$D_0T_0C_0A_0$	81	86	83	$D_0T_0C_0A_0$	101	105	102
$D_0T_0C_1A_0$	81	87	82	$D_0T_0C_1A_0$	101	104	101
$D_0T_1C_0A_1$	94	232 (4)	110	$D_0T_1C_0A_1$	107	150 (1)	116
$D_0T_1C_1A_1$	94	232 (4)	108	$D_0T_1C_1A_1$	107	187 (2)	115
$D_1T_0C_0A_0$	81	89	85	$D_1T_0C_0A_0$	101	104	104
$D_1T_0C_1A_0$	81	94	90	$D_1T_0C_1A_0$	101	104	104
$D_1T_1C_0A_1$	96	161 (2)	107	$D_1T_1C_0A_1$	107	150 (2)	114
$D_1T_1C_1A_1$	96	166 (2)	110	$D_1T_1C_1A_1$	107	180 (3)	115

Columns *cplex*, $SimUG$ and $SimULS$ represent respectively the optimums, the results of greedy algorithm and those of the local search algorithm. By the nature of a greedy algorithm, SimUG cannot scheduled all activities (see instances $D_0T_1C_0A_1$, $D_0T_1C_1A_1$, $D_1T_1C_0A_1$, $D_1T_1C_1A_1$). In this case, a penalty α is applied, and the corresponding score is rising up to 246% from optimality. In contrast, *Cplex* and $SimULG$ always schedule all activities. $SimULG$ reaches optimality for Italy$-D_0T_0C_1A_0$ instance, and always improves the results obtained by the greedy algorithm. The gap with optimality is less than 18%.

5 Conclusion

In this paper we have briefly introduced SimUSanté's problem and proposed a local search algorithm *SimuLS* to solve it. *SimuLS* is based on five neighborhood operators dedicated to SimUSanté's problem. Four of them allow to schedule activities but only one without modify solution. The last operator destroys the solution in order to escape from a local minimum. *SimuLS* is experimented on instances from CB-CTT, adapted to the SimUSanté's problem. The results obtained are compared to the optimal solutions provided by CPLEX. Contrary to *SimuG*, all activities are scheduled by *SimuLS*. It is a first step towards building an efficient metaheuristic to solve SimUSanté's problem.

References

Brucker, P. AND Knust, S., 2001, “Resource-Constrained Project Scheduling and Timetabling", *Springer*, Burke E., Erben W. (eds) Practice and Theory of Automated Timetabling III.

Caillard S., Brisoux-Devendeville L., and Lucet C., 2020, “A Planning Problem with Resource Constraints in Health Simulation Center", *Springer*, Le Thi H., Le H., Pham Dinh T. (eds) Optimization of Complex Systems: Theory, Models, Algorithms and Applications.

Cooper, T.B. and Kingston, J.H., 1995, “The complexity of timetable construction problems", *Springer*, Burke E., Ross P. (eds) Practice and Theory of Automated Timetabling.

Di Gaspero, L. and McCollum, B. and Schaerf, A., 2007, “Curriculum-based CTT - Technical Report", *The Second Int. Timetabling Competition*

Schaerf A., 1999, “A Survey of Automated Timetabling", *Artificial Intelligence Review*, Vol. 13, pp. 87-127.

Computing lower bounds for the cumulative scheduling problem

Jacques Carlier[1], Antoine Jouglet[1], and Abderrahim Sahli[2]

[1] Heudiasyc UMR CNRS 7253, Sorbonne Universités, Université de Technologie de Compiègne, Compiègne, France
jacques.carlier@hds.utc.fr, antoine.jouglet@hds.utc.fr
[2] LIGM UMR CNRS 8049, Université Paris-Est-Marne-La-Vallée, Paris, France
abderrahim.sahli@esiee.fr

Keywords: Cumulative Scheduling Problem, Energetic Reasoning, Lower Bounds.

1 Introduction

In this paper, we consider the Cumulative Scheduling Problem (Carlier 1987). An instance of this problem is composed of a set of n tasks $J = \{1, \ldots, n\}$. These tasks have to be scheduled without preemption by a resource of a given capacity C. Each task $i \in J$ cannot be scheduled before its release date r_i, has a duration p_i, is characterized by a tail q_i and needs c_i units of the resource to be processed. A schedule consists in assigning a starting time $s_i \geq r_i$ to each task i in such a way that the capacity of the resource is never exceeded : $\forall t, \sum_{i \in \{j \in J | s_j \leq t < s_j + p_j\}} c_i \leq C$.

In this paper, we propose some algorithms to compute lower bounds for the optimisation version of the cumulative scheduling problem (CuSP Optimisation). In CuSP Optimisation we have to find a schedule which minimizes the makespan $C_{max} = \max_{i \in J} \{s_i + p_i + q_i\}$. Let C^*_{max} be the optimal value of a given instance of CuSP Optimisation. The special case of CuSP Optimisation where $\forall i \in J, c_i = 1$ corresponds to the m-parallel machine scheduling problem $Pm|r_i, q_i|C_{max}$. Several lower bounds of C^*_{max} have been described for $Pm|r_i, q_i|C_{max}$ (Horn 1974, Labetoulle *et. al.* 1984, Carlier and Pinson 1998, Haouari 2003). In (Carlier, Pinson, Sahli and Jouglet submitted), we provided caracterizations of some lower bounds for CuSP Optimisation to analyse their structural differences. It leaded to the elaboration of new algorithms for Energetic Reasoning (ER) (Baptiste *et. al.* 2001) and we discussed the transformation of the destructive energetic bound (the ER based checker for CuSP Decision(C_{max})) into constructive energetic lower bounds of C^*_{max}. In the remainder let $LB_0(J) = \max_{i \in J} \{r_i + p_i + q_i\}$ be a trivial lower bound which can be easily computed in $O(n)$ time. The first constructive energetic lower bound, named $LB_2^{ER}(J)$, relies on particular tasks for which there is at least an interval of the time horizon in which they are necessarily scheduled because of their release dates and tails. Such tasks are called *crossing tasks.* The concept of crossing tasks is related to *core times.* The second constructive energetic lower bound, named $LB_3^{ER}(J)$ relies on ER. Both $LB_2^{ER}(J)$ and $LB_3^{ER}(J)$ were theoretically characterized in (Carlier J., Pinson E., Sahli A. and Jouglet A. submitted).

Section 2 is devoted to the introduction of the energetic approach initially proposed for the decision version of CuSP and its reformulation in the context of CuSP Optimisation. Section 3 explains the notion of crossing tasks which is the main concept used in $LB_2^{ER}(J)$ and which has also an important role in $LB_3^{ER}(J)$. We then describe an algorithm in $O(n \log n)$ time for $LB_2^{ER}(J)$. In Section 4, we describe an algorithm in $O(n^2)$ time and an algorithm in $O(\alpha(n) n \log n \log(\max_{i \in J} p_i))$ for $LB_3^{ER}(J)$, where $\alpha(n)$ is the inverse function of Ackermann.

2 The energetic reasoning in CuSP Optimisation

A lot of works of the literature considers the decision version of the CuSP in which all tasks have to be completed before a given value of C_{max}. Being given a value C_{max}, we denote this problem by CuSP Decision(C_{max}). In CuSP Decision(C_{max}), tails q_i are replaced by deadlines $d_i(C_{max}) = C_{max} - q_i$. Therefore, each task i has to processed in interval $[r_i, d_i(C_{max})]$. It can lead to unfeasible instances. The *Energetic Reasoning* (ER) (Erschler and Lopez 1990) (Erschler 1991) (Baptiste *et. al.* 1999) is a very well known technique to solve CuSP Decision(C_{max}) allowing feasibility tests and time-bound adjustments. Given a time interval $[\alpha, \delta]$, ER is based on the computation of the minimal part, named *energy*, of the tasks that must be processed in any feasible schedule between times α and δ. The minimal energy required by task i over $[\alpha, \delta]$ is obtained from positions of i that overlap as less as possible with the interval. The difference between the length of a given interval multiplied by C and the sum of the tasks energies is called the *slack* of the interval. If we can find an interval with a negative slack, then the instance is unfeasible. While the slack has to be non-negative on any interval, it is sufficient to test at most $O(n^2)$ particular intervals (Baptiste *et. al.* 1999). It permitted to exhibit a checker which runs in $O(n^2)$ time (Baptiste *et. al.* 2001). Derrien and Petit (Derrien and Petit 2014) have later reduced the number of intervals which has to be considered. Ouellet and Quimper (Ouellet and Quimper 2018) described an $O(n \log^2 n)$ algorithm. Recently, we provided a $O(\alpha(n) n \log n)$ algorithm for the checker (Carlier, Sahli, Jouglet and Pinson submitted), where $\alpha(n)$ is the inverse function of Ackermann. We also provided an $O(n^2)$ algorithm for time-bound adjustments (Carlier *et. al.* 2020).

In the context of CuSP Optimisation, we use ER in algorithms in which the value of C_{max} dynamically changes during the execution. Thus, the deadline $d_i(C_{max}) = C_{max} - q_i$ of task i is also modified. Actually, it is simpler to manipulate directly tails q_i which are constant. Therefore, we propose a reformulation of ER with tails which manipulates directly C_{max}. Instead of considering intervals, we now equivalently manipulate triplets $(\alpha, \gamma, C_{max})$ which corresponds to intervals $[\alpha, \delta = C_{max} - \gamma]$ in CuSP Decision(C_{max}).

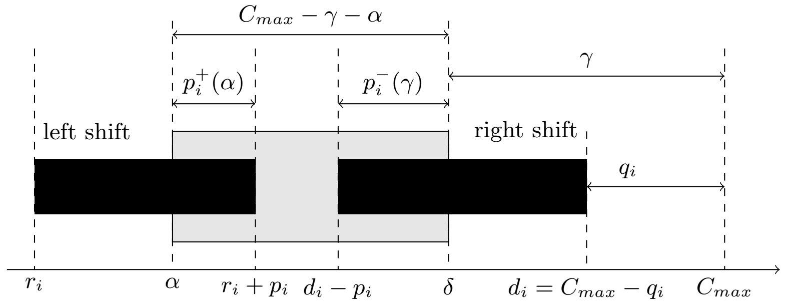


Fig. 1. Intersection energy.

For given values of C_{max}, $\alpha \in \{0, \ldots, C_{max}\}$ and $\gamma \in \{0, \ldots, C_{max} - \alpha\}$, we define:

- $\delta = C_{max} - \gamma$
- $\forall i \in J$ $p_i^+(\alpha) = \min(\max(0, r_i + p_i - \alpha), p_i)$, $p_i^-(\gamma) = \min(\max(0, q_i + p_i - \gamma), p_i)$ and $W_i(C_{max}, \alpha, \gamma) = c_i \min(p_i^+(\alpha), p_i^-(\gamma), C_{max} - \alpha - \gamma)$ is the energy of task i.
- The total required energy by tasks is $W(C_{max}, \alpha, \gamma) = \sum_i W_i(C_{max}, \alpha, \gamma)$. The slack, which is the difference between the maximum energy available over $[\alpha, C_{max} - \gamma]$ and the total required energy by tasks, is $S(C_{max}, \alpha, \gamma) = C(C_{max} - \gamma - \alpha) - W(C_{max}, \alpha, \gamma)$.

There exists a schedule with makespan C_{max} only if $\forall(\alpha, \gamma)$ with $\alpha \in \{0, \ldots, C_{max} - 1\}$ and $\gamma \in \{0, \ldots, C_{max} - \alpha - 1\}$, we have $S(C_{max}, \alpha, \gamma) \geq 0$. In fact, by adapting results of

(Baptiste *et. al.* 1999, Derrien and Petit 2014), there are only $O(n^2)$ couples (α, γ) values to consider for a given value of C_{max}.

3 $LB_2^{ER}(J)$: a constructive lower bound based on the notion of crossing-tasks

Given a makespan C_{max}, a task i is called a C_{max}-*crossing-task* if and only if there exists an interval of time in which task i is necessarily scheduled, *i.e.* if $C_{max} - q_i - p_i < r_i + p_i$. If a task i is necessarily scheduled during interval $[t, t+1)$, i is called a (C_{max}, t)-crossing-task ($t \in \{C_{max} - q_i - p_i, \dots, r_i + p_i - 1\}$).

We provide an algorithm to compute the lower bound $LB_2^{ER}(J)$ which corresponds to the smallest value of $C_{max} \geq LB_0(J)$ for which for any time $t \in \{0, \dots, C_{max}\}$, the sum of capacities required by (C_{max}, t)-crossing tasks in J is lower than or equal to C. Let $\chi(C_{max})$ be the set of C_{max}-crossing-tasks and let $\chi(C_{max}, t)$ be the set of (C_{max}, t)-crossing tasks. Thus, note that $LB_2^{ER}(J)$ corresponds to the smallest value $C_{max} \geq \max_{i \in J} (r_i + p_i + q_i)$ for which for any time $t \in \{0, \dots, C_{max}\}$ we have $\sum_{i \in \chi(C_{max}, t)} c_i \leq C$.

Note that a C_{max}-crossing-task i becomes crossing at time $C_{max} - q_i - p_i$ while it is not crossing anymore from time $r_i + p_i$. Let T be the list of dates in $\{r_i + p_i | i \in J\}$. Our algorithm iterates over the different $t \in T$ in the non-increasing order. At each iteration of the algorithm, we maintain an AVL-tree CT in such a way that it contains all (C_{max}, t)-crossing-task. We also maintain a variable C_{CT} in such a way it corresponds to the sum of the capacities of the crossing-tasks over $[t-1, t)$. At each iteration, we will verify that $C_{CT}(C_{max}, t-1) \leq C$. It allows to ensure that at the end of the algorithm C_{max} has been adjusted to the smallest value $C_{max} \geq LB_0(J)$ for which for any time $t \in \{0, \dots C_{max}\}$ we have $C_{CT}(C_{max}, t) \leq C$. To maintain these properties, we use a forward linked list allowing the tasks i which are not crossing at time $t - 1 \geq C_{max} - q_i - p_i$ to be known : these tasks have to be removed from CT. We also use another forward list allows the tasks i which are crossing at time $t - 1 \geq r_i + p_i$ to be known and which have therefore to be inserted in CT. When $C_{CT} > C$, we adjust C_{max} in such a way that $C_{CT} \leq C$. This algorithm runs in $O(n \log n)$ time. It is analogous to the sweep algorithm of (Beldiceanu and Carlsson 2002) verifying the cumulative constraint. It uses additional data structures for adjusting C_{max}.

4 LB_3^{ER}: a constructive lower bound based on the energies

We also provide two algorithms to compute $LB_3^{ER}(J)$ which corresponds to the smallest value of $C_{max} \geq LB_2^{ER}(J)$ for which for any $\forall(\alpha, \gamma)$ with $\alpha \in \{0, \dots, C_{max} - 1\}$ and $\gamma \in \{0, \dots, C_{max} - \alpha - 1\}$, we have $S(C_{max}, \alpha, \gamma) \geq 0$.

Our first algorithm uses twice an adjustment procedure of C_{max}. Indeed, the r_i and q_i play a symmetrical role. Therefore, for each given $\alpha \in \{r_i, r_i + p_i, C_{max} - q_i - p_i | i \in J\}$, we check the couples (α, γ) with $\gamma \in \{C_{max} - r_i - p_i, q_i + p_i, q_i, \alpha + q_i - r_i | i \in J\}$ such that $\gamma < C_{max} - \alpha$. Next, we build the instance in which the r_i and the q_i values are interchanged and we apply the same procedure. It ensures that all relevant couples (α, γ) identified by (Baptiste *et. al.* 1999, Derrien and Petit 2014) are considered during the algorithm. The adjustment procedure iteratively considers the different pertinent values of α in an outer loop while. For each value of α it then considers the pertinent values of γ in decreasing order, allowing the right bound of the associated interval to increase iteratively while we maintain the value of the required energies of the task in this interval. Each time it is detected that the slack is negative on the current interval, the value of C_{max} is adjusted and our data structures are updated to continue the consideration of the other intervals. The whole algorithm runs in $O(n^2)$ time and uses only simple data structures (arrays and forward linked lists).

Our second algorithm relies on the direct use of our checker described in (Carlier J., Sahli A., Jouglet A. and Pinson E. submitted) to do a dichotomic search on $LB_3^{ER}(J)$. The complexity is theoretically attractive : $O(\alpha(n)n \log n \log(\max_{i\in J} p_i))$, where $\alpha(n)$ is the inverse function of Ackermann.

A drawback is that we don't compute the energetic balance of each classical interval which should be useful for computing adjustments. Moreover, the checker uses very complex data structures which makes it very hard to implement.

Acknowledgements

These works are partially financed by the project 2018-0062H of the *Gaspard Monge Program for Optimization, operations research, and their interactions with data science.*

References

Baptiste P., Le Pape C. and Nuijten W., 1999, "Satisfiability tests and time-bound adjustments for cumulative scheduling problems", *Annals of Operations Research*, 92, pp. 305-333.

Baptiste P., Le Pape C., Nuijten W., 2001, "Constraint-based scheduling : applying constraint programming to scheduling problems", *International Series in Operations Research and Management Science*, vol 39, Kluwer.

Beldiceanu N. and Carlsson M., 2002, "A new multi-resource cumulative constraint with negative heights", *CP*, 2470, pp. 63-79.

Carlier J., 1987, "Scheduling jobs withe release dates and tails on identical machines to minimize the makespan", *European Journal of Operational Research*, 29, pp. 298-306.

Carlier J., Pinson E., 1998, "Jackson's pseudo preemptive schedule for the $Pm/r_i, p_i/C_{max}$ scheduling problem", *Annals of Operations Research*, 83(0), pp. 41-58 (1998)

Carlier J., Pinson E., Sahli A. and Jouglet A., to appear, "An $O(n^2)$ algorithm for time-bound adjustments for the cumulative scheduling problem" , *European Journal of Operational Research.*

Carlier J., Pinson E., Sahli A. and Jouglet A., submitted, "Comparison of three classical lower bounds for the cumulative scheduling problem".

Carlier J., Sahli A., Jouglet A. and Pinson E., submitted, "A nearly $o(n \log n)$ checker algorithm for the cumulative scheduling problem".

Derrien A., Petit T., 2014, "A new characterization of relevant intervals for energetic reasoning", In *International conference on principles and practice of constraint programming*, pp. 289-297.

Erschler J., Lopez P., 1990, "Energy-based approach for task scheduling under time and resources constraints", In *Proceedings of the 2^{nd} international workshop on project management and scheduling*, pp. 115-121.

Erschler J., Lopez P., Thuriot, 1991, "Raisonnement temporel sous contraintes de ressource et problèmes d'ordonnancement", *Revue d'Intelligence Artificielle*, 5(3), pp. 7-32.

Haouari M., Gharbi A., 2003, "An improved max-flow-based lower bound for minimizing maximum lateness on identical parallel machines", *Operations Research Letters*, 31(1), pp. 49-52.

Horn W., 1974, "Some simple scheduling algorithms", *Naval Research Logistics Quaterly*, 21, pp. 177-185.

Labetoulle J., Lawler E., Lenstra J., Rinnooy Kan ., 1984, "Preemptive scheduling of uniform machines subject to release dates", *Progress in combinatorial optimization* (Waterloo Ont, 1982), Academic Press, pp. 245-261.

Ouellet .Y, Quimper C.G., 2018, "A $O(n \log^2 n)$ checker and $O(n^2 \log n)$ filtering algorithm for the energetic reasoning", In, *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 477-494.

Ultimate Instance Reduction for the Routing Open Shop

Ilya Chernykh[1,2,3]

[1] Sobolev Institute of Mathematics, Russia
idchern@math.nsc.ru
[2] Novosibirsk State University, Russia
[3] Novosibirsk State Technical University, Russia

Keywords: Routing open shop, instance reduction, lower bound, optima localization.

1 Introduction

In the routing open shop problem a fleet of mobile machines has to process a set of immovable jobs located at the nodes of some transportation network, described by an undirected edge-weighted graph $G = \langle V; E\rangle$, where each node contains at least one job, and weight $\mathrm{dist}(u, v)$ represents travel times between nodes u and v. Each machine M_i has to perform an operation O_{ji} on each job J_j in open shop environment, the processing times p_{ji} are given. All the machines start from the same node v_0 referred to as *the depot* and have to return to the depot after processing all the job. No restriction on the machines traveling are in order: any number of machines can travel over the same edge of the network simultaneously, machines are allowed to visit each node multiple times. However, machine has to reach a node prior to be able to process jobs located there. The goal is to minimize the makespan $R_{\max}$, *i.e.* the completion time of the last machine's activity (either traveling back to the depot or performing an operation on a job located at the depot). The problem is clearly a generalization of the metric traveling salesman problem and therefore is NP-hard in strong sense even for single machine. On the other hand, it generalizes the classical open shop problem, which is well-known to be NP-hard for the case of three and more machines, and is polynomially solvable for the two-machine case (Gonzalez T.F. and Sahni S. 1976). Surprisingly, the routing open shop is NP-hard even in the two-machine case on the transportation network consisting of at least two nodes (including the depot) (Averbakh I. *et. al.* 2006). We use notation $ROm||R_{\max}$ for the routing open shop with m machines. Optional notation $G = X$ in the second field is used in case we want to specify the structure of the transportation network, with X being the name of the structure (*e.g.* K_p or *tree*). A set of instances of the $ROm|G = X|R_{\max}$ problem is denoted by $\mathcal{I}_m^X$ (or $\mathcal{I}_m$ for a general case of unspecified X).

The routing open shop problem was introduced by Averbakh I. *et. al.* (2005). In our research we utilize the *standard lower bound* on the optimal makespan from the same paper:

$$\bar{R} = \max\left\{\ell_{\max} + T^*, \max_{v\in V}\left(d_{\max}(v) + 2\mathrm{dist}(v_0, v)\right)\right\}. \tag{1}$$

Here $\ell_{\max} = \max\limits_i \sum\limits_{j=1}^{n} p_{ji}$ is the maximum machine load, $d_{\max}(v) = \max\limits_{j\in\mathcal{J}(v)}\left(\sum\limits_{i=1}^{m} p_{ji}\right)$ is the maximum length of job from node v, with $\mathcal{J}(v)$ being the set of jobs located at v, while T^* is the TSP optimum on G. The problem under research is so-called *optima localization* and can be described as follows: how much (by what factor) can optimal makespan differ from the standard lower bound $\bar{R}$ for a given class of instances $\mathcal{K}$? More precisely, for some class $\mathcal{K}$ we want to find

$$\alpha(\mathcal{K}) = \sup_{I\in\mathcal{K}} \alpha(I) = \sup_{I\in\mathcal{K}} \frac{R^*_{\max}(I)}{\bar{R}(I)}.$$

Here $R^*_{\max}(I)$ and $\bar{R}(I)$ denote optimal makespan and the value of $\bar{R}$ for I, respectively, and $\alpha(I)$ is referred to as the *abnormality* of instance I.

It is known that for the classical two-machine open shop (which can be denoted as $RO2|G = K_1|R_{\max}$ for consistency) optimal makespan always coincides with the standard lower bound, therefore $\alpha\left(\mathcal{I}_2^{K_1}\right) = 1$ (Gonzalez T.F. and Sahni S. 1976). It is not the case for the three-machine problem, where optimal makespan can reach as much as $\frac{4}{3}\bar{R}$ (Sevastyanov S.V. and Tchernykh I.D. 1998). The value of $\alpha\left(\mathcal{I}_4^{K_1}\right)$ is still an open question, however we have no evidence that it is greater than $\frac{4}{3}$. Needless to say, that similar research for the routing open shop is probably harder even for $m = 2$, because the value $\alpha\left(\mathcal{I}_m^{K_p}\right)$ might depend both on m and p. However, it was recently established that $\alpha\left(\mathcal{I}_3^{K_2}\right) = \frac{4}{3}$ (Chernykh I. and Krivonogova O. 2020).

Current research for two machines up to the moment is as follows:

1. $\alpha\left(\mathcal{I}_2^{K_2}\right) = \frac{6}{5}$ (Averbakh I. *et. al.* 2005);
2. $\alpha\left(\mathcal{I}_2^{K_3}\right) = \frac{6}{5}$ (Chernykh I. and Lgotina E. 2016);
3. $\alpha\left(\mathcal{I}_2^{tree}\right) = \frac{6}{5}$ (Krivonogova O. and Chernykh I. 2019).

This paper addresses a natural question: how to stop this infinite series of incremental results and still reach an ultimate goal of discovering the general value $\alpha\left(\mathcal{I}_2\right)$.

2 Instance transformations

The research of some extremal (with respect to the standard lower bound) properties of the set of instances (such as optima localization) is often based on some instance transformation procedures. Suppose we have some transformation which obtains instance $\tilde{I}$ from I. Such a procedure is called *reversible* if any feasible schedule for $\tilde{I}$ can be treated as a feasible schedule for I. Reversibility means that $R^*_{\max}(\tilde{I}) \geqslant R^*_{\max}(I)$. The transformation $I \to \tilde{I}$ is referred to as *valid* if it preserves the standard lower bound: $\bar{R}(\tilde{I}) = \bar{R}(I)$. Obviously for any valid and reversible transformation $I \to \tilde{I}$ we have $\alpha(\tilde{I}) \geqslant \alpha(I)$. That observation serves as a foundation for the following approach to investigate the abnormality $\alpha(\mathcal{I})$ for some set of instances:

1. Describe a valid reversible transformation on $\mathcal{I}$ which *simplifies* the instance (*i.e.* reduces number of jobs to some constant, or simplifies the structure of the transportation network).
2. Describe the image $\tilde{\mathcal{I}}$ of $\mathcal{I}$ under that transformation. Find $\alpha(\tilde{\mathcal{I}})$.

There is a well-known transformation which reduces the number of jobs, referred to as *job aggregation* or *job grouping.* The idea is to combine a set of jobs into a single one adding up the processing times independently for each machine. Such a procedure was used, *e.g.*, in (Sevastyanov S.V. and Tchernykh I.D. 1998) for the classic open shop problem, and in (Chernykh I. and Lgotina E. 2016, Krivonogova O. and Chernykh I. 2019) for the two-machine routing open shop. While the procedure is clearly reversible, its validity has to be maintained explicitly. For example, it is possible to perform valid job aggregation for any instance of $Om||C_{\max}$ so that the resulting instance would contain at most $2m - 1$ jobs (Sevastyanov S.V. and Tchernykh I.D. 1998). As for $RO2||R_{\max}$, one can aggregate jobs in such a valid manner that every node (except for at most one) has a single job, and the "exceptional" one (if any) contains at most 3 jobs (Chernykh I. and Lgotina E. 2016). Such an exceptional node v is referred to as *overloaded*:

$$\varDelta(v) = \sum_{j \in \mathcal{J}(v)} \sum_i p_{ji} > \bar{R} - 2\mathrm{dist}(v_0, v).$$

However, it would be of the most interest to describe some valid reversible transformation to simplify the structure of G. An example of such a reduction is so-called *terminal edge contraction*, which can be described as follows. Suppose G contains a terminal node $v \neq v_0$ with a single job J_j in $\mathcal{J}(v)$. Let u be the node adjacent to v, and $\tau = \mathrm{dist}(u, v)$. We translate the job J_j to the node u, increase its operations processing times by 2τ, and eliminate the obsolete node v. Such a transformation is reversible, as one can treat the processing of a new operation O_{ji} as a concatenation of traveling of M_i from u to v, processing of the initial operation and traveling back to u. It is proved in (Chernykh I. and Lgotina E. 2019) that for any instance $I \in \mathcal{I}_2$ one can perform a valid transformation $I \to \tilde{I}$ such that the transportation network in $\tilde{I}$ contains at most two terminal nodes. This helps to efficiently reduce any tree to a chain. On the other hand a graph might have a complex structure even without terminal edges. Below we describe a new approach to the instance reduction which allows to significantly simplify the structure of a transportation network preserving the standard lower bound $\bar{R}$.

Consider an instance $I \in \mathcal{I}_2$. Let $\varDelta = \sum\limits_{i,j} p_{ji}$ be the *total load* if I. Note that (1) implies

$$\varDelta \leqslant 2\ell_{\max} \leqslant 2(\bar{R} - T^*). \tag{2}$$

Let cycle σ be an optimal solution of the underlying TSP. Any edge $e \notin \sigma$ is referred to as *chord*. A chord e is referred to as *critical* if removing it from G increases the standard lower bound $\bar{R}$.

Lemma 1. *Any instance $I \in \mathcal{I}_2$ contains at most one critical chord, which is incident to the depot v_0.*

Proof. Note that the definition (1) does not depend on any distance between two non-depot nodes, therefore a chord may be critical only if it is incident to v_0. Suppose a chord $[v_0, v]$ is critical and τ is the new distance between v_0 and v after removing e from G. Then $\bar{R} < 2\tau + d_{\max}(v) \leqslant T^* + d_{\max}(v)$. Assume we have another critical chord $[v_0, u]$, therefore $\bar{R} < T^* + d_{\max}(u)$. Combining those two inequalities we obtain $2\bar{R} < d_{\max}(u) + d_{\max}(v) + 2T^* \leqslant \varDelta + 2T^*$. Lemma is proved by contradiction with (2). □

Lemma 2. *Let $I \in \mathcal{I}_2$, node v is overloaded and chord $[v_0, u]$ is critical. Then $u = v$.*

Proof. We have $\varDelta(v) > \bar{R} - 2\mathrm{dist}(v_0, v) \geqslant \bar{R} - T^*$ and $d_{\max}(u) > \bar{R} - T^*$. Assume $u \neq v$, then $\varDelta \geqslant \varDelta(v) + d_{\max}(u) > 2(\bar{R} - T^*)$. Lemma is proved by contradiction with (2). □

Theorem 1. *Let $I \in \mathcal{I}_2$ such that the depot v_0 is overloaded. Then $\alpha(I) = 1$.*

Proof. Note that $\varDelta(v_0) > \bar{R}$. It follows from Lemma 2 that I contains no critical chords, therefore eliminating all the chords is a valid (and reversible) transformation of I. Now let us replace all the jobs except the ones in the depot with a new single job J' with operations processing times $p'_i = T^* + \sum\limits_{J_j \notin \mathcal{J}(v_0)} p_{ji}$, and locate J' at v_0. Obsolete nodes (all except v_0) can now be removed from G, therefore G is transformed into a single-node graph and instance is reduced to the classic $O2||C_{\max}$ problem, for which we know that optimal makespan coincides with the standard lower bound. Such a transformation is reversible, as soon as we can treat the processing of operations of job J' as traveling along the optimal cycle and processing the jobs on the way. It is therefore sufficient to prove the validity of the transformation: $\sum_i p'_i = 2T^* + \varDelta - \varDelta(v_0) \leqslant 2T^* + 2(\bar{R} - T^*) - \varDelta(v_0) < \bar{R}$. □

Now we describe a *chain contraction* transformation. Suppose G contains a chain $C = (v - v_1 - v_2 - \ldots - v_k - u)$, all the nodes $v_1, \ldots, v_k$ are of degree 2, and none of them is the depot. Let τ be the length of chain (the distance between v and u along C) and $\mathcal{J}(C)$ is the set of jobs from nodes $v_1, \ldots, v_k$. We now replace the subchain $v_1 - \ldots - v_k$ with a new *special* node v_C containing single job J_C with processing times $p_{Ci} = \tau + \sum\limits_{J_j \in \mathcal{J}(C)} p_{ji}$, and set weights of edges $[v, v_C]$ and $[v_C, u]$ to zero.

Such a transformation is not reversible in general. To make it reversible we need to apply certain restriction on schedules for the transformed instance:

1. If machine arrives at J_C from one end (say, from v), the machine is considered to be at another end (say, u) after the completion of operation of job J_C.
2. Any machine can bypass the node v_C, but this takes τ time units.

We say that the chain contraction transformation is *conditionally reversible*, meaning that we obtain a special node which has to be treated as described above.

The main result of this paper is the following

Theorem 2. *For any instance $I \in \mathcal{I}_m$ there exists a combination of valid chord eliminations and chain contractions $I \to \tilde{I}$ such that $\tilde{I}$ contains at most $2m$ nodes from which at most m are special.*

Moreover, the structure of the resulting instance $\tilde{I}$ is not arbitrary. For instance, for $m = 2$ the most general structure we need to investigate is the cycle $(v_0 - v_1 - v_2 - v_3 - v_0)$ with additional chord $[v_0, v_2]$ and v_1, v_3 being special nodes. Our working conjecture is that for any instance I of such a special structure $\alpha(I) = \frac{6}{5}$, and therefore $\alpha(\mathcal{I}_2) = \frac{6}{5}$. Theorem 2 can still be useful for a general $ROm||R_{\max}$ problem, although the research for the tight optima localization interval for $m \geqslant 3$ is difficult even for the classic $Om||C_{\max}$ problem.

Acknowledgements

This research was supported by the Russian Foundation for Basic Research, projects 20-01-00045 and 20-07-00458.

References

Averbakh I., Berman O., Chernykh I., 2005, "A 6/5-approximation algorithm for the two-machine routing open shop problem on a 2-node network", *European Journal of Operational Research*, Vol. 166, pp. 3-24.

Averbakh I., Berman O., Chernykh I., 2006, "The routing open-shop problem on a network: complexity and approximation", *European Journal of Operational Research*, Vol. 173, pp. 521-539.

Chernykh I. and Krivonogova O., 2020, "On the Optima Localization for the Three-Machine Routing Open Shop", *Lecture Notes in Computer Science*, Vol. 12095, pp. 274-288.

Chernykh I. and Lgotina E., 2016, "The 2-machine routing open shop on a triangular transportation network", *Lecture Notes in Computer Science*, Vol. 9869, pp. 284-297.

Chernykh I. and Lgotina E., 2019, "Two-machine routing open shop on a tree: instance reduction and efficiently solvable subclass", submitted to *Optimization Methods and Software.*

Gonzalez T.F. and Sahni S., 1976, "Open shop scheduling to minimize finish time", *J. Assoc. Comput. Mach.*, Vol. 23, pp. 665-679.

Krivonogova O. and Chernykh I., 2019, "Optima localization for the two-machine routing open shop on a tree (in Russian)", submitted to *Diskretnyj Analiz i Issledovanie Operacij.*

Sevastyanov S.V. and Tchernykh I.D., 1998, "Computer-aided way to prove theorems in scheduling", *Algorithms — ESA'98 Lecture Notes in Computer Science*, Vol. 1461, pp. 502-513.

Optima Localization for the Routing Open Shop: Computer-aided Proof

Ilya Chernykh[1,2,3] and Olga Krivonogova[1]

[1] Sobolev Institute of Mathematics, Novosibirsk, Russian Federation
idchern@math.nsc.ru
[2] Novosibirsk State University, Novosibirsk, Russian Federation
krivonogova.olga@gmail.com
[3] Novosibirsk State Technical University, Novosibirsk, Russian Federation

Keywords: routing open shop, instance transformation, optima localization, computer-aided approach.

1 Introduction

In the open shop problem sets of jobs $\mathcal{J} = \{J_1, \dots, J_n\}$ and machines $\mathcal{M} = \{M_1, \dots, M_m\}$ are given. Each of job J_j has to be processed by each machine M_i in arbitrary order, and this operation takes a given processing time p_{ji}. The goal is to minimize *the makespan* $C_{\max}$ which is defined as a maximum completion time of the operation. We use notation $Om||C_{\max}$ for the problem with m machines. It is known (Gonzalez T.F. and Sahni S. 1976) to be polynomially solvable in the case $m = 2$ and NP-hard for $m \geq 3$.

We consider the routing open shop problem being a generalization of the metric TSP and the open shop problem. Routing open shop, introduced in (Averbakh I. *et. al.* 2006, Averbakh I. *et. al.* 2005), can be described as follows. Jobs are located at the nodes of a transportation network described by an edge-weighted graph $G = \langle V; E\rangle$, each node contains at least one job. The weight $\mathrm{dist}(u, v)$ represents the travel time of any machine between those nodes. Mobile machines are initially located at some predefined node $v_0 \in V$ referred to as *the depot.* All the machines have to travel between nodes to process jobs in an openshop-like environment, and to return to the depot after completion of all the operations. The makespan $R_{\max}$ is the *return time* moment of the last machine after completion of all its operations, and has to be minimized. We denote this problem as $ROm||R_{\max}$, or as $ROm|G = X|R_{\max}$ in the case we want to specify the structure of the transportation network. Problem is known to be NP-hard even in trivial cases with single machine (equivalent to the metric TSP) and with two machines and just two nodes of the network (including the depot)(Averbakh I. *et. al.* 2006). The latter case is denoted as $RO2|G = K_2|R_{\max}$.

Consider the following *standard lower bound* on the optimal makespan, proposed in Averbakh I. *et. al.* (2005):

$$\bar{R} = \max\left\{\ell_{\max} + T^*, \max_{v\in V}(d_{\max}(v) + 2\mathrm{dist}(v_0, v))\right\}.$$

Here $\ell_{\max} = \max\limits_i \sum\limits_{j=1}^{n} p_{ji}$ is the maximum machine load, $d_{\max}(v) = \max\limits_{j\in\mathcal{J}(v)} d_j = \max\limits_{j\in\mathcal{J}(v)} \left(\sum\limits_{i=1}^{m} p_{ji}\right)$ is the maximum length of job from node v, with $\mathcal{J}(v)$ being the set of jobs located at v. T^* denotes the TSP optimum on G with distance function $\mathrm{dist}(u, v)$.

One of the directions of the research of an NP-hard optimization problem is *optima localization*, i.e. the search of tight upper bound on the optimum in terms of the lower bound LB. More precise, the tight optima localization interval is an interval of type $[LB, \rho LB]$

with the smallest possible value of ρ guaranteed to contain an optimum value for any problem instance from a given set. The first tight optima localization interval for scheduling problems was found for $O3||C_{\max}$ in (Sevastyanov S.V. and Tchernykh I.D. 1998). This research required massive computer-aided enumeration based on the branch-and-bound method.

For the routing open shop problem this question was partly studied for the case of two machines. It is proved in Averbakh I. *et. al.* (2005) that optimum of any instance of $RO2|G = K_2|R_{\max}$ belongs to an interval $[\bar{R}, \frac{6}{5}\bar{R}]$, and the bounds are tight. Lately this result was generalized for the $RO2|G = K_3|R_{\max}$ (Chernykh I. and Lgotina E. 2016) and $RO2|G = tree|R_{\max}$ problems (Krivonogova O. and Chernykh I. 2019). Optima localization for the problem with three or more machines is still an open question even in case $G = K_2$.

2 Instance simplification operations

The research of the optima localization for the two-machine case is based on an instance reduction procedure which uses two simplification operations: job aggregation and terminal edge contraction.

Job aggregation operation (also known as *grouping*) utilizes a simple idea of replacing a number of jobs from the same node with a single aggregated job for which processing times equal to the total processing time of the respective operations combined. We use job aggregation to simplify the instance preserving the standard lower bound $\bar{R}$. A natural question arises, is it possible to perform a job aggregation of a whole set of jobs at some nodes. To answer that question, we use the following definition.

Definition 1. *A node v from $G(I)$ of some problem instance I is* overloaded *if*

$$\Delta(v) = \sum_{J_j \in \mathcal{J}(v)} d_j > \bar{R} - 2\text{dist}(v_0, v).$$

Otherwise the node is referred to as underloaded.

The job aggregation of the whole set of jobs in node v preserves $\bar{R}$ if and only if the node v is underloaded.

Another operation, terminal edge contraction, is based on the following idea: translate a single job from a terminal node v to an adjacent one u, modifying processing times of all of its operations to include travel times (back and forth) between v and u.

Again, we want to perform an edge contraction operation only if it does not lead to the growth of the standard lower bound $\bar{R}$. Otherwise, the edge is called overloaded. The following definition describes the exact condition, under which an edge is overloaded.

Definition 2. *Let $v \neq v_0$ is a terminal node in G and there is a single job $J_j \in \mathcal{J}(v)$. Let $e = [u, v]$ be an edge incident to v. Then edge e is* overloaded *if*

$$d_j + 2m\text{dist}(u, v) + 2\text{dist}(v_0, u) > \bar{R},$$

and is underloaded *otherwise.*

Overloaded elements make the instance somehow problematic. Fortunately, the number of such elements is rather small.

Lemma 1. *Any instance of the $ROm||R_{\max}$ problem contains at most $m - 1$ overloaded elements.*

Moreover, the number of jobs in the simplified instance is small. One of the main results of our research is the following

Lemma 2. *Let I be an instance of the $ROm||R_{\max}$ problem and any job aggregation in I leads to the growth of $\bar{R}$. Then every underloaded node in I contains exactly one job, and all the overloaded nodes (if any) contain at most $2m-1$ jobs altogether.*

Instance simplification preserving the lower bound allows one to reduce the search for the tight optima localization interval to the case with small number of jobs (depending on m) and with simpler structure of the transportation network. The next section covers the first attempts to discover optima localization interval for the three-machine routing open shop.

3 Optima localization for $RO3|G = K_2|R_{\max}$

For any instance of $RO3|G = K_2|R_{\max}$ we use v to denote the node other than the depot.

Back in 1998 Sevastyanov and Chernykh used a computer program to prove that for any instance of $O3||C_{\max}$ (which is equivalent to $RO3|G = K_1|R_{\max}$) optimal makespan does not exceed $\frac{4}{3}$ times standard lower bound. The program is based on an intelligent branch-and-bound-style enumeration of subsets of instances (with infinite cardinality). For each subset a *critical instance* with the greatest ratio of upper and lower bounds of the optimal makespan was found with a help of linear programming. The proof follows from the facts that the enumeration is complete (union of the subsets considered coincides with the whole sets of instances), and for each critical instance found the upper bound is within the range of $\frac{4}{3}\bar{R}$. It took about 200 hours of running time to complete the proof, including building the structure of subsets and the search for critical instances for each one. As it was clear that a direct application of the same approach would take enormous amount of time, we focused our research on the possibilities to make the proof-building process more efficient. As a result, we were able to complete the research of the optima localization of the $RO3|G = K_2|R_{\max}$ problem and to prove the following theorem constructively.

Theorem 1. *For any instance of the $RO3|G = K_2|R_{\max}$ problem there exists a feasible schedule S such that $R_{\max}(S) \leq \frac{4}{3}\bar{R}$.*

One part of the proof is based on a description of a set of sufficient conditions which allow to reduce an instance to the case of $O3||C_{\max}$. Another one used the computer-aided approach with some fine-tuning applied. As a result, the proof-building process was complete in about 28 hours.

Let us focus on the running time reduction techniques. First idea was to try to reduce the set of instances as much as possible without loss of generality. This is done by means of the following two lemmas.

Lemma 3. *For any instance of $RO3|G = K_2|R_{\max}$ with underloaded node v and $p_{\max} = \max p_{ji} \geqslant \frac{2}{3}\bar{R}$ the optimal makespan does not exceed $\frac{4}{3}\bar{R}$.*

Lemma 4. *Let I be an instance for $RO3|G = K_2|R_{\max}$ problem such that $\Delta(v_0) > 2\bar{R}$. Then $R^*_{\max} \leqslant \frac{4}{3}\bar{R}$.*

The influence of the application of different combinations of these restrictions on the running time for one of the special cases of the problem is presented in Table 1.

As one can observe, that influence is not that noticeable. Luckily, we discovered another reserve which surprisingly allowed one to reduce running time significantly.

Second idea was to reduce the set of instances by using symmetries induced by different enumerations of jobs and machines.

	$\Delta(v_0) \leq 2\bar{R}$	$\Delta(v_0)$ is arbitrary
$p_{\max} \leq \frac{2}{3}\bar{R}$	17:52 min.	19:27 min.
$p_{\max}$ is arbitrary	20:27 min.	29:31 min.

Table 1. Running time of the original program depending on the restriction applied.

	$\Delta(v_0) \leq 2\bar{R}$	$\Delta(v_0)$ is arbitrary
$p_{\max} \leq \frac{2}{3}\bar{R}$	00:10 min.	01:45 min.
$p_{\max}$ is arbitrary	00:09 min.	02:14 min.

Table 2. Running time of the modified program depending on the restriction applied.

Further ways to improve efficiency are based on details of the computer-aided approach and cannot be fully disclosed in the format of the current abstract. The results are covered in Table 2.

Thus we were able to reduce the running time (for one of the special cases) by the factor of almost 200, which gives us hope that the computer-aided approach can still be used for wider classes of problems, *i.e.* $O4||C_{\max}$ (an intriguing case, as we no evidence that the optimal makespan can be greater than $\frac{4}{3}\bar{R}$), $RO3|G = K_3|R_{\max}$ and so on.

4 Conclusion

The main results of this paper are the following.

1. Description of the extremal properties of overloaded elements of $ROm||R_{\max}$ problem.
2. The optima localization of the special case of the $RO3|G = K_2|R_{\max}$ problem.
3. Developments of the computer-aided approach with a significant reduction of the running time.

An intriguing open question from (Sevastyanov S.V. and Tchernykh I.D. 1998) still remains: does there exist an analytic proof of Theorem 1 (as well as the optima localization result for $O3||C_{\max}$), such that doesn't require any computer-aided enumeration.

Acknowledgements

This research was supported by the Russian Foundation for Basic Research, project 20-01-00045.

References

Gonzalez T.F. and Sahni S., 1976, "Open shop scheduling to minimize finish time", *J. Assoc. Comput. Mach.*, Vol. 23, pp. 665-679.

Averbakh I., Berman O., Chernykh I., 2006, "The routing open-shop problem on a network: complexity and approximation", *European Journal of Operational Research*, Vol.173, pp. 521–539.

Averbakh I., Berman O., Chernykh I., 2005, "A 6/5-approximation algorithm for the two-machine routing open shop problem on a 2-node network", *European Journal of Operational Research*, Vol. 166, pp. 3–24.

Sevastyanov S.V. and Tchernykh I.D., 1998, "Computer-aided way to prove theorems in scheduling", *Algorithms - ESA '98 Lecture Notes in Computer Science*, Vol. 1461, pp. 502–513.

Chernykh I. and Lgotina E., 2016, "The 2-machine routing open shop on a triangular transportation network", *Lecture Notes in Computer Science*, Vol. 9869, pp. 284-297.

Krivonogova O. and Chernykh I., 2019, " Optima localization for the two-machine routing open shop on a tree (in russian)", submitted to *Diskretnyj Analiz i Issledovanie Operacij.*

A new tool for analysing and reporting solutions for the RCPSP and MMRCPSP

José Coelho[1,3], Mario Vanhoucke[1,2], and Ricardo Amaro[3]

[1]Ghent University, Belgium
e-mail: jose.coelho, mario.vanhoucke@ugent.be

[2]Vlerick Business School, Belgium
[3]Universidade Aberta, Portugal
e-mail: jose.coelho@uab.pt

Keywords: RCPSP, datasets, reporting results.

1. Introduction

In the paper written by Vanhoucke and Coelho (2018), a new method is proposed to facilitate the reporting of results for the single- and multi-mode RCPSP. We have now extended this method with a website where researchers can download and upload solutions without much intervention, which is the topic of this abstract.

The new website does not want to replace the well-known existing libraries such as the PSPLIB proposed in Kolisch and Sprecher (1996), the MMLIB proposed in Peteghem and Vanhoucke (2014) or the generic OR-LIBRARY proposed by Beasley (1990), but rather serves as a complement. The website reports data about many benchmark datasets from the literature in a standardized way, and also provides the best LB/UB/optimal values, the best known solutions (start times of each activity), and also information about project indicators (network and resource indicators). We have saved exactly one result file for each run of a complete dataset, and the performance of the procedure used is calculated against the CPM lower bound as well against the current best LBs and UBs.

We also present two new datasets for the RCPSP. The first so-called NetRes has already been proposed earlier in Vanhoucke and Coelho (2018), which is a large set of 30 activity instances that spans a wide range for the topological network structure. The second set is totally new and is proposed in Coelho and Vanhoucke (2020) and contains a small set of very hard instances with 20 to 30 activities. This so-called CV set contains the smallest possible instances that we could find for which no optimal solutions could be found using the fast and efficient branch-and-bound procedures from the literature.

In the remainder of this abstract, we will detail how the results are reported (Section 2). In Section 3, we describe how we will update the website tables with best known solutions for the RCPSP and the MMRCPSP. Section 4 provides an illustrative example of an experiment with NetRes. In Section 5, we show the diversity of the new CV dataset, and we conclude in Section 6.

2. Reporting new results

The method we propose for reporting new results is done using a single *data file* per dataset (in CSV format) rather than one file per instance, containing one line per instance. Each line contains all possible data for that instance, such that user can easily know the network and resource indicators for each instance in the set. The results are given in a singe *result file* (also in CSV format). Consequently, our method requires only a single file for each run and avoids the need to submit one result per instance. Not only the values of the LBs and UBs are made available, but also the obtained solutions by the author of the new algorithm (the start times of each activity), and these results can be interesting for other researchers.

A software tool – a client tool - was developed to allow users the read and modify the results file if they have found new and better results. In doing so, the LBs and UBs are checked automatically for errors or inconsistencies. If no errors are found, the results file is updated and a reference to the new paper for new solutions is given. The tool is also easy to use for selecting only a subset of instances of a dataset (e.g. only the open or closed files or the files with LBs x% from the best known UB) and the instances will be automatically be selected for the user in a so-called *instance file*.

The website is in solutionsupdate.ugent.be and is integrated in the projectmanagement.ugent.be/research/data. The website will maintain and update Table 2 and Table 3 of Vanhoucke and Coelho (2018) that contains data from several datasets. Other tables for other project scheduling problems can be also added in the future.

Even if no new results are found, the website can be used to submit results before the submission of the paper, and in doing so, the authors will have a confirmation that there results contains no inconsistencies (such as UBs lower than a strong LB). This can be done easily using the client tool, but when the results are put online, it also gives the reviewers the possibility to check.

3. Update of tables of BKS on RCPSP and MMRCPSP

In this section we report current results for the tables that we intend to keep updated in the website. *Table 1* displays the current best-known results for the RCPSP and is an update of Table 2 published in Vanhoucke and Coelho (2018). More specifically, we updated the table with the new CV set and the Patterson set. For the NetRes set, we also reported the results for the 1kNetRes set, which contains results for a subset of NetRes in which each instance is selected in steps of 1,000 (reducing the number of instances to e.g. 540,000 to 540 for the NR(SP) set). We can now compare the results with the table in the original paper to see the progress made in the last few years by many authors. The table reports the number of open instances in the PSPLIB (J60 to J120) have been reduced. This data is not easily detectable from the PSPLIB website as done in Table 1.

Table 1. Best-known results for the RCPSP

Dataset	Subset	#Instances	#Open	%CPM	GAP
CV		623	623	142.21%	3.3
RG30		1,800	116	39.27%	2.0
RG300		480	377	956.71%	35.2
DC1		1,800	0	26.57%	0.0
DC2		720	210	274.20%	7.6
PSPLIB	J30	480	0	13.38%	0.0
	J60	480	37	10.37%	6.3
	J90	480	66	9.43%	7.5
	J120	600	290	29.01%	8.0
NetRes	NR(SP) | 1k	540,000 | 540	25,591 | 12	78.8% | 72.9%	5.3 | 1.8
	NR(AD) | 1k	480,000 | 480	44,855 | 7	98.8% | 102.4%	5.6 | 1.1
	NR(LA) | 1k	720,000 | 720	246 | 0	58.4% | 58.9%	4.6 | 0.0
	NR(TF) | 1k	720,000 | 720	23,544 | 0	68.3% | 64.7%	6.4 | 0.0
	NR(RC) | 1k	540,000 | 540	10,333 | 0	66.3% | 71.6%	6.0 | 0.0
	NR(RU) | 1k	270,000 | 270	3,761 | 0	73.6% | 77.0%	9.3 | 0.0
	NR(VAR) | 1k	540,000 | 540	4,722 | 0	87.3% | 91.9%	4.3 | 0.0
Patterson		110	0	18.04%	0.0

As mentioned earlier, with this updated data, a reviewer can easily check whether some new results on the RCPSP are within a valid range by e.g. checking the percentage deviation of the LB over the CPM. Also, the sum of time units of both best lower bounds and best upper bounds is provided, and this indicator can be checked in the same way than the %CPM. This does not rule out the possibility of less credible researchers to invent and manipulate results, but prevents errors unwillingly made by the researchers. Nevertheless, the reviewer can also ask the researcher to submit a result file to the website, so the results can always be checked, even if there are no new LBs or UBs.

Table 2 displays the current best-known results for the MMRCPSP, and is an update of Table 3 published in Vanhoucke and Coelho (2018). The LBs are updated with the work of Stürck (2018), and compared with the version published in the paper, this new data lead to a larger number of instances closed in the MMLIB. This illustrates and highlights the importance of research on LBs as much as on UBs. Note that in the Boctor instances, the GAP between the UBs and the LBs is very high. This is mainly because no good LBs exist for these instances, since these instances do not contain non-renewable resource. The MMLIB site is no longer available, but the final UB values from 2018 are used in our website to guarantee we have used to most recent results. As for the RCPSP, a reviewer can also check new results.

Table 2. Best-known results for the MMRCPSP

Dataset	Subset	#Instances	#Open	%CPM	GAP
PSPLIB	J10 | J12 | J14 | J16 | J18 | J20	536 | 547 | 551 | 550 | 552 | 554	0 | 0 | 0 | 0 | 0 | 0	32% | 27% | 24% | 19% | 18% | 17%	0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0
	J30	552	245	12.28%	6.5
Boctor	Boct50	120	120	22.74%	52.6
	Boct100	120	120	22.91%	103.6
MMLIB	MMLIB50	540	95	22.29%	9.3
	MMLIB100	540	151	21.35%	10.8
	MMLIB+	3240	2439	78.77%	37.2

4. An example of an experiment with NetRes

The NetRes set was proposed in Vanhoucke and Coelho (2018), and the goal was to create a set with high diversity in terms of the project indicators, but also to provide a large number of instances available such that researchers can select subsets they need. Several analyses are done in the original paper, but we have select Table 5 of the original paper and replicate results in *Table 3* that measures the impact of the project indicators using the exact procedure of Demeulemeester and Herroelen (1992). An instance is considered hard if it could not be solved in 1 second, and the table shows the percentage of hard instances of each value of the project indicator (SP, AD, LA, TF, OS, RC and RS).

Table 3. Percentage of hard instances in NetRes depending on each project indicator

	SP	AD	LA	TF	OS	RC	RS
0-0,1	54%	-	4.9%	0.4%	59%	0%	7.4%
0,1-0,3	13%	1.5%	0.3%	0.5%	31%	9.1%	7.2%
0,3-0,5	0.2%	4.5%	0.1%	2.8%	1.3%	4.9%	1.3%
0,5-0,7	0%	13%	0.1%	8%	0.4%	3.8%	0.5%
0,7-0,9	0%	4.9%	0%	17%	0%	1.8%	0.1%
0,9-1	-	1.9%	-	24%	0%	1.4%	0%

As we can see in *Table 3*, most of the instances in this set are closed, but we can now visualize where the most complex instances are for each indicator. All the findings are more or less known (except for the new project indicators AD, LA and TF). For example, parallel networks (low SP and OS values) are harder to solve, and for the RC indicator, an easy/hard/easy phase transition is found, which confirms the results of Herroelen and De Reyck (1999). A similar effect is found for the AD indicator, and the indicators LA and RS provide more hard instances when the indicator is low. The TF indicator provides harder instances when it is high.

The *Table 3* is an example of an experiment that could not be easily done if no instances are available for all values of all these indicators. Vanhoucke et. al. (2016) have shown that most sets are not diverse enough, and only contain instances with values between 0 and 1 for some indicators, while others are largely ignored.

We expect that the NetRes set will be interesting for research where statistical tests are used extensively. A deeper study into the relation between a given project indicator and the performance of a solution procedure requires data that spans the full range of complexity. The client tool can help selecting the subset of instances necessary for such a study. Moreover, the large volume of instances with solutions could potentially be interesting for researcher using machine learning making use of the current best-known solutions on a large amount of data to train the data.

5. Diversity of dataset CV

Table 4 displays the distribution of the CV instance set for several project indicators used in Vanhoucke et. al. (2016). Recall that this set contains instances that are currently unsolvable. The table shows that this set of hard instances still contains instances with diversity in the network structure and resource constraints, and hence, not only contains instances with very parallel activities. All topological indicators are spread over a wide interval except for LA that is concentrated around values below 0,2. For the resource indicators, the RS is not very diverse and most of the instances have a value lower than 0,2. The diversity is higher for the other resource indicators, with RU greater than 2, RC between 0,2 and 0,5, RF greater than 0,8. This set is said to be very hard to solve, and researchers could focus their research time trying to solve these instances to optimality.

Table 4. Distribution of instances in CV dataset by several project indicators

#Activities	#Resources	CNC	OS	SP	AD
20-21 #4	1 #1	0-1 #405	0-0,1 #85	0-0,1 #232	0-0,2 #8
22-23 #18	2 #39	1-2 #176	0,1-0,2 #416	0,1-0,2 #316	0,2-0,4 #79
24-25 #41	3 #85	2-3 #24	0,2-0,3 #99	0,2-0,3 #62	0,4-0,6 #243
26-27 #95	4 #498	3-4 #6	0,3-0,4 #13	0,3-0,4 #12	0,6-0,8 #229
28-30 #465		4-8 #12	0,4-0,6 #10	0,4-0,5 #1	0,8-1 #64
LA	**TF**	**RC**	**RF**	**RU**	**RS**
0-0,2 #592	0-0,2 #28	0,2-0,3 #57	0,5-0,6 #9	1-2 #34	0-0,1 #533
0,2-0,4 #10	0,2-0,4 #48	0,3-0,4 #258	0,6-0,7 #13	2-3 #84	0,1-0,2 #87
0,4-0,6 #11	0,4-0,6 #138	0,4-0,5 #285	0,7-0,8 #153	3-4 #505	0,2-0,3 #3
0,6-0,8 #7	0,6-0,8 #220	0,5-0,6 #10	0,8-0,9 #245		
0,8-1 #3	0,8-1 #189	0,6-0,8 #13	0,9-1 #203		

The reason why we claim these instances are hard is that we have tried to solve these instances using 20 hours of CPU time for each instance with the procedure presented in Coelho and Vanhoucke, M. (2018), and we have reported the best found LB and UB. The percentage over the CPM of LBs is 129%, and this percentage increases to 142% when compared with the UBs, leaving enough space to find improvements for the 623 instances.

It is interesting to note that we have kept these instances as small as possible. Most instances contain 20 activities, and go up to 30 activities maximum, and some of them make use of only 1 renewable resource.

6. Conclusion

In this abstract, we present a new contribution to the academic community with a tool to keep the current results for the RCPSP and MMRCPSP updated at all times. The tool intends to save the latest results from all datasets in a standardized way, validates new results and provides performance indicators. We also provided a new large dataset NetRes that is diverse in several project indicators, allowing doing analyses for several project indicators, and a second new dataset CV with only small instances that are still not solved to optimality. We hope and believe that this tool and the new dataset can be used in new research studies, which can lead to entirely new solution procedures that can solve small but very hard instances to optimality.

References

Beasley J.E., 1990, “OR-Library: Distributing Test Problems by Electronic Mail ", Journal of the Operational Research Society, Vol. 41 (11), pp. 1069-1072.

Coelho, J., Vanhoucke, M. (2018). An exact composite lower bound strategy for the resource-constrained project scheduling problem. Computers & Operations Research, 93, 135–150.

Coelho, J., Vanhoucke, M. (2020). Going to the core of hard resource-constrained project scheduling instances. Computers & Operations Research, 121, 104976.

Demeulemeester E., W. Herroelen, 1992, “A branch-and-bound procedure for the multiple resource-constrained project scheduling problem”, Management Science, Vol. 38, pp. 1803–1818.

Herroelen W., B. De Reyck, 1999, “Phase transitions in project scheduling”, Journal of the Operational Research Society, Vol. 50, pp. 148–156.

Kolisch R., A. Sprecher, 1996, “PSPLIB – A project scheduling problem library”, European Journal of Operational Research, Vol. 96, pp. 205-216

Peteghem V.V., M. Vanhoucke, 2014, “An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances”, European Journal of Operational Research, Vol. 235, pp. 62-72

Vanhoucke M., J. Coelho, J. Batselier, 2016, “An overview of project data for integrated project management and control”, Journal of Modern Project Management, Vol. 3 (2), pp. 6–21.

Vanhoucke M., J. Coelho, 2018, "A tool to test and validate algorithms for the resource-constrained project scheduling problem", Computers & Industrial Engineering, Vol. 118 (1), pp. 251 – 265

Stürck C., 2018, “Exakte Methoden und Matheuristiken für das Multi-Mode Resource-Constrained Project Scheduling Problem (Exact and matheuristic approaches for the multi-mode resource-constrained project scheduling problem)”, Dissertation, Helmut-Schmidt-University, Hamburg

Adaptive Robust Parallel Machine Scheduling

Izack Cohen[1], Krzysztof Postek[2] and Shimrit Shtern[3]

[1] Faculty of Engineering, Bar-Ilan University, Ramat-Gan, Israel
izack.cohen@biu.ac.il
[2] Faculty of Electrical Engineering, Mathematics and Computer Science, University of Technology, Delft, The Netherlands
k.s.postek@tudelft.nl
[3] Faculty of Industrial Engineering and Management, Technion - Israel Institute of Technology, Haifa, Israel
shimrits@technion.ac.il

Keywords: robust optimization, machine scheduling, adaptive scheduling.

1 Problem definition:

Parallel machine scheduling (PMS) problems are multi-stage scheduling problems, which are widely researched owing to their theoretical importance and multiple applications in manufacturing, cloud computing, and project management, among others. Real-life PMS settings involve uncertainty about task duration, which may be characterized by the randomness of each task duration and, possibly, a dependence between task durations.

An ideal scheduling approach should accommodate uncertainty to ensure realistic guarantees on the objective function value and permit adjustments of later-stage decisions based on different observed task lengths (e.g., different duration realizations of the task scheduled first may result in different allocation decisions of the next tasks).

Real-life parallel machine scheduling problems can be characterized by: (i) limited information about the exact task duration at scheduling time, and (ii) an opportunity to reschedule the remaining tasks each time a task has completed processing and a machine becomes idle. Robust scheduling has been used to deal with the first characteristic. However, the existing literature on robust scheduling does not explicitly consider the second characteristic – the possibility to adjust decisions as more information about the tasks' duration becomes available, despite the fact that re-optimizing the schedule every time new information emerges is a standard practice.

2 Methodology/results:

In this paper, we develop a robust optimization based scheduling approach that takes into account, at the beginning of the planning horizon, the possibility that scheduling decisions can be adjusted. We demonstrate that this adaptive approach can lead to better here-and-now decisions. To that end, we develop the first mixed integer linear programming model for *adjustable robust scheduling*, where we minimize the worst-case makespan. Using this model, we show via a numerical study that adjustable scheduling leads to solutions with better and more stable makespan realizations compared to static approaches.

We focus on makespan minimization, which is a standard performance measure for PMS. Indeed, makespan minimization is used for load balancing, an important issue for many scheduling applications. When deciding whether to use the expected value or worst-case value, several factors should be considered. Optimizing over an expectation requires specifying the full probability distribution of task duration, information that is often not readily available or is costly to acquire. Moreover, the makespan of a single realization can significantly differ from the expected value; thus, if the exact scheduling problem is not repeated multiple times, optimizing over the expected value may not be translated into good performance in practice. In contrast, much less information is needed when specifying a set that includes all the reasonable duration realizations, and a worst-case optimization approach provides a guarantee on the performance of any realization in such a set. Therefore, we choose a setting where the scheduler minimizes the worst-possible makespan of a set of tasks over some uncertainty set, which captures all reasonable scenarios within the support of the distribution. This is in line with the paradigm of Robust Optimization (RO), where the best solution is sought under the assumption that the problem's parameters are initially unknown and that, given the decisions, nature picks their worst-possible values from an *uncertainty set* consisting of outcomes that include the true realization with high probability.

We consider the classical version of PMS, where m identical machines process $n \geqslant m$ tasks that are available at the start of the scheduling horizon. For this problem, we construct a mixed integer linear optimization problem for minimizing the worst-case makespan, which includes all possible later-stage (re-)scheduling decisions. We compare the adaptive formulation's optimal scheduling decisions and optimal worst-case makespan to those of the optimal static allocation (SA) and static list (SL) policies.

In contrast to the majority of previous works, which compare naive implementations of the SA and SL policies without re-optimization (i.e., re-scheduling) as more information is revealed, we consider the more realistic rolling horizon implementation of these policies. Under this implementation, whenever one of the machines becomes idle, the scheduler can alter the initial order of tasks by re-solving an optimization problem with the extra information included.

3 Managerial implications:

We outline our main managerial insights for the studied setting. The insights are relevant to schedulers within multiple domains that can be modeled via PMS such as production lines in which machines process a set of tasks, computer multiprocessors ("cloud processing") for processing jobs, shipyards and ports in which ships are loaded and unloaded, doctors who treat patients in a walk-in clinic or triage setting, and teachers who educate student groups, just to name a portion of the potential use-cases.

First, our study shows that capturing the uncertainty and the relations between the durations of different tasks is vital to a realistic assessment of the makespan. Indeed, there are many settings in which the probabilistic knowledge about task durations is limited or costly to attain. In such circumstances, it is rather easy to design a polyhedral or ellipsoidal uncertainty set that frames the involved uncertainty. Ben Tal et al.(2009) provide guidance and probabilistic guarantees in favor of designing uncertainty sets that balance the level of conservatism and the probability that a constraint is violated by a scenario. Ideally, we would like to design the smallest uncertainty set that still captures the meaningful scenarios (e.g., the probability that a scenario is not included within the uncertainty set is lower than a pre-specified threshold).

Secondly, whenever the optimal wait-and-see decisions can be taken into account in the planning stage, this should be done as it lowers the maximum possible project makespan that the scheduler can promise. In other words, a bid prepared by a decision-maker who accommodates wait-and-see decisions and thus can commit to a lower makespan (and cost) would be more competitive than a bidder that does not explicitly take into account the possibility that decisions can be adapted. In particular, our experiments point out that the average advantage of adaptive-based bids is estimated to be $5-9\%$ over its non-adaptive (i.e., 'regular' RO) counterpart. We note that an adaptive policy need not necessarily be achieved by solving our mixed integer linear formulation. Indeed, it is likely that heuristic methods can be of help as well, and should be explored as an alternative to static policies.

While the previous point dealt with the superiority of adaptive robust policies over their static counterparts in the planning and contract stage, they are also preferable in the implementation stage. Specifically, policies that take the later-stage adaptivity of the decisions into account remain preferable even when the static policies are re-optimized every time new information becomes available (rolling-horizon). A hint into the reason for this is provided by the $42-59\%$ of the problem instances in which an adaptive policy yielded different first-stage decisions compared to a SA policy. That means that the adaptive policies not only offer better project makespan guarantees, but also select decisions that lead to better realized duration.

A very attractive feature of the adaptive policies, as revealed through our experiments, is that their performance is comparable to the perfect hindsight policy (e.g., the average difference between the promised and max perfect hindsight makespans was $0.0-0.1\%$ for the optimal adaptive policy compared to $5.7-9.8\%$ for the static robust policy). This suggests that the adaptive robust policy not only protect the decision-maker against adversarial realizations of reality but it also performs close to the perfect hindsight policy. Thus, the typical criticism about the conservatism of static robust policies (i.e., the high price paid for robustness) does not apply to the adaptive scheduling policy.

In conclusion, while robust SA policies are widely investigated and used in risk averse settings, they may achieve inferior performance in practice compared to adaptive alternatives. Since the performance gap between an optimal adaptive policy and a static one is quite significant, we recommend allocating resources for finding good adaptive policies, even if those policies are not necessarily optimal. We believe that these adaptive policies will grant their users competitive advantages both in the proposal bidding stage and in the implementation stage.

Acknowledgements

We are grateful to Michael Pinedo for consultations at the early stage of this work. Also, we thank Esther Julien for her help.

References

Ben-Tal, A., El Ghaoui, L., & Nemirovski, A. (2009). Robust optimization. Princeton University Press.

The Resource-Constrained Project Scheduling Problem: New Benchmark Results

Creemers S[1][2]

[1] IESEG School of Management, France
s.creemers@ieseg.fr
[2] KU Leuven, Belgium
stefan.Creemers@kuleuven.be

Keywords: project scheduling, RCPSP, lower bound.

1 Introduction

Over the past decades, the resource-constrained project scheduling problem (RCPSP) has become a standard problem in the operations research literature. The goal of the RCPSP is to schedule a set of project activities $V = \{0, 1, \ldots, n\}$ such that the makespan of the project is minimized, while satisfying precedence and resource constraints. Precedence constraints impose that an activity $i : i \in V$ can only start upon completion of all its predecessors $\mathbf{P}_i = \{j|(j,i) \in E\}$, where E is the set of precedence constraints. Resource constraints, on the other hand, impose that an activity i can only be scheduled if sufficient resources are available. There are K renewable resource types, and the availability of each resource type $k : k \in \mathcal{K} = \{1, 2, \ldots, K\}$ is denoted by R_k. Each activity i requires $r_{i,k}$ units of resource k. A solution to the RCPSP is a schedule $S = \{S_0, S_1, \ldots, S_n\}$, where S_i is the starting time of activity i. The project starts at time $S_0 = 0$, and completes at S_n, where activities 0 and n are dummy activities that represent the start and the completion of the project, respectively. In addition, define $\mathscr{A}(S,t) = \{i : S_i \leq t \wedge (S_i + p_i) \geq t\}$, the set of activities in schedule S that are active at time t, where p_i is the duration of activity i. Without loss of generality, we assume $p_i \in \mathbb{N}$ for all $i : i \in V$. The RCPSP can then be formulated as a combinatorial optimization problem:

$$\begin{array}{lll}
\min & S_n & \\
\text{s.t.} & S_i + p_i \leq S_j & \forall (i,j) \in E \\
& \displaystyle\sum_{i \in \mathscr{A}(S,t)} r_{i,k} \leq R_k & \forall t \geq 0, \forall k \in \mathcal{K} \\
& S_i \geq 0 & \forall i \in V.
\end{array}$$

Note that we assume that activities are executed without preemption, and that scheduling decisions are made at discrete points in time.

Blazewicz et al. (1983) have shown that the RCPSP is strongly $\mathscr{NP}$-hard, which explains the abundance of heuristic solution methods in the literature (refer to, e.g., Kolisch and Padman, 2001; Kolisch and Hartmann, 2006; Hartmann and Briskorn, 2010). In this abstract, however, we focus on exact methods. Among exact methods, the branch-and-bound (BB) procedures of Demeulemeester and Herroelen (1992, 1997), Mingozzi et al. (1998), and Sprecher (2000) are still the most successful approaches to solve the RCPSP. In what follows, we compare the performance of several lower bounds (LBs), and compare the performance of four BB procedures and the state-of-the-art procedure of Demeulemeester and Herroelen (1997).

2 Branch-and-bound procedures

A BB procedure is an iterative algorithm that implicitly enumerates the state space of a combinatorial optimization problem using a search tree. The root node of the tree corresponds to the original optimization problem, and its child nodes correspond to subproblems or feasible solutions (also called "leaf nodes"). Each iteration, a search strategy is used to select the active node from which new nodes (i.e., children) are generated using a branching scheme. If it can be shown (using dominance rules and/or bounds) that the optimal solution cannot be found by visiting the children of a node, that node is fathomed (i.e., its branch is pruned from the search tree). Eventually, if all nodes have been (implicitly) visited, the procedure stops, and the optimal solution is obtained as the best feasible solution found in any of the leaf nodes.

Several BB procedures for solving the RCPSP have been proposed in the literature. In most of these procedures, nodes correspond to partial schedules, and a depth-first search strategy is used to select the next active node. The procedures, however, use different branching schemes and pruning methods (i.e., dominance rules and LBs). In this abstract, we consider the following branching schemes:

– The precedence tree branching scheme proposed by Patterson et al. (1989), and later on adopted by Sprecher (1998).
– The delay alternatives branching scheme proposed by Christofides et al. (1987), and later on adopted by Demeulemeester and Herroelen (1992, 1997).

For other branching schemes, refer to Stinson et al. (1978), Mingozzi et al. (1998), and Igelmund and Radermacher (1983). Even though a depth-first search strategy is generally accepted as the best choice, in this abstract, we will also consider a breadth-first search strategy. This results in four BB procedures to be tested.

3 Dominance rules

Several dominance rules have been proposed in the literature. The state-of-the-art procedure of Demeulemeester and Herroelen (1992, 1997) uses the following dominance rules:

– DH1: if an activity i cannot be scheduled together with any other activity, it should be started at the first time possible.
– DH2: if a pair of activities i and j cannot be scheduled together with any other activity, they should be started at the first possible time possible.
– LLS: we can fathom a node that is associated with a (partial) schedule where an activity i can be scheduled 1 time unit earlier (i.e., where activity i can be left-shifted). A global variant of this local left shift rule can also be devised (see e.g., Schrage, 1970).
– CUT: we can fathom a node associated with (partial) schedule S' if we previously have saved another (partial) schedule S'' that dominates schedule S'. Note that Demeulemeester and Herroelen (1992, 1997) use a hash function to save (partial) schedules.

From these dominance rules, we will only consider CUT. In contrast to Demeulemeester and Herroelen (1992, 1997), however, we do not overwrite a schedule if a schedule is already stored at a given hash (i.e., we keep track of all schedules).

4 LBs

In the literature, we distinguish between two classes of LBs: (1) complex procedures that are used to obtain a very tight LB and (2) fast procedures that can be evaluated as part of a subroutine in the nodes of a BB procedure. In this abstract, we focus on the latter class of LBs. Several of these LBs have been proposed in the literature (refer for instance to Brucker et al. (1999, 2003), Klein and Scholl (1999), Knust (2015), and Coelho and Vanhoucke (2018)). In this abstract, we consider the following LBs:

– CPL: the critical path LB.
– CS: the critical sequence LB that was introduced by Stinson et al. (1978).
– LB3: the node-packing LB as implemented by Demeulemeester and Herroelen (1997). This bound ranks activities in a list based on the number of "companions" each activity has (i.e., the number of activities with which it can be scheduled in parallel). To construct the LB, activities (and their companions) are removed from the list.

Next, we propose two new LBs:

– LB4: an extension of LB3 that recalculates the list of activities each time an activity (and its companions) is removed from the list.
– LBO: an overflow LB that determines the overflow (i.e., the remaining work contents) of activities that cannot be fully scheduled in a schedule that uses the critical path as a baseline; the remaining work contents is scheduled after the critical path.

The performance of the different LBs is evaluated for the instances of the well-known J30, J60, J90, and J120 PSPLIB data sets (Kolisch and Sprecher, 1996). For each data set, Figure 1 reports how often a LB is one of the dominant LBs. Conversely, Figure 2 reports how often a LB is uniquely the dominant LB. From the results it is clear that LBO performs very well, especially if larger projects are considered.

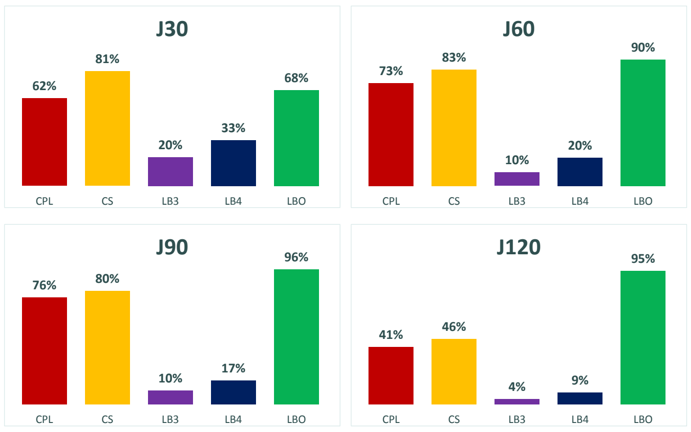


Fig. 1. How often is a LB the best LB

5 Results

In this section, we compare the performance of the procedure of Demeulemeester and Herroelen (1997) (as implemented in the Rescon software; see also Deblaere et al. (2009)) and four other BB procedures:

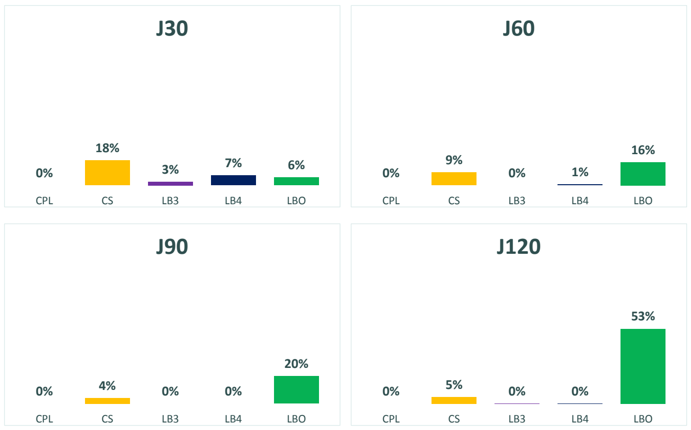


Fig. 2. How often is a LB better than any other LB

– Precedence tree branching scheme and depth-first search strategy.
– Precedence tree branching scheme and breadth-first search strategy.
– Delay alternatives branching scheme and depth-first search strategy.
– Delay alternatives branching scheme and breadth-first search strategy.

Each of these BB procedures is equipped with the CUT dominance rule, and with LBs CPL, CS, LB3, LB4, and LBO. In addition, the procedures that adopt a breadth-first search strategy evaluate a trivial upper bound in each node. We use these procedures to solve all instances of the J30 PSPLIB data set. In order to make a fair comparison, all tests are performed on the same computer: an Intel I5 2.6GhZ computer with 8GB of working memory.

The result of the preliminary test are reported in Table 1. From the table it is clear that we easily outperform the procedure of Demeulemeester and Herroelen (1997). In addition, the results also show that, in contrast to popular belief, a breadth-first search strategy almost performs as good as a depth-first search strategy.

Table 1. Benchmark results for procedure of Demeulemeester and Herroelen (1997) and various other procedures that are equipped with dominance rule CUT and LBs CPL, CS, LB3, LB4, and LBO

	DH1997	Precedence Tree		Delay Alternatives	
		Depth First	Breadth First	Depth First	Breadth First
# Nodes	19.30E6	167.14E6	14.48E6	11.66E6	6.36E6
CPU (sec)	374	1185	209	207	149

We have also performed a number of other preliminary tests that use new dominance rules and LBs that have not been discussed in this abstract. The results of these experiments have shown that we can solve all instances of the PSPLIB J30 data set in 27 seconds (while visiting 6.15E6 nodes). If we minimize the number of visited nodes, we end up with 1.29E6 nodes over all instances of the J30 data set (with a CPU time of 183 seconds). Compared to the state-of-the-art procedure of Demeulemeester and Herroelen (1997), this results in a improvement of factor 13.8 for CPU times and of 14.8 for memory requirements.

References

Blazewicz J., Lenstra J.K. and Rinnooy Kan A.H.G., 1983, "Scheduling subject to resource constraints: Classification and complexity", *Discrete Applied Mathematics*, Vol. 5, pp. 11–22.

Brucker P., Drexl A., Möhring R., Neumann K. and Pesch E., 1999, "Resource-constrained project scheduling: Notation, classification, models, and methods", *European Journal of Operational Research*, Vol. 112, pp. 3–41.

Brucker P., Knust S., 2003, "lower bounds for resource-constrained project scheduling problems", *European Journal of Operational Research*, Vol. 149, pp. 302–313.

Christofides N., Alvarez-Valdes R. and Tamarit J.M., 1987, "Project scheduling with resource constraints: A branch and bound approach", *European Journal of Operational Research*, Vol. 29, pp. 262–273.

Coelho J., Vanhoucke M., 2018, "An exact composite lower bound strategy for the resource-constrained project scheduling problem", *Computers and Operations Research*, Vol. 93, pp. 135–150.

Deblaere F., Demeulemeester E. and Herroelen W., 2009, "RESCON: Educational project scheduling software", *Computer Applications in Engineering Education*, Vol. 19, pp. 327–336.

Demeulemeester E., Herroelen W., 1992, "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem", *Management Science*, Vol. 38, pp. 1803–1818.

Demeulemeester E., Herroelen W., 1997, "New benchmark results for the resource-constrained project scheduling problem", *Management Science*, Vol. 43, pp. 1485–1492.

Hartmann S., Briskorn D., 2010, "A survey of variants and extensions of the resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 207, pp. 1–14.

Igelmund G., Radermacher F.J., 1983, "Preselective strategies for the optimization of stochastic project networks under resource constraints", *Networks*, Vol. 13, pp. 1–28.

Klein R., Scholl A., 1999, "Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling", *European Journal of Operational Research*, Vol. 112, pp. 322–346.

Kolisch R., Sprecher A., 1996, "PSPLIB: A project scheduling problem library", *European Journal of Operational Research*, Vol. 96, pp. 205–216.

Kolisch R., Padman R., 2001, "An integrated survey of deterministic project scheduling", *Omega*, Vol. 29, pp. 249–272.

Kolisch R., Hartmann S., 2006, "Experimental investigation of heuristics for resource-constrained project scheduling: An update", *European Journal of Operational Research*, Vol. 174, pp. 23–37.

Knust S., 2015, "Lower bounds on the minimum project duration". *in Schwindt and Zimmermann, Eds. Handbook on Project Management and Scheduling: Vol. 1*, Springer: Heidelberg, pp. 43–55.

Mingozzi A., Maniezzo V., Ricciardelli S. and Bianco L., 1998, "An Exact Algorithm for the Resource-Constrained Project Scheduling Problem Based on a New Mathematical Formulation", *Management Science*, Vol. 44, pp. 714–729.

Schrage L., 1970, “Solving resource-constrained network problems by implicit enumeration: Nonpreemptive case”, *Operations Research*, Vol. 18, pp. 225–235.

Sprecher A., 2000, “Scheduling Resource-Constrained Projects Competitively at Modest Memory Requirements”, *Management Science*, Vol. 46, pp. 710–723.

Stinson J.P., Davis E.W. and Khumawala B., 1978, “Multiple resource-constrained scheduling using branch-and-bound”, *AIIE Transactions*, Vol. 10, pp. 252–259.

Patterson J.H., Słowiński R., Talbot F.B. and Węglarz J., 1989, “An algorithm for a general class of precedence and resource constrained scheduling problems”. *Advances in Project Scheduling*, Elsevier: Amsterdam, pp. 3–28.

Scheduling and Routing Workers Teams for Ground Handling at Airports with Column Generation

Giacomo Dall'Olio[1], Rainer Kolisch[1]

TUM School of Management, Technical University of Munich, Germany
giacomo.dallolio, rainer.kolisch@tum.de

Keywords: Ground Handling Scheduling, Airport Opertations, Column Generation, Vehicle Routing.

1 Introduction

Air traffic has been constantly increasing over the past decades, and its annual growth for the next ten years is estimated to 4.6% (Boeing Commercial Airplanes 2019). An efficient management of the airport infrastructures is therefore crucial in order to avoid congestions and delays which are causes for high costs and customer dissatisfaction. Poor planning of ground handling is one of the main sources of delays (Oreschko *et. al.* 2011). Ground handling consists of those services which are necessary to prepare the aircraft for its next flight and are performed at the gates or at parking positions. Such services include baggage loading and unloading, interior cleaning of the aircraft and refueling. Aircrafts are kept on the ground for a limited amount of time, which causes ground handling tasks to have restricted time windows within they can be performed. It is desirable to get the ground handling tasks done as soon as possible, to make sure that the aircrafts are ready before the scheduled take-off time. Since ground handling tasks, from now on simply denoted as tasks, are interdependent, any delay could propagate to other tasks. Missing the due date of a task might lead to a flight delay, which translates to penalty costs and reduced quality service for the ground handler. Specialized workforce, the ground personell, is responsible for performing the tasks. Each ground worker has a qualification level, which allows her/him to perform tasks with a requirement equal or lower to her/his own. The planner has to assign the workers to the tasks according to their qualification level, and schedule the tasks avoiding workforce shortage and meeting the due dates.

In this paper, we propose a solution method, for the mentioned problem, based on the branch and price framework, where column generation is used to find a lower bound.

2 Problem Definition

Planning ground handling is a combination of routing, assignment and scheduling problems. The tasks are performed by teams of workers. The workers are grouped into teams making sure they have an adequate qualification to perform the assigned tasks. The qualifications are definded as hierarchical skill levels. Workers can perform a task of a certain level only if their skill level is equal or higher. Since the tasks are located at different parking positions, we have to plan a route for the workers, so that they are present at the locations of the tasks in time to carry them out. A schedule for all the tasks has to be found, so that the tasks are performed as soon as possible.

Some of the tasks can be performed in more than one execution mode. An execution mode defines the number of workers needed to carry out the task in a certain amount of time. Modes which require more workers to perform a task also require less time. Teams can only perform tasks which entail a mode requiring a number of workers equal to the number of members of the team. In order to avoid complex synchronizing interactions, the workers leave the depot in teams, reach one or more task locations at which they perform the corresponding tasks and return back to the depot. Teams are not fixed for the whole time horizon, since the workers are free to form new ones as they come back to the depot.

Let us define $\mathcal{I}$ as the set of tasks and $\mathcal{K}$ as the set of possible skill levels. Each task i has to be performed within its time window $[ES_i, LF_i]$. The set $\mathcal{M}_i = \left\{m_i^{min}, ..., m_i^{max}\right\}$ represents the different modes in which task i can be peformed. When task i is performed in a certain mode, the number of workers needed corresponds to m_i, while $p_{i,m}$ is the corresponding execution time; notice that $p_{i,m} > p_{i,m+1}$. The earliest finish time is therefore$EF_i = ES_i + p_{i,m^{\max}}$. We define a tour r as the sequence of tasks carried out by a team composed by f_r members with skill levels equal or higher than q_r. During its tour, the team can perform tasks with a skill level requirement equal or lower than q_r which entails an execution mode equal to f_r. The tour r specifies the start time S_i^r and end time F_i^r for each performed task. A feasible tour r must therefore be compliant with the following:

$$\begin{aligned} S_i^r &\geq ES_i \\ F_i^r &\leq LF_i \\ S_i^r + p_{i,m} &= F_i \\ S_j^r &\geq F_i^r + d_{i,j} \end{aligned}$$

where i and j represent two consecutive tasks in the tour sequence and $d_{i,j}$ is the time needed to go from i to j. Since our goal is to complete the tasks as soon as possible, we introduce a penalty for each scheduled task, that is the difference between its earliest finish time and its actual finish time. We can therefore define the cost c^r of a tour r as

$$c^r = \sum_{i \in \mathcal{I}_r} (F_i^r - EF_i) \tag{1}$$

where $\mathcal{I}_r$ is the set of tasks performed during the tour. Supposing we can generate all possible team tours, we can write down the following formulation:

$$\begin{aligned} \textbf{min} \quad & \sum_{k=1}^{K} \sum_{r \in \Omega_k} c_k^r \lambda_k^r && && (2) \\ \textbf{s.t.} \quad & \sum_{r \in \Omega_k} a_{k,i}^r \lambda_k^r \geq 1 && \forall k \in \mathcal{K}, \forall i \in \mathcal{I} && (3) \\ & \sum_{k'=k}^{K} \sum_{r \in \Omega_{k'}} b_{k',t}^r \lambda_{k'}^r \leq \sum_{k=k'}^{K} N_{k'} && \forall k \in \mathcal{K}, \forall t \in \mathcal{T} && (4) \\ & \lambda_k^r \in [0,1] && \forall k \in \mathcal{K}, \forall r \in \Omega_k && (5) \end{aligned}$$

The tours are grouped by skill level k in order to simplify the notation. The set of tours of skill level k is Ω_k and λ_k^r is the binary variable which is 1 if tour r of skill level k is selected in the solution, 0 otherwise. The parameter $a_{k,i}^r$ is equal to 1 if the team from tour r performs task i, 0 otherwise. The parameter $b_{k,t}^r$ is equal to the number f of team members (which need to own a skill level of at least k) for those instants t when the team is operating, 0 otherwise. The overall number of available workers of skill level k is denoted as N_k. Constraint (3) enforces that every task is performed. Constraint (4) ensures that the number of workers is not exceeded in any time instant.

3 Literature Review

Given its strategic importance, ground handling has been considerably investigated in the literature. Nevertheless, not many publications tackle the problem from a combined scheduling and routing point of view. In Dohn *et. al.* (2009) teams are fixed before they are routed across the tasks. There is no schedule time optimization since the focus is to maximize the number of tasks performed. Fink *et. al.* (2019) focus on the Abstract VRP with Workers and Vehicle Synchronization (AVRPWVS), which they apply to ground handling. This problem, however, does not include any kind of qualifications or skills, which are necessary in such a setting. In the AVRPWVS, workers need to be synchronized in time and space at the task locations. This dramatically increases the complexity, making it hard to solve real-world instances. Dohn *et. al.* (2009) as well as Fink *et. al.* (2019) use a column generation approach, which is known to have good performances in solving vehicle routing problems with time windows (see Desrochers *et. al.* (1992)). Manpower allocation with hierarchical skill levels has been investigated, on a general level, by Bellenguez-Morineau and Néron (2007). Practical applications can be found in Cordeau *et. al.* (2010) and Firat and Hurkens (2012). In these papers, the travel time needed to move from the location of a task to another one is neglected, differently from our problem setting. The multi-mode RCPSP has been solved to optimality by Sprecher and Drexl (1998). In Hartmann and Briskorn (2010) a survey on the topic can be found.

4 Proposed Solution Approach

We propose the use of column generation to find lower bounds, and branch and price to find the optimal integer solution. We define the *continuous master problem* (MP) as the linear relaxation of the model proposed in Section 2. The value of an optimal solution of the MP is therefore a lower bound for the original problem. Furthermore, we introduce the *restricted master problem* (RMP), which has exactly the same structure of the MP, but is defined over a subset of tours $\Psi \subset \Omega$. Column generation consists of an iterative process where RMP is solved and the values of the dual variables are used to generate new promising tours. A new tour can improve the current RMP solution only if its reduced cost is negative. If it possibile to generate a new feasible tour with a negative reduced cost, the tour is added to Ψ and a new iteration of the column generation starts. Otherwise, the current solution of the RMP cannot be improved, therefore it is optimal for the MP and its value is a valid lower bound for the original problem. The reduced cost of a tour r with f team members working at level k is the following:

$$c_k^r - \sum_{i \in \mathcal{I}} a_{k,i}^r \mu_{k,i} + \sum_{t \in \mathcal{T}} b_{k,t}^r \delta_{k,t} \tag{6}$$

where μ and δ are respectively the values of the dual variables corresponding to constraints (3) and (4). The reduced cost of a team tour can be interpreted as follows. For each task performed during the tour, a penalty has to be paid if the end time is subsequent to the earliest finish time (c_k^r). The first summation is a reward obtained for every performed task while the second summation is a penalty paid for using limited resources (i.e. workers) at specific instants in time. The pricing problem is the problem of finding a tour of minmum reduced cost. Since a tour has a predefined number of team members f who work at a maximum skill level q, we have to solve the pricing problem multiple times with different settings. When solving a pricing problem for a team of f workers working at level q, only the tasks involved are those which entail an execution mode with f workers and whose skill level requirement equal or less than q. We can model the pricing problem with a time expanded network, where we have two types of nodes for each task: *start nodes* and *end nodes*. For each suitable task i, the network encompasses a start node for each possible start time of i, and a leave node for each instant from EF_i until the end of the time horizon. Each start node has one outgoing *execution arc* connecting it to an end node according to the execution time. The weight of an execution arc from node (i, t_S) to (i, t_F) corresponds to the reward for performing task i, plus the penalty for ending i at t_F (if any) and the penalty for keeping f workers busy from t_S to t_F. An end node (i, t_i) is connected to a start node (j, t_j) with a *travel arc* if $i \neq j$ and $t_j = t_i + d_{i,j}$. Travel arcs also connect the source node to all start nodes and all end nodes to the sink node. Start and sink nodes represent respectivly the leaving from and the returning to the depot. The weights of the travel arcs represent the penalty for keeping the workers busy from t_i to t_j. If the origin of the arc is the source node, $t_i = t_j - d_{\text{depot},i}$ while if if the destination of the arc is the sink node, $t_j = t_i + d_{i,\text{depot}}$. Eventually, two consecutive end nodes (i, t_i) and $(i, t_i + 1)$ referring to the same task i are connected with a *waiting arc*. The weight of a waiting arc corresponds to the penalty for keeping workers busy, therefore it follows the rule for travel arcs. Given the described network, the pricing problem can be solved finding a shortest path from the source node to the sink node.

5 Experimental Study

In order to verify the quality of our approach we will test the proposed algorithm on data from a major European airport. Based on these data, we generated various realistic test instances. The instances cover from 30 minutes up to 4 hours of a working day. Since the flights are not equally distributed during the day, we differentiate the instances in *low*, *medium* and *high* workload. The final results of the experimental study will be presented in the conference.

References

Bellenguez-Morineau, O., and Néron, E. , 2007, “A branch-and-bound method for solving multi-skill project scheduling problem", *RAIRO-operations Research*, Vol. 41(2), 155-170.

Boeing Commercial Airplanes, 2019, *Current Market Outlook 2019-2038*, Internet.

Cordeau, J. F., Laporte, G., Pasin, F., and Ropke, S., 2010, “Scheduling technicians and tasks in a telecommunications company.", *Journal of Scheduling*, Vol. 13(4), 393-409.

Desrochers, M., Desrosiers, J., and Solomon, M., 1992, “A new optimization algorithm for the vehicle routing problem with time windows.", *Operations research*, Vol. 40(2), 342-354.

Dohn, A., Kolind, E. and Clausen, J, 2009, “The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach.", *Computers & Operations Research*, Vol. 36(4), 1145-1157.

Frey, M., Desaulniers, G., Kiermaier, F., Kolisch, R., and Soumis, 2019, “Column generation for vehicle routing problems with multiple synchronization constraints.", *European Journal of Operational Research*, Vol. 272(2), 699-711.

Firat, M., and Hurkens, C. A. J., 2012, “An improved MIP-based approach for a multi-skill workforce scheduling problem.", *Journal of Scheduling*, Vol. 15(3), 363-380.

Hartmann S. and Briskorn D., 2010, “A survey of variants and extensions of the resource-constrained project scheduling problem.", *European Journal of Operational Research*, Vol. 207(1), 1-14.

Oreschko, B., Schultz, M., and Fricke, H., 2011, “Skill analysis of ground handling staff and delay impacts for turnaround modeling", *Air Transp. Oper*, pp. 310-318.

Sprecher A. and Drexl A., 1998, “Solving multi-mode resource-constrained project scheduling problems by a simple, general and powerful sequencing algorithm.", *European Journal of Operational Research*, Vol. 107(2), 431-450.

Robust scheduling for target tracking with wireless sensor network considering spatial uncertainty

Florian Delavernhe[1], André Rossi[3], Marc Sevaux[4]

[1] Université d'Angers, LERIA, F-49045 Angers, France
florian.delavernhe@univ-angers.fr
[2] Université Paris-Dauphine, LAMSADE, UMR 7243, CNRS, F-75016 Paris, France
andre.rossi@dauphine.psl.eu
[3] Université Bretagne Sud, Lab-STICC, UMR 6285, CNRS, F-56321 Lorient, France
marc.sevaux@univ-ubs.fr

1 Introduction

A wireless sensor network (WSN) is a set of sensors, randomly deployed in an area often hard or dangerous to access and without any infrastructure. Hence, the batteries of the sensors are not refillable which limits the lifetime of the network, *i.e.*, how long it can operate. There are several types of sensors for different applications, and we focus in this work on the target tracking. In such applications, the network aims to monitor a set of moving targets (planes, trains, terrestrial vehicles,...), whose spatial trajectories are estimated. It means that, at instant t in the time horizon, we have an estimation of the position of each target. However, this estimation may not be accurate, and the difficulty of the problem is to cover the targets considering the highest possible deviation from their estimated trajectories. Moreover, in order to preserve energy in the network for future mission, at most one sensor per target should be used at any time. Finally, all the data collected by the sensors have to be transmitted to a base station. The problem is to find a robust schedule to continuously monitor the targets and to transfer the data. This schedule is robust because it has to maximize a spatial stability radius, such that, it stays feasible as long as the targets are not deviated for more than the value of the stability radius from their estimated position. The targets are covered at every instant t as long as, they are located in the disc of radius equals to the stability radius and centered on the estimated position of the target. In this work, we propose (i) a discretization method on the geometric data, (ii) two upper bounds on the value of the stability radius, and (iii) a method that uses the discretized data and the upper bounds to compute a robust schedule.

2 Definition of the problem

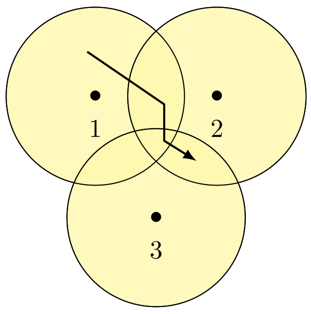
Let J be the set of the n targets that should be monitored. Each target j has an estimated trajectory such that at instant t, the estimated position of j is

$P_j(t)$. For each target, its estimated trajectory is represented using a collection of waypoints. The trajectory is a sequence of segments between the waypoints. *i.e.*, a trajectory is a suite of segments. The network is a set I of m sensors and a base station where the data is sent. A sensor can receive or transmit data only with the base station or another sensor if it is in its neighborhood $N(i)$, *i.e.*, if the distance is less than the communication range R_C. For a target j and a instant t, we define $\rho_j(t, R)$ as the set of all points that are in the disc of radius R and centered on $P_j(t)$. There are three types of energy consumption for a sensor:

– monitoring a target (p^S Watts),
– transmitting data (p^T Watts),
– receiving data (p^R Watts).

3 Discretization

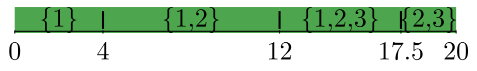
The following figure is an example, where three sensors (1,2 and 3) are deployed to cover a single target (the black arrow) in the horizon of time $H = [0, 20]$:



Discretization is the necessary transformation of the geometric data of the problem into a set of discretized data that can be used for modeling and solving the problem. The aim is to represent the trajectory of each target as a set of time windows with a set of candidate sensors associated to each window, that can monitor the target during the entire time window. Let's call a face f a set of spatial points that are covered by the same set of sensors $S(f)$. Monitoring a target j at time t is therefore monitoring all the faces where j can possibly be. Hence, for a stability radius R, we need to cover all the faces with a non empty intersection with $\rho_j(t, R)$. The intersection of all these faces defines the face to cover (if the intersection is empty, then the target cannot be covered). For example, if a target needs to be covered in the face $\{1\}$ and the face $\{1, 2\}$, the set of candidate sensors is $\{1\} \cap \{1, 2\} = \{1\}$.

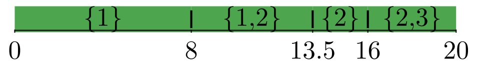
Thus, the trajectory of a target j is represented as a sequence of faces to be covered, associated to the set K_j of time windows. The time windows are delimited by time instants called ticks, such that a tick is either *entering* which means that a new sensor is candidate, or *leaving* when a sensor is no more candidate.

With a $R = 0$, the time windows and candidate sensors over the horizon of time, in our example, are:



When increasing the stability radius, the spatial uncertainty covered is increasing. It delays the instant where a sensor is guaranteed to cover a target and is advancing the moment where a sensor is no more candidate to cover the target. The evolution of a tick depends on the segment of the estimated spatial trajectory where it is located. Therefore, increasing the stability radius corresponds to moving the ticks and is modifying the length of the time windows. This may change the set of candidate sensors when two ticks are equal.

In our example, with a certain value of R, the tick moved and changed the candidate sensors such that we obtain:



Another difficulty is that, for each target j, there are instants t where time window can appear when the stability radius reaches a certain value r. Indeed, it possible that some points in $\rho_j(t, r)$ are no longer covered by a sensor that was initially covering $P_j(t)$ even if the corresponding tick did not move. It means that a new time window is appearing in K_j at t when the stability radius reaches this specific value of R. In our example, a time window is appearing at the time corresponding to the last estimated waypoint if the stability radius is great enough to allow the target to be out of the range of sensor 2.

In order to find all these time windows, we need to look at each intersection of the segments of the estimated trajectory of a same target, *i.e.*, when the target is changing of direction. For each sensor covering an extremity e of a segment, there is a potential time window w. It appears at the time instant the target is estimated to be at e, when the stability radius is equal to the sensing range of the sensor minus the distance between the sensor and e. Indeed, a segment is always leaving the range of a sensor starting by one of its extremity or by the estimated frontier between two faces (initial ticks).

To conclude, an increase of the stability radius is modifying the length of the time windows, adding new time windows, and adding new or changing the sets of candidate sensor. All of this issues are depending on the coordinates of the sensors and the segments of the trajectories.

4 Upper bounds on the stability radius

Two upper bounds were found and implemented for our solving method. The first bound searches the first value of R that creates an empty face to cover. The stability radius cannot exceed this value and it always corresponds to either the intersection between two ticks, or the apparition of a new time window. First, we need to compute, for each extremity, the last time window that will appear. Because each of the time window is corresponding to one sensor no longer candidate for this extremity, the last time window is corresponding to an empty face. These values are easily computed with the distance between the extremity and the sensors. The second part of this bound is, for each segment, to find the lowest value of R that corresponds to an intersection between two ticks such that the intersection point is no longer covered by any other sensor when the intersection occurs. These values are computed using the position of the sensors and the segments' coordinates.

The second bound computes the value of R such that there is not enough energy in a set of candidate sensors to cover the length of the corresponding time window. In that purpose, for each face in the estimated trajectories of the targets, we compute the total sum of the batteries of the sensors in range and the total energy needed. We order these sensors by increasing values of R for which they are no longer in range of any point of the trajectories in the face. Afterwards, we remove the batteries of these sensors, one by one, until the sum of the batteries of the remaining sensors are not enough to cover the face.

5 Solving Process

Because the time windows and the corresponding sets of candidate sensors are depending on the value of the stability radius, a single linear program cannot be solved to maximize R. Therefore, we use a dichotomy method on the values of R which modify these sets. For each value tested by the dichotomy, the following satisfactory linear program is solved:

$$\sum_{j\in J}\sum_{k\in K_j|i\in S_j(k)} x_{jik}\, p^S + p^R \sum_{i'\in N(i)} f_{i'i} + p^T \sum_{i'\in N(i)} f_{ii'} \leq E_i \quad \forall i \in I \qquad (1)$$

$$\sum_{j\in J}\sum_{k\in K_j|i\in S_j(k)} x_{jik} + \sum_{i'\in N(i)} f_{i'i} - \sum_{i'\in N(i)} f_{ii'} = 0 \quad \forall i \in I \qquad (2)$$

$$\sum_{i\in S_j(k)} x_{jik} = \Delta_k^j \quad \forall j \in J, k \in K_j \qquad (3)$$

$$\delta \geq 0 \qquad (4)$$

$$x_{jik} \geq 0 \quad \forall j \in J, k \in K_j, i \in S_j(k) \qquad (5)$$

$$f_{ii'} \geq 0 \quad \forall i \in I, i' \in N(i) \qquad (6)$$

With Δ_k^j the size of the k-th time window of the sensor j and x_{jik} the time sensori i is monitoring j in its k-th time window. Constraints (1) correspond to the limitations of the batteries. Constraints (2) are flow constraints, where the data collected and received by a sensor is transmitted. Constraints (3) sets that the sum of the activities of the sensors in a time window is equal to its length.

References

1. Lersteau, C., Rossi, A., Sevaux, M. (2018). Minimum energy target tracking with coverage guarantee in wireless sensor networks. European Journal of Operational Research, 265(3), 882-894.

Exact solution of the two-machine flow shop problem with 3 operations

Federico Della Croce[1], Fabio Salassa[1] and Vincent T'kindt[2]

[1] Politecnico di Torino, Italy
[federico.dellacroce,fabio.salassa]@polito.it
[2] University of Tours, Laboratory of Theoretical and Applied Computing (EA 6300), ERL CNRS 7002 ROOT, Tours, France
tkindt@univ-tours.fr

Keywords: Two-machine flow shop with three operations, ILP modeling, Exact approach.

1 Introduction

We consider a two-machine flow shop problem with three operations originally proposed in (Gupta et al. 2004). There is a set of n jobs being available at time zero to be processed on a two-machine flow-shop. Each job i has three operations, where the first operation has processing time a_i and must be performed on the first machine. The third operation has processing time b_i and must be performed on the second machine. Finally, the second operation has processing time c_i and can be performed either on machine 1 immediately after the first operation or on machine 2 immediately before the third operation. The operations of the same job cannot be processed concurrently, nor can any machine process more than one job at a time. We assume that preemption is not allowed, i.e., any operation once started must be completed without interruption. The goal is to minimize the makespan denoted by $C_{\max}$. As mentioned in (Gupta et al. 2004), this problem applies to several situations where a machine-independent setup operation is needed on each job between the two operations. The setup time is job-dependent and both machines are equipped with the required tooling for the setup. Then, the setup of an individual job is performed either while the job is still mounted on the first machine after the completion of the first operation or once the job is mounted on the second machine before the start of the second operation. The problem has strong similarities with the two-machine flow shop problem with common due date and jobs selection considered in (T'kindt et al. 2007) and (Della Croce et al. 2017). By using the extended three-field notation of (T'kindt, Billaut 2006) this latter problem is denoted by $F2|d_i = d, \, unknown \ d \, |\epsilon(d/n_T)$ where the number of jobs $n - n_T$ to be selected (here n_T is the number of tardy, hence discarded, jobs) is given in advance and the aim is to find the minimum value of d. For problem $F2|d_i = d, \, unknown \ d \, |\epsilon(d/n_T)$, the best available exact approach is able to solve very large size instances in limited CPU time (less than 30 seconds in the worst case for instances with $n = 100000$).

From every instance of the original 3-operation two-machine flow shop problem, it is possible to generate a special $F2|d_i = d, \, unknown \ d \, |\epsilon(d/n_T)$ problem as follows. Every job i of the original problem induces two "coupled" incompatible jobs i_1 and i_2 of the jobs selection problem where i_1 has the second operation of i assigned to the first machine, while i_2 has the second operation of i assigned to the second machine. Correspondingly, job i_1 has processing times $\alpha_{i_1} = a_i + c_i$ and $\beta_{i_1} = b_i$, while job i_2 has processing times $\alpha_{i_2} = a_i$ and $\beta_{i_2} = b_i + c_i$. Thus, we reduce to a two-machine flow shop problem with $2n$ jobs where exactly n compatible jobs out of the $2n$ jobs have to be selected.

2 ILP formulation

Consider the $2n$ jobs generated from the original problem as indicated above with processing times α_i (β_i), $1 \leq i \leq 2n$, on machine M_1 (M_2). When the set Ω of selected jobs is fixed, the minimization of the makespan for these jobs can be done in polynomial time by the so-called Johnson's algorithm (Johnson 1954): schedule first the jobs with $\alpha_i \leq \beta_i$ in non-decreasing order of α_i, followed by the jobs with $\alpha_i > \beta_i$ in non-increasing order of β_i. Without loss of generality, let us assume that the $2n$ jobs are indexed according to their position in the Johnson's schedule.

Let d be the unknown common due date, or equivalently the makespan of the selected jobs. Let us associate to each job i a binary variable x_i that indicates if job i is selected or not. A first ILP model is as follows.

$$\min d \tag{1}$$

$$\alpha_1 x_1 + \sum_{i=1}^{2n} \beta_i x_i \leq d \tag{2}$$

$$\sum_{i=1}^{2n} \alpha_i x_i + \beta_{2n} x_{2n} \leq d \tag{3}$$

$$\sum_{i=1}^{j} \alpha_i x_i + \sum_{i=j}^{2n} \beta_i x_i \leq d, \quad \forall j = 2, ..., 2n-1 \tag{4}$$

$$x_i + x_k = 1 \ \ \forall i, k \ \ \textit{incompatible} \tag{5}$$

$$x_i \in \{0, 1\} \ \ \forall i \in 1, ..., n \tag{6}$$

Here, constraints (2–4) are critical-path constraints which define the value of d. Notice that d is always determined by the sum of the processing times of jobs $1, .., j$ on the first machine plus the sum of the processing times of jobs $j, .., 2n$ on the second machine where j depends on the selected early jobs and therefore constraints (2–4) consider all possible values of j with $1 \leq j \leq 2n$. Notice that in the critical path constraints (2–4), we explicited constraint (2) corresponding to $j = 1$ and constraint (3) corresponding to $j = 2n$. Constraints (5) represent the incompatibility constraints between each pair of coupled jobs so that there will be exactly n early (selected) jobs. Finally, constraints (6) indicate that the x_i variables are binary.

Due to the presence of constraints (4) that generate $O(n^2)$ nonzeroes in the constraints matrix, the above model is limited in size as it induces an out-of-memory status of the solver if problems with several thousands of variables are considered.

As mentioned in (Della Croce et al. 2017), there exists an equivalent ILP formulation with $O(n)$ nonzeroes in the constraints matrix that can be obtained by introducing variables $y_j = \sum_{i=1}^{j} \alpha_i x_i + \sum_{i=j}^{2n} \beta_i x_i$ and constraints $y_i = y_{i-1} - \beta_{i-1} x_{i-1} + \alpha_i x_i, \ \ \forall i \in 2, .., 2n$.

$$\min d \tag{7}$$

$$y_1 = \alpha_1 x_1 + \sum_{i=1}^{2n} \beta_i x_i \tag{8}$$

$$y_{2n} = \sum_{i=1}^{2n} \alpha_i x_i + \beta_{2n} x_{2n} \tag{9}$$

$$y_i = y_{i-1} - \beta_{i-1} x_{i-1} + \alpha_i x_i \ \ \forall i = 2, ..., 2n-1 \tag{10}$$

$$y_i \leq d, \ \ \forall i = 1, ..., 2n \tag{11}$$

$$x_i + x_k = 1, \ \ \forall i, k \ \ incompatible \tag{12}$$

$$x_i \in \{0, 1\} \ \ \forall i \in 1, ..., n, \ \ y_i \geq 0 \ \ \forall i \in 1, ..., n \tag{13}$$

Let us denote by ILP_c the above ILP. Interestingly enough, the addition of the incompatibility constraints makes the problem much more difficult both for CPLEX 12.9 solver applied to ILP_c and to the constraint generation approach of (Della Croce et al. 2017) adapted in order to incorporate the incompatibility constraints. We tested both solution approaches on a Computer Intel i5 @1.6 GHz and 8 G of RAM. We considered a standard distribution of processing times with a_i, b_i and c_i uniformly distributed in the range [1...100] and tested 10 distinct instances for each problem size considering a CPU time limit of 60 seconds per instance. With this distribution, CPLEX 12.9 solver applied to model (7–13) already failed to solve to optimality one instance with 200 jobs, while the constraint generation approach of (Della Croce et al. 2017) was limited to 600 jobs runnning out of time on one instance with 700 jobs. We remark however that constraints (8–9) in ILP_c can be modified as follows where $\alpha_{[\min_2]}$ and $\beta_{[\min_2]}$ indicate the second smallest processing time on the first and the second machine respectively.

$$y_1 = \alpha_1 x_1 + \alpha_{[\min_2]}(1 - x_1) + \sum_{i=1}^{2n} \beta_i x_i \tag{14}$$

$$y_{2n} = \sum_{i=1}^{2n} \alpha_i x_i + \beta_{[\min_2]}(1 - x_{2n}) + \beta_{2n} x_{2n} \tag{15}$$

Indeed, if $x_1 = 1$, then constraint (14) coincides with constraint (8), while if $x_1 = 0$, then the critical path on the first selected job has processing time on the first machine not inferior to $a_{[\min_2]}$. Similar consideration holds with respect to constraint (15) taking into account the critical path on the last selected job and its processing time on the second machine. At the time of the conference we will also discuss how $a_{[\min_2]}$ and $b_{[\min_2]}$ can be increased without loss of optimality. In the reminder we denote by ILP_{ic} the improved ILP formulation that substitutes in ILP_c constraints (8–9) with constraints (14–15).

Hence, we can then successfully adapt to our problem the constraint generation approach proposed in (Della Croce et al. 2017) according to the scheme depicted in Algorithm 1. There, we denote by $F2^{3op}$ our problem formulated according to the ILP_{ic} model and by $F2^{3op}_{rel}$ its relaxation induced by the elimination of constraints (10) and considering constraints (11) only for $i = 1, 2n$.

Algorithm 1 Contraint Generation Algorithm

```
1: End=$False$
2: while !End do
3:     Solve $F2^{3op}_{rel}$: $\bar{x}$ is its solution and $OPT(F2^{3op}_{rel})$ its value
4:     Compute $d(\bar{x})$ the optimal value of the ILP of $F2^{3op}$ with added constraints $x = \bar{x}$
5:     if $(d(\bar{x}) = OPT(F2_{rel}))$ then
6:         End=$True$
7:     else
8:         Let $\mathcal{C}$ be the constraint giving $d(\bar{x})$ in the ILP of $F2^{3op}$ for $\bar{x}$
   //      ($\mathcal{C}$ is the most violated constraint)
9:         Add $\mathcal{C}$ to $F2^{3op}_{rel}$
10:    end if
11: end while
12: return $\bar{x}$ as the optimal solution of $F2^{3op}$
```

Algorithm 1 is a constraint generation approach solving initially problem $F2^{3op}_{rel}$ and then considering a separation procedure adding to the relaxation any inequality of the original formulation that is violated by the current solution. We tested both CPLEX 12.9 solver applied to model ILP_{ic} and Algorithm 1 on instances generated according to the same distribution considered above (10 instances for each problem size). The related results are depicted in Table 1 where it is shown that Algorithm 1 is clearly superior and solves within 60 seconds instances with up $n = 30000$. Detailed computational results on several other different distributions will be presented at the conference.

n	CPLEX		Algorithm 1	
	t_{avg}	t_{max}	t_{avg}	t_{max}
1000	0.6	1	0.1	1
4000	9.3	13	0.6	1
7000	24.1	36	1.2	2
10000	CPU limit		3.7	5
15000	CPU limit		6.2	9
20000	CPU limit		10.1	13
25000	CPU limit		24.3	35
30000	CPU limit		25.9	43
35000	CPU limit		CPU limit	

Table 1. Comparing CPLEX applied to model ILP_{ic} and Algorithm 1

Acknowledgements

This work has been partially supported by "Ministero dell'Istruzione, dell'Università e della Ricerca" Award "TESUN-83486178370409 finanziamento dipartimenti di eccellenza CAP. 1694 TIT. 232 ART. 6".

References

Della Croce, F., Koulamas, C., T'kindt, V.: A constraint generation approach for two-machine shop problems with jobs selection. *European Journal of Operational Research*, 259: 3, 898–905 (2017).

Gupta, J.N.D., Koulamas, C.P., Kyparisis, G.J., Potts,C.N.,Strusevich, V.A.: Scheduling Three-Operation Jobs in a Two-Machine Flow Shop to Minimize Makespan. *Annals of Operations Research*, 129: 1-4, 171–185 (2004).

Johnson, S.: Optimal two and three stage production schedules with set-up time included. *Naval Research Logistics Quarterly*, 1, 61–68 (1954).

Tkindt, V. , Billaut, J.-C.: Multicriteria scheduling: Theory, models and algorithms (2nd edition). Heidelberg, Germany: Springer-Verlag (2006).

T'kindt, V., Della Croce, F., Bouquard, J.L.: Enumeration of Pareto Optima for a Flowshop Scheduling Problem with Two Criteria. *INFORMS Journal on Computing*, 19: 1, 64–72 (2007)

Adversarial bilevel scheduling on a single machine

Della Croce F.[1] and T'kindt V.[2]

[1] DIGEP, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy, CNR, IEIIT, Torino, Italy.
federico.dellacroce_a@polito.it
[2] Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée (EA 6300), ERL CNRS 7002 ROOT, Tours, France,
and
DIGEP, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy,
tkindt@univ-tours.fr

Keywords: Single machine, bilevel optimization.

1 Introduction

In this contribution we focus on a particular setting in which two agents are concerned by the scheduling of a set of n jobs. The first agent, called the *leader*, can take some decisions before providing the jobset to the second agent, called the *follower*, who then takes the remaining decisions to solve the problem. As an example, the leader could select a subset of $n' \leq n$ jobs that the follower has to schedule. Notice that the decisions the agents can take are exclusive: in this example, the follower cannot decide the jobs to schedule and the leader cannot schedule the jobs. This setting falls into the category of *bilevel optimization* (Dempe et al. 2015). In such problems it is assumed that the leader and the follower follow their own objectives which can be contradictory, so leading to very hard optimization problems. Recently, many papers on bilevel combinatorial optimization appeared, here we refer to (Caprara et al. 2016, Della Croce et al. 2019, Fischetti et al. 2017, Fischetti et al. 2018, Fischetti et al. 2019) just to mention a few. On the other hand, to the authors knowledge, the literature on bilevel scheduling is much more limited. We refer here to (Abass 2005, Karlof and Wangs 1996, Kis and Kovacs 2012). We focus in the following on single machine scheduling under the adversarial framework where the goal of the leader is to make the follower solution as bad as possible and provide several exact polynomial time algorithms for different objective functions when the leader can only modify data of the problem.

2 Adversarial bilevel single machine scheduling

2.1 Sum of completion times

It is assumed that, given a list of n jobs with processing times p_j^F, the follower is scheduling jobs so that their sum of completion times, denoted by $\sum_j C_j^F$, is minimum. This is doable in polynomial time by applying the so-called SPT rule (*Shortest Processing Times* first). Let be the initial processing times p_j so that $p_1 \leq ... \leq p_n$. Then, the leader has to decide how to fix quantities q_j so that with $p_j^F = p_j + q_j$, the follower optimal solution is the worst possible. Obviously, it is of no interest for the leader that some $q_j < 0$. In addition, the leader has a budget so that $\sum_j |q_j| \leq Q$, with $Q \in \mathbb{N}$ given. This problem is referred to as $1|ADV-p|\sum_j C_j^F$, with $ADV-p$ meaning that it concerns an adversarial bilevel problem in which the leader can only modify the processing time values.

Theorem 1. *The* $1|ADV-p|\sum_j C_j^F$ *problem can be solved in* $O(n\log(n))$ *time. The leader sets:*

- $q_j = P - p_j$, $\forall j = 1..\big(k_P - Q - k_P P + \sum_{i=1}^{k_P} p_i\big)$,
- $q_j = P - p_j + 1$, $\forall j = \big(k_P - Q - k_P P + \sum_{i=1}^{k_P} p_i\big)..k_P$,
- $q_j = 0$, $\forall j = k_P + 1..n$,

with $P = argmax_{0 \le t \le \sum_j p_j}\Big((kt - \sum_{j=1}^{k} p_j) \le Q | p_1 \le ... \le p_k \le t \text{ and } p_{k+1} > t\Big)$*, and* k_P *the job such that* $p_{k_P} \le P < p_{k_P+1}$*. The follower applies the SPT rule on the* $p_j^F = p_j + q_j$*'s.*

2.2 Weighted sum of completion times

Now, let us assume that, in addition to the previous problem, jobs are also attached weights w_j^F and the follower is scheduling jobs so that their weighted sum of completion times, denoted by $\sum_j w_j^F C_j^F$, is minimum. Whenever the processing times are fixed, this is doable in polynomial time by applying the so-called WSPT rule (*Weighted Shortest Processing Times* first). Let be the initial processing times p_j so that $\frac{p_1}{w_1^F} \le ... \le \frac{p_n}{w_n^F}$. Again, the leader has to decide how to fix quantities q_j so that with $p_j^F = p_j + q_j$, the follower optimal solution is as worse as possible. Obviously, it is of no interest for the leader that some $q_j < 0$. This problem is referred to as $1|ADV - p|\sum_j w_j^F C_j^F$.

We first consider the relaxed version where $q_j \in \mathbb{R}, \forall j = 1..n$, denoted by $1|ADV - p, q_j \in \mathbb{R}|\sum_j w_j^F C_j^F$.

Theorem 2. *The* $1|ADV - p, q_j \in \mathbb{R}|\sum_j w_j^F C_j^F$ *problem can be solved in* $O(n\log(n))$ *time. The leader sets:*

- $q_j = \frac{(Q + \sum_{\ell=1}^{k_R} p_\ell) w_j^F}{\sum_{\ell=1}^{k_R} w_\ell^F} - p_j$, $\forall j = 1..k_R$
- $q_j = 0$, $\forall j = (k_R + 1)..n$,

with $R = \frac{Q - \sum_{j=1}^{k_R} p_j}{\sum_{j=1}^{k_R} w_j^F}$ *and* k_R *the job such that* $\frac{p_{k_R}}{w_{k_R}^F} \le R < \frac{p_{k_R+1}}{w_{k_R+1}^F}$*. The follower applies the WSPT rule on* $p_j^F = p_j + q_j$ *and* $w_j^F = w_j$, $\forall j = 1..n$.

The optimal solution of the $1|ADV - p|\sum_j w_j^F C_j^F$ problem can be obtained by solving iteratively the relaxed version: first solve it with the initial Q value and round down the computed q_j's. Then, on the remaining quantity $Q' = (Q - \sum_j q_j)$ solve again the relaxed problem to modify processing times. This process is iterated until all initial budget Q is assigned to jobs. As there are at most n iterations, this leads to an exact algorithm than can be implemented in $O(n^2)$ time.

Let us turn to the other possible adversarial problem in which the leader can only modify the weights of the follower. So, for the follower's problem we set $p_j^F = p_j$ and $w_j^F = w_j + q_j, \forall j = 1..n$, with $q_j \in \mathbb{N}$. This problem is referred to as $1|ADV - w|\sum_j w_j^F C_j^F$ and as previously, $1|ADV - w, q_j \in \mathbb{R}|\sum_j w_j^F C_j^F$ refers to the relaxed version with real valued q_j's.

Theorem 3. *The* $1|ADV - w, q_j \in \mathbb{R}|\sum_j w_j^F C_j^F$ *problem can be solved in* $O(n\log(n))$ *time. The leader sets:*

- $q_j = \frac{(Q + \sum_{\ell=k_R}^{n} w_\ell) p_j^F}{\sum_{\ell=k_R}^{n} p_\ell^F} - w_j$, $\forall j = k_R..n$
- $q_j = 0$, $\forall j = 1..(k_R - 1)$,

with $R = \frac{\sum_{j=k_R}^{n} p_j^F}{Q+\sum_{j=k_R}^{n} w_j}$ *and* k_R *the job such that* $\frac{p_{k_R-1}^F}{w_{k_R-1}} < R \leq \frac{p_{k_R}^F}{w_{k_R}}$. *The follower applies the WSPT rule on* $p_j^F = p_j$ *and* $w_j^F = w_j + q_j$, $\forall j = 1.n$.

The $1|ADV-w|\sum_j w_j^F C_j^F$ problem can be solved by iteratively solving the relaxation with real valued q_j's to dispatch the initial leader's budget Q. Again, this leads to an $O(n^2)$ optimal algorithm.

2.3 Maximum lateness

Assume that each job j is defined by a processing time p_j and a due date d_j. The aim, for the follower, is to schedule jobs so as to minimize the maximum lateness, defined by $L_{max}^F = \max_{j=1..n}(C_j^F - d_j^F)$. The leader can modify either the processing times or the due dates. Without loss of generality, let us assume that $d_1 \leq ... \leq d_n$.

We first focus on the problem where the leader can only modify the processing times, which is referred to as $1|ADV-p|L_{max}^F$. As the due dates remain unchanged, we set $d_j^F = d_j$, $\forall j = 1..n$. Besides, $p_j^F = p_j + q_j$ is the processing time value of the follower's problem. It is trivial to show that $q_j \in \mathbb{N}$ in order to make increasing the optimal solution value of the follower's problem. Besides, it is known that the $1||L_{max}$ problem is solved to optimality by the EDD rule (Earliest Due Dates first). So the follower builds the optimal sequence by sorting jobs by non decreasing values of the d_j^F's which is not impacted by any variations in the processing time values. Consequently, the $1|ADV-p|L_{max}^F$ problem can be solved in $O(n\log(n))$ time by sorting jobs according to EDD rule and then set:

- $q_k = Q$ with k the earliest job having $(C_k - d_k) = L_{max}^*$ and L_{max}^* the L_{max} value of the EDD schedule,
- $q_j = 0$, $\forall j = 1..n$, $j \neq k$.

Let us consider the problem in which the leader can only modify the due dates, which is referred to as $1|ADV-d|L_{max}^F$. Then, we set $p_j^F = p_j$ and $d_j^F = d_j + q_j$, $\forall j = 1..n$.

Theorem 4. *The* $1|ADV-d|L_{max}^F$ *problem can be solved in* $O(n\log(n))$ *time. The leader sets:*

- $q_\ell = D - d_\ell \leq 0$, $\forall \ell \in \cup_{j\in\mathcal{T}} B_j \cup \mathcal{T}$,
- *and* $q_\ell = 0$, *otherwise.*

with:

- $\mathcal{T} = \{j / C_j^F - d_j^F = L_{max}^*\}$, *with* L_{max}^* *the value of the initial EDD sequence,*
- $B_j = \{k < j | \nexists \ell \in \mathcal{T}, \text{ with } k < \ell < j\}$, $\forall j \in \mathcal{T}$,
- $\alpha_\ell = (d_\ell - d_j) \leq 0$ *and* $[\ell]$ *is the* ℓ*-th* α_u *value when sorted by non decreasing values, i.e.* $\alpha_{[1]} \leq ... \leq \alpha_{[n']}$ *with* $n' = |\cup_{j\in\mathcal{T}} B_j|$,
- k *such that* $(|\mathcal{T}|+k)\alpha_{[k]} - \sum_{\ell=1}^{k}\alpha_{[\ell]} \leq Q \leq (|\mathcal{T}|+k+1)\alpha_{[k+1]} - \sum_{\ell=1}^{k+1}\alpha_{[\ell]}$,
- *and* $D = \lfloor \frac{Q+\sum_{\ell=1}^{k}\alpha_{[\ell]}}{|\mathcal{T}|+k} \rfloor$.

The follower applies the EDD rule on $p_j^F = p_j$ *and* $d_j^F = d_j + q_j$, $\forall j = 1.n$.

Acknowledgements

This work has been partially supported by "Ministero dell'Istruzione, dell'Università e della Ricerca" Award "TESUN-83486178370409 finanziamento dipartimenti di eccellenza CAP. 1694 TIT. 232 ART. 6".

References

Abass S.A.: Bilevel programming approach applied to the flow shop scheduling problem under fuzziness. Computational Management Science. 2: 279–293 (2005)

Caprara, A., Carvalho, M., Lodi, A., Woeginger, G.: Bilevel Knapsack with Interdiction Constraints. INFORMS Journal on Computing. 28, 319–333 (2016)

Della Croce, F., Scatamacchia, R.: Lower Bounds and a New Exact Approach for the Bilevel Knapsack with Interdiction Constraints. In: Lodi A., Nagarajan V. (eds) Integer Programming and Combinatorial Optimization. IPCO 2019. Lecture Notes in Computer Science, vol 11480, 155–167. Springer International Publishing (2019)

Dempe, S., Kalashnikov, V. Perez-Valdes, G.A., Kalashnikova, N., 2015, "Bilevel programming problems", *Springer*.

Fischetti, M., Ljubić, I., Monaci, M., Sinnl, M.: Interdiction Games and Monotonicity, with Application to Knapsack Problems. INFORMS Journal on Computing. 31, 390–410 (2019)

Fischetti, M., Ljubić, I., Monaci, M., Sinnl, M.: A New General-Purpose Algorithm for Mixed-Integer Bilevel Linear Programs. Operations Research. 65, 1615–1637 (2017)

Fischetti, M., Ljubić, I., Monaci, M., Sinnl, M.: On the use of intersection cuts for bilevel optimization. Mathematical Programming. 172, 77–103 (2018)

J.K. Karlof, J.K., Wang, W.: Bilevel programming applied to the flow shop scheduling problem. Computers and Operations Research. 23:5, 443–451 (1996).

Kis, T., Kovacs, A.: On bilevel machine scheduling problems. OR Spectrum. 34, 43–68 (2012)

A Conjunctive-disjunctive Graph Modeling Approach for Job-Shop Scheduling Problem with Changing Modes

[1]Xavier Delorme, [2]Gérard Fleury, [2]Philippe Lacomme, [3]Damien Lamy

[1] Mines Saint-Etienne, Univ Clermont Auvergne, CNRS, UMR 6158 LIMOS,
Institut Henri Fayol, F - 42023 Saint-Etienne France
delorme@emse.fr

[2] Université Clermont Auvergne,
Campus des Cézeaux, 1 rue de la Chebarde
TSA 60125 - CS 60026
63178 Aubière cedex France
placomme@isima.fr, gefleury@isima.fr

[3] Mines Saint-Etienne, Institut Henri Fayol, F - 42023 Saint-Etienne, France
damien.lamy@emse.fr

Keywords: Scheduling, Reconfiguration, Job-shop, Conjunctive-Disjunctive Graph.

1. Introduction

Reconfigurable Manufacturing Systems (RMS) have been defined in (Mehrabi et al., 2000) as an effective approach to deal with unpredictable and high-frequency market changes that are facing industries. To cope with such changes, the production systems must be adaptive and able to evolve in order to consider 1) changes in parts of existing products; 2) fluctuations in demands; 3) evolution in legal regulations and 4) evolution in process technology. Meanwhile, the scheduled operations remain partially manual like material handling, carrying and processing jobs as stressed by (Napolitano, 2012). The assignment of operators to operations must include personal skills, training and experience in order to match the competences and/or functionalities required by the operations to be performed (Ferjani et al., 2017; Grosse et al., 2015). In RMS, the sequential execution of operations may depends on the job operation sequence that can refer to Flow-shop, Job-shop, etc. Including flexibility for processing operations remains possible at each step of the job-sequence. Meanwhile, reconfigurability is the capacity of a set of machines to be reconfigured in a period of time, which can be seen as setup times. Machine activation delay may include cleaning the working zone, loading, positioning and unloading the parts (jobs) and can imply costs coming from energy expenditures, equipment maintenance and labor as stressed by (Borgia et al., 2013). Hence, in RMS a solution is composed by a set of configurations applied sequentially and thus a sequence-dependent processing time of operations and sequence dependent setup times have to be considered in such a production system. If sequence depend setup times are features of several research projects in the scheduling community as stressed in (Sharma and Jain, 2016; Shen et al., 2018), these works generally consider setup times at the operation level, whereas several modifications of the system may occur in RMS requiring several resources to be inactive during reconfigurations.

Hence, the problem addressed in this research project is different from the one introduced in (Essafi et al., 2012) since it does not encompass design and line balancing but only machine operations, and is concerned with makespan minimization and not minimization of the cost of the line. Actually, the problem is closer to the former vision provided by (Liles and Huff, 1990) who first indicated the necessity to schedule efficiently operations in reconfigurable manufacturing environments. As stressed by (Azab and Naderi, 2015) very few papers deal with scheduling in RMS. In their research work, they addressed reconfigurations in the context of Flow-shop production systems, but they did not investigate graph modelling.

The present paper is dedicated to scheduling in reconfigurable manufacturing systems where operators assignment to machine allows to define several modes meaning that processing time of operations is varying according to chosen configurations. The work specifically focuses on the graph modelling of the problem in the context of a Job-shop-like production system and introduces encoding and decoding of solutions.

2. Graph modelling and representation of solutions

The problem under study is stated as a reconfigurable job-shop manufacturing system where a set J of n jobs has to be scheduled $J = \{J_1, J_2 \dots J_n\}$ on a set $M = \{M_1, \dots, M_m\}$ of m machines. Each job in J consists in a set of operations $O_j = \{O_{1j}, \dots, O_{mj}\}$. The whole system operates under configurations. Moving from a configuration to another may affect specific machines, resulting in variations in processing times of operations. Hence, each operation O_{ij} has a processing time P_{ij}^k where k denotes the chosen configuration. Configuration differs from setup times, since transition between configurations can affect several machines and configurations can be activated only when these machines are inactive. Considering two configurations k_1 and k_2, identifying machines that are concerned by a switch from configuration k_1 to k_2 is achieved through vectors $R_{k_1 k_2}^{M_u}$, where each value of the vector is valued 0 if the machine M_u is not concerned with transition, 1 otherwise. A reconfiguration time $T_{k1,k2}$ is required when switching from a configuration k_1 to k_2. The objective is to schedule efficiently operations and to define configuration assignments in order to minimize the completion time of all operations (makespan). In the following, the data bellow are considered, where $M_u(P_{ij}^k)$ denotes the processing time on machine M_u according to configurations.

Table 1. processing times of operations in configuration 1

Product	O_{1j}	O_{2j}	O_{3j}
$j = 1$	$M_1(10)$	$M_2(6)$	$M_3(17)$
$j = 2$	$M_2(15)$	$M_1(10)$	$M_3(20)$
$j = 3$	$M_3(4)$	$M_2(10)$	$M_1(20)$

Table 2. processing times of operations in configuration 2

Product	O_{1j}	O_{2j}	O_{3j}
$j = 1$	$M_1(13)$	$M_2(4)$	$M_3(12)$
$j = 2$	$M_2(12)$	$M_1(17)$	$M_3(23)$
$j = 3$	$M_3(7)$	$M_2(16)$	$M_1(10)$

Table 3. Definition of $R_{k_1 k_2}^{M_u}$

Configurations	k_1	k_2
k_1		(1;1;1)
k_2	(1;1;1)	

Tables 1, 2 and 3 introduce data of a 3 jobs, 3 machines Job-shop Scheduling Problem, where processing times of operations depend on configurations. As can be seen in Table 1 and 2, processing times of operations on machine M_1 are different whether configurations k_1 or k_2 are selected. Assignment of machines when switching from a configuration to another is introduced in Table 3. In this problem, all machines are affected by a change in configuration, and hence, they must be all inactive when switching from a configuration to another and without of generality the reconfiguration time $T_{k1,k2}$ is set to 1 time unit.

In scheduling problems it is classical to use a conjunctive-disjunctive graph approach that have been proved to be efficient by (Roy and Sussmann, 1964). For the incumbent problem, a conjunctive-disjunctive graph $G(V, A, E)$ is considered where V corresponds to the operations, A denotes the arcs and E defines the edges. Initial arcs correspond to precedencies in jobs sequence of operations (i.e. an arc (O_{ij}, O_{ij+1}) exists in G between two successive operations of i). E refers to edges that have to be oriented and initially contains edges relevant to operations that have to be processed on the same machines, and all edges that refer to configurations. An edge is considered between operations O_{ij} and O_{kj} if they can be processed in two different configurations that are impacting processing times of both operations. Similarly to (Dauzère-Pérès and Paulli, 1997) for the Flexible Job-shop, different shape lines can connect operations in order to distinguish configuration switches and machine disjunctions. The objective is to assign a configuration to each operation and to defined edges that connect operations using the same machine. The Figure 1 gives an example of a conjunctive-disjunctive graph after choosing configurations for operations.

For sake of clarity, two graphs are presented in Figure 1, the first one (A) concerns edges related to machine disjunctions (dashed lines), and the second one (B) displays edges related to configuration switches (dotted lines). In this figure, operations modeled with grey nodes are processed into configuration 1, and the ones with white nodes are processed into configuration 2. As operations of a given job are ordered, edges connecting two operations with different assigned configurations can be removed (*i.e.* edge between $(M_1; M_3)$ on job J_1 is useless) when other operation are present between them.

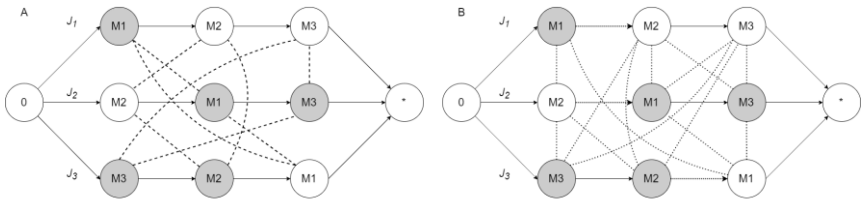


Figure 1. Graph A with edges for machine disjunctions and graph B with configuration disjunctions.

Modeling solutions is an important preliminary step before defining complex operators such as metaheuristics or local search. Indirect representations are widely spread in literature for scheduling problems (Cheng et al., 1996). For the incumbent problem two vectors are used. The first one (R) is a vector by repetition (Bierwirth et al., 1996) which is an ordered list of job numbers (a job numbers is in the list m times with m the number of machines) and each occurrence of a job corresponds to one of its operations. The second vector (C) is the configuration vector which is a list of configurations under which operations are processed. Both vectors represent a solution which is an orientation of all arcs (Fig. 2) considering $R = [1;2;2;3;3;1;2;1;3]$ and $C = [1;2;1;1;1;2;1;2;2]$. Considering these vectors, defining a solution consists in reading the vectors from left to right applying an extension of the Bierwith vector rules for graph generation. Figure 2 shows the evaluated graph after execution of one longest path algorithm.

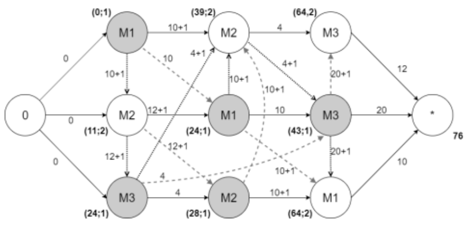


Figure 2. Evaluated conjunctive graph

In Figure 2, dashed arrows define sequence of operations on machines, while dotted arrows define reconfiguration switches. Each arc modeling reconfiguration switches are valued ($P_{ij}^{k} + T_{kk'}$). Starting time and configuration of operations are in bold in the figure. According to the vector R, the first operation scheduled is the first operation of job J_1 and according to the vector C, it is processed with configuration 1, hence its processing time is 10 according to Table 1. The second operation in vector R is the first of job J_2 processed on M_2 with configuration 2, hence a reconfiguration switch occurs after operation O_{11} that required a 1 time unit of reconfiguration that delay the operation O_{12} starting time at 11. The third scheduled operation is O_{22} on machine M_1 and configuration 1 and a reconfiguration occurs after O_{12}, and O_{22} will start at 24 (ending time of O_{12} plus reconfiguration time). The fourth scheduled operation is O_{13} that is the first operation on M_3, also processed with configuration 1, and hence, it starts after the last operation that affected M_3, with configuration 2. This process iterates until the end of both vectors R and C. The obtained Gantt chart is given in Figure 3.

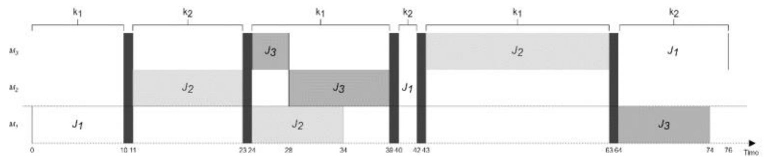


Figure 3. Gantt chart corresponding to evaluated Graph

As stressed on Figure 3, 5 reconfigurations are operated along the time horizon and they are respectively scheduled at times [10;11], [23;24],[39;40],[42;43] and [63;64]. As all machines are affected by these reconfigurations, it is not possible to schedule operations earlier, considering the given vectors R and C.

The Gantt of figure 3 does not define an optimal solution and could be further improved by local search operator for example. Future research is now directed on the design of a metaheuristic that will consider the encoding vectors including specific operator such as construction heuristics, neighborhoods and local search operators. An effective local search approach should rely on an exploration of the critical paths that must create operator on the vector by changing order of operations in vector R, changing configurations in vector C, or both.

3. Conclusion

This work is at the corner stone of both scheduling and reconfigurable manufacturing systems communities since reconfigurations and setup times are very similar notions that are closed to the flexible terminology used in scheduling. In this research project, the Job-shop is extended with reconfiguration schemes. When a reconfiguration occurs, specific machines are affected and have to be stopped in order to apply the new configuration to the production system. It is possible to address small-scale instances using linear solvers but medium and large-scale instances remain intractable. The use of metaheuristics seems appropriate and will concern the upcoming research prospects. To this purpose, a conjunctive-disjunctive graph model is proposed. Adjoined with proper representation of solutions it is possible to map an element from the coding space with the proposed graph model through a decoding procedure. Two vectors are used to represent orientations of arcs and selection of configurations in the graph. In addition with metaheuristics, local search procedures relying on critical path exploration are currently investigated.

References

Azab, A., Naderi, B., 2015. Modelling the Problem of Production Scheduling for Reconfigurable Manufacturing Systems. Procedia CIRP 33, 76–80.

Bierwirth, C., Mattfeld, D.C., Kopfer, H., 1996. On permutation representations for scheduling problems. In: International Conference on Parallel Problem Solving from Nature. Springer, pp. 310–318.

Borgia, S., Matta, A., Tolio, T., 2013. STEP-NC compliant approach for setup planning problem on multiple fixture pallets. Journal of Manufacturing Systems 32, 781–791.

Cheng, R., Gen, M., Tsujimura, Y., 1996. A tutorial survey of job-shop scheduling problems using genetic algorithms - I. Representation. Computers & Industrial Engineering 30, 983–997.

Dauzère-Pérès, S., Paulli, J., 1997. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Annals of Operations Research 70, 281–306.

Essafi, M., Delorme, X., Dolgui, A., 2012. A reactive GRASP and Path Relinking for balancing reconfigurable transfer lines. International Journal of Production Research 50, 5213–5238.

Ferjani, A., Ammar, A., Pierreval, H., Elkosantini, S., 2017. A simulation-optimization based heuristic for the online assignment of multi-skilled workers subjected to fatigue in manufacturing systems. Computers & Industrial Engineering 112, 663–674.

Grosse, E.H., Glock, C.H., Jaber, M.Y., Neumann, W.P., 2015. Incorporating human factors in order picking planning models: framework and research opportunities. International Journal of Production Research 53, 695–717.

Liles, D.H., Huff, B.L., 1990. A computer based production scheduling architecture suitable for driving a reconfigurable manufacturing system. Computers & Industrial Engineering 19, 1–5.

Mehrabi, M.G., Ulsoy, A.G., Koren, Y., 2000. Reconfigurable manufacturing systems: Key to future manufacturing. Journal of Intelligent Manufacturing 11, 403–419.

Roy, B., Sussmann, B., 1964. Les problemes d'ordonnancement avec contraintes disjonctives. SEMA, Rapport de recherche n°9.

Sharma, P., Jain, A., 2016. A review on job shop scheduling with setup times. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 230, 517–533.

Shen, L., Dauzère-Pérès, S., Neufeld, J.S., 2018. Solving the flexible job shop scheduling problem with sequence-dependent setup times. European Journal of Operational Research 265, 503–516.

Near-Linear Approximation Algorithms for Scheduling Problems with Setup Times

Max Deppert[1] and Klaus Jansen[1]

University of Kiel, Germany
made,kj@informatik.uni-kiel.de

Keywords: Scheduling Theory, Approximation Algorithms, Setup Times.

1 Problem Definition

Scheduling problems with setup times have been intensively studied for over 30 years now; in fact, they allow very natural formulations of scheduling problems.

In the general scheduling problem with setup times, there are m identical and parallel machines, a set J of $n \in \mathbb{N}$ jobs $j \in J$, $c \in \mathbb{N}$ different classes, a partition $\dot{\bigcup}_{i=1}^{c} C_i = J$ of c nonempty and disjoint subsets $C_i \subseteq J$, a *processing time* of $t_j \in \mathbb{N}$ time units for each job $j \in J$ and a *setup* (or *setup time*) of $s_i \in \mathbb{N}$ time units for each class $i \in [c]$. The objective is to find a schedule which minimizes the makespan while holding the following. All jobs (or its complete sets of job pieces) are scheduled. Whenever a machine switches processing from one job to another, a setup may be necessary. There are various types of setups discussed; here we focus on *sequence-independent* batch setups, i.e. a setup only gets necessary when switching from one class of jobs to another *different* class on a machine and it does not depend on the previous job/class. All machines are *single-threaded* (jobs (or job pieces) and setups do not intersect in time on each machine) and no setup is preempted. There are three variants of scheduling problems with setup times which have been gaining the most attention in the past. There is the *non-preemptive* case where no job may be preempted, formally known as problem $\mathrm{P|setup{=}s_i|C_{max}}$. Another variant is the *preemptive* context, namely $\mathrm{P|pmtn,setup{=}s_i|C_{max}}$, where a job may be preempted at any time but be processed on at most one machine at a time, so a job may not be parallelized. In the generous case of *splittable* scheduling, known as $\mathrm{P|split,setup{=}s_i|C_{max}}$, a job is allowed to be split into any number of job pieces which may be processed on any machine at any time.

2 Related results

Monma and Potts (1989) began their investigation of these problems considering the preemptive case. They found first dynamic programming approaches for various single machine problems polynomial in n but exponential in c. Furthermore, they showed NP-hardness for $\mathrm{P|pmtn,setup{=}s_i|C_{max}}$ even if $m = 2$. In a later work Monma and Potts (1993) found a heuristic which resembles McNaughton's preemptive wrap-around rule; see also (McNaughton 1959). It requires $\mathcal{O}(n)$ time for being $(2 - (\lfloor \frac{m}{2} + 1 \rfloor)^{-1})$-approximate. Notice that this ratio is truly greater than $\frac{3}{2}$ if $m \geq 4$ and the asymptotic bound is 2 for $m \to \infty$. Monma and Potts also discussed the problem class of *small batches* where for any batch i the sum of one setup time and the total processing time of all jobs in i is smaller than the optimal makespan, i.e. $s_i + \sum_{j \in C_i} t_j \leq OPT$. Most suitable for this kind of problems, they found a heuristic that first uses list scheduling for complete batches followed by an attempt of splitting some batches so that they are scheduled on two different machines. This second approach needs a running time of $\mathcal{O}(n + (m+c)\log(m+c))$ and considering only small batches it is $(\frac{3}{2} - \frac{1}{4m-4})$-approximate if $m \leq 4$ whereas it is

$(\frac{5}{3} - \frac{1}{m})$-approximate for small batches if m is a multiple of 3 and $m \geq 6$. Then Chen (1993) modified the second approach of Monma and Potts. For small batches Chen improved the heuristic to a worst case guarantee of $\max\{\frac{3m}{2m+1}, \frac{3m-4}{2m-2}\}$ if $m \geq 5$ while the same time of $\mathcal{O}(n + (m + c)\log(m + c))$ is required.

Schuurman and Woeginger (1999) studied the preemptive problem for *single-job-batches*, i.e. $|C_i| = 1$. They found a PTAS for the uniform setups problem $s_i = s$. Furthermore, they presented a $(\frac{4}{3} + \varepsilon)$-approximation in case of arbitrary setup times. Both algorithms have a running time linear in n but exponential in $1/\varepsilon$. Then Xing and Zhang (2000) turned to the splittable case. Without other restrictions they presented an FPTAS if m is fixed and a $\frac{5}{3}$-approximation in polynomial time if m is variable. They give some simple arguments that the problem is weakly NP-hard if m is fixed and NP-hard in the strong sense otherwise. More recently Mäcker *et. al.* (2015) made progress to the case of non-preemptive scheduling. They used the restrictions that all setup times are equal $(s_i = s)$ and the total processing time of each class is bounded by γOPT for some constant γ, i.e. $\sum_{j\in C_i} t_j \leq \gamma OPT$. Mäcker et al. found a simple 2-approximation, an FPTAS for fixed m, and a $(1 + \varepsilon)\min\{\frac{3}{2}OPT, OPT + t_{\max} - 1\}$-approximation (where $t_{\max} = \max_{j\in J} t_j$) in polynomial time if m is variable. Jansen and Land (2016) found three different algorithms for the non-preemptive context without restrictions. They presented an approximation ratio 3 using a next-fit strategy running in time $\mathcal{O}(n)$, a 2-dual approximation running in time $\mathcal{O}(n)$ which leads to a $(2 + \varepsilon)$-approximation running in time $\mathcal{O}(n\log(\frac{1}{\varepsilon}))$, as well as a PTAS. Recently Jansen *et. al.* (2019) found an EPTAS for all three problem variants. For the preemptive case they assume $|C_i| = 1$. They make use of n-fold integer programs, which can be solved using the algorithm by Hemmecke, Onn, and Romanchuk. However, even after some runtime improvement the runtime for the splittable model is $2^{\mathcal{O}(1/\varepsilon^2 \log^3(1/\varepsilon))} n^2 \log^3(nm)$, for example. These algorithms are interesting answers to the question of complexity but they are useless for solving actual problems in practice. Therefore the design of *fast* (and especially polynomial) approximation algorithms with small approximation ratio remains interesting.

3 New Results

For all three problem variants we give a 2-approximate algorithm running in time $\mathcal{O}(n)$ as well as a $(\frac{3}{2} + \varepsilon)$-approximation with running time $\mathcal{O}(n\log(\frac{1}{\varepsilon}))$. With some runtime improvements we present some very efficient near-linear approximation algorithms with a constant approximation ratio equal to $\frac{3}{2}$. In detail, we find a $\frac{3}{2}$-approximation for the splittable case with running time $\mathcal{O}(n+c\log(c+m)) \leq \mathcal{O}(n\log(c+m))$. Also we will see a $\frac{3}{2}$-approximate algorithm for the non-preemptive case that runs in time $\mathcal{O}(n\log(T_{\min}))$ where $T_{\min} = \max\{\frac{1}{m}N, \max_{i\in[c]}(s_i + t^{(i)}_{\max})\}$, $t^{(i)}_{\max} = \max_{j\in C_i} t_j$ and $N = \sum_{i=1}^{c} s_i + \sum_{j\in J} t_j$. For the most complicated case of these three problem contexts, the preemptive case, we study a $\frac{3}{2}$-approximation running in time $\mathcal{O}(n\log(c + m)) \leq \mathcal{O}(n\log n)$. For the long version we refer to (Deppert and Jansen 2018). Especially the last result is interesting; we make progress to the general case where classes may consist of an *arbitrary* number of jobs. The best approximation ratio was the one by Monma and Potts (1993) mentioned above. All other previously known results for preemptive scheduling used restrictions like *small batches* or even *single-job-batches*, i.e. $|C_i| = 1$. As a byproduct we give some new *dual* lower bounds.

References

B. Chen. A better heuristic for preemptive parallel machine scheduling with batch setup times. *SIAM Journal on Computing*, 22(6):1303–1318, 1993.

M. A. Deppert and K. Jansen. Near-linear approximation algorithms for scheduling problems with batch setup times. *CoRR*, abs/1810.01223, 2018.

K. Jansen, K. Klein, M. Maack, and M. Rau. Empowering the configuration-ip - new PTAS results for scheduling with setups times. In A. Blum, editor, *Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124 of *LIPIcs*, pages 44:1–44:19, 2019.

K. Jansen and F. Land. Non-preemptive scheduling with setup times: A PTAS. In P. Dutot and D. Trystram, editors, *European Conference on Parallel and Distributed Computing (Euro-Par 2016)*, volume 9833 of *LNCS*, pages 159–170. Springer, 2016.

A. Mäcker, M. Malatyali, F. Meyer auf der Heide, and S. Riechers. Non-preemptive scheduling on machines with setup times. In F. Dehne, J. Sack, and U. Stege, editors, *Symposium on Algorithms and Data Structures (WADS 2015)*, volume 9214 of *LNCS*, pages 542–553. Springer, 2015.

R. McNaughton. Scheduling with deadlines and loss functions. *Management Science*, 6(1):1–12, Oct. 1959.

C. L. Monma and C. N. Potts. On the complexity of scheduling with batch setup times. *Operations Research*, 37(5):798–804, 1989.

C. L. Monma and C. N. Potts. Analysis of heuristics for preemptive parallel machine scheduling with batch setup times. *Operations Research*, 41(5):981–993, 1993.

P. Schuurman and G. J. Woeginger. Preemptive scheduling with job-dependent setup times. In R. E. Tarjan and T. J. Warnow, editors, *Symposium on Discrete Algorithms (SODA 1999)*, pages 759–767.

W. Xing and J. Zhang. Parallel machine scheduling with splitting jobs. *Discrete Applied Mathematics*, 103(1-3):259–269, 2000.

Efficiency and Equity in the Multiple Organization Scheduling Problem

Martin Durand, Fanny Pascual

Sorbonne Université, CNRS, LIP6, 4 place Jussieu, 75005 Paris, France
martin.durand@lip6.fr,fanny.pascual@lip6.fr

Keywords: multi agents scheduling, fairness, makespan minimization.

1 Introduction

The Multi Organization Scheduling Problem (MOSP) (Pascual *et al.* 2007) deals a set of n organizations $\{O_1, \dots, O_n\}$ which each owns both a set of identical parallel machines, and a set of sequential tasks to execute. The objective is to minimize the completion time of the last task completed on the machines shared by the organizations (the makespan), under an additional constraint: no organization should increase the last completion time of its tasks in the shared system, compared to the case where it executes its own tasks on its own machines. This last constraint is called the *rationality constraint*, and ensures that all the organizations have incentive to share their machines. More formally, let us denote by C^i_{loc} is the makespan of Organization O_i if it schedules its own tasks on its owns machines - this scheduled is assumed to be given by the organization (it can minimizes the makespan of Organization O_i ,or not) - and is called the *local makespan of* O_i. Given any schedule S of all the tasks on all the machines, we denote by $C_i(\mathcal{S})$ the makespan of O_i, i.e. the maximum completion time of a task of O_i in S. Our problem is the following one:

$$\text{minimize } C_{\max}(\mathcal{S}) \text{ such that, for each } i \in \{1, \dots, n\}, C_i(\mathcal{S}) \leq C^i_{loc}.$$

Interest of cooperation. Let us first show that sharing machines does not only allow organizations which have many tasks and few machines to decrease theirs makespans by given tasks to the other organizations, but that each organization may decrease its makespan. Let us consider the following instance, in which all organizations can benefit from cooperating.

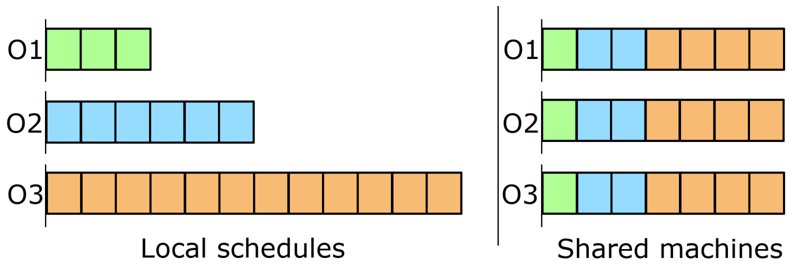


Fig. 1. An example in which each organization benefits from cooperating

There are $n = 3$ organizations, each one having only one machine. All the tasks are of length 1. Organization O_1 owns 3 tasks, O_2 owns 6 tasks, and O_3 owns 12 tasks. On figure 1 we can see on the left the local schedules (schedules in which each organization schedules its own tasks on its owns machines), and on the right a schedule in which the organizations share their machines. In this example, sharing the machines allow each organization to decrease its makespan. This example can be extended for higher values of n. We have shown that the best improvement which can be obtained for each organization simultaneously is a factor n, and that this ratio can be obtained on some instances.

Focus of this paper and map of the paper. Besides analyzing the best possible benefit that organizations can mutually have by sharing their machines, our aim is to focus on the *efficiency* of algorithms (where the efficiency is thought in term of makespan – the date at which all the tasks have been computed), and on the *equity* of algorithms for MOSP (it is not suitable that, even if the returned schedule fulfills the rationality constraint, the machines which are free are used only for the tasks of a single organization while some tasks of the other organizations are waiting). These two aspects may be antagonist, and our aim is to see to which extent, since what we want would be a schedule with a small makespan and in which machines are shared with equity.

In Section 2, we focus on efficiency: we analyze the highest possible increase of a local makespan which may occur if we want a schedule which minimizes the (global) makespan. We then look at the problem where each organization agrees to increases its makespan (compared to its local makespan) by a factor $(1 + \epsilon)$.

In Section 3, we focus on fairness: we introduce a new problem which consists in maximizing the minimal gain (decrease of its makespan) of an organization. Before presenting these results, we start by reviewing existing work on MOSP.

State of art. The Multi Organization Scheduling Problem (Pascual *et al.* 2007) has been introduced with parallel rigid tasks (tasks that need to be executed in parallel on several machines) and has mainly been studied from an approximation viewpoint. The best approximate algorithm is a 3-approximation algorithm when the organizations schedule locally the tasks in decreasing order of their heights (the height of a task is the number of machines needed to execute the task), or a 4-approximation algorithm in the general case (Dutot *et. al.* 2011). For sequential tasks (tasks that need to be executed on one machine only), the best known algorithm is a 2-approximate algorithm (Cohen *et. al.* 2010) (in the sequel, all the papers – as well as our results – deal with sequential tasks).

Some papers also consider a relaxed version of MOSP: it is assumed that the organizations tolerate a bounded degradation on the makespan of their own tasks, and the aim is to minimize the global makespan. This problem is denoted by $(1 + \alpha)$-MOSP (Ooshita *et. al.* 2009) when it is assumed that each organization accepts to increase the maximum completion time of its tasks by a factor at most $(1+\alpha)$. A $\frac{3}{2}$-approximate algorithm for 2-MOSP has been given (Cordeiro *et. al.* 2011). The closest work in spirit to what we will do in Section 2 is a study of $(1+\alpha)$-MOSP on unrelated machines (Ooshita *et. al.* 2009, Ooshita *et. al.* 2012). In this setting, Ooshita et al. show that, when there is no cooperation ($\alpha = 0$), the makespan can be m times higher than in the optimal makespan without the rationality constraint. When $\alpha > 0$, the authors also give a $(2 + \frac{2}{\alpha})$-approximate algorithm for $(1 + \alpha)$-MOSP.

2 Efficiency vs. increase of the local makespans

In this section, we study how the aim of minimizing the makespan is in opposition with the rationality constraint.

2.1 Necessary trade-off between the (global) makespan and the increase of local makespans.

We have shown that in an optimal schedule for the makespan minimization, an organization may increase its makespan up to a factor m, where m is the number of machines (due to lack of space the proof is omitted). This value, that could be called "the price of efficiency", is high. We will now assume that organizations may accept to increase their makespans in order to get an efficient schedule, but only if this does not increase to much their makespans. We will now assume that each organization agrees to increase a little bit its makespan: it will accept a schedule in which its makespan is increased by a factor at most $(1+\epsilon)$ compared to its local makespan.

Let $\epsilon \geq 0$. We assume that each organization O_i agrees to have a makespan at most equal to $(1+\epsilon)C^i_{loc}$. If $\epsilon = 0$, this is the MOSP. Otherwise, each organization agrees to increase a little bit its makespan (the higher α is, the higher an organization agrees to increase its makespan). We call $(1+\epsilon)$-MOSP, the problem where we wish to minimize the makespan with these relaxed constraints:

$$\text{minimize } C_{\max}(\mathcal{S}) \text{ such that, for each } i \in \{1,\ldots,N\}, C_i(\mathcal{S}) \leq (1+\epsilon)C^i_{loc}.$$

Thanks to a specific instance, we give a lower bound on the approximation ratio of any algorithm for $(1+\epsilon)$-MOSP with respect to the optimal makespan when there is no rationality constraints: this shows what we loose, in term of makespan, due to the relaxed rationality constraint. When $\epsilon = 0$, we obtain the following proposition.

Proposition 1 *There is no algorithm which returns schedules which fulfill the rationality constraint, and which is less than* 2*-approximate with respect to the global makespan.*

This bound improves the previous one, $\frac{3}{2}$, which had been given (Pascual *et al.* 2007) first for two organizations and then (Cohen *et. al.* 2010) for more than two organizations. Furthermore, in (Cohen *et al.* 2011) the authors show that no approximation algorithm for MOSP has a ratio asymptotically better than 2 w.r.t. the global makespan (when m tends towards the infinity) when we add the constraint that on the returned schedule, each machine schedules the tasks of its organization (if any) before the tasks of other organizations. This constraint is thus not necessary to obtain the asymptotic ratio of 2.

2.2 A PTAS for the makespan minimization with a bounded increase on the local makespan

We adapt the polynomial approximation scheme (PTAS) presented (Hall and Shmoys 1989) for a scheduling problem (makespan minimization with delivery times), to get a PTAS with resource augmentation for our problem. More precisely: given a fixed $\epsilon > 0$, and a fixed number of organizations n, we will get a polynomial time algorithm which returns a schedule with a makespan at most $(1+\epsilon)$ times the optimal makespan, and in which the makespan of each organization is at most $(1+\epsilon)$ times its local makespan. The rationality constraint may thus be violated, but the increase of the makespans of the organizations is bounded, and may be acceptable if ϵ is small.

In the previous sections, we have assumed either that the rationality constraint should be fulfilled (but we then had as only objective function to minimize the makespan, and the gains for the organizations – the decrease of their makespans – in the returned schedule could be very different), or we have even assumed than we can relax (in a bounded way) the rationality constraint to get a schedule with an even smaller makespan. In the following section, we focus on fairness issues: we will keep the rationality constraint, and our focus will not be to decrease the makespan, but to get schedule in which *all* the organizations decrease their makespans by a factor as large as possible.

3 Maximizing the Minimal Decrease of the Local Makespans

Given a schedule $\mathcal{S}$, the gain $g_i(\mathcal{S})$ of Organization O_i represents how much Organization O_i has decreased its makespan in $\mathcal{S}$ in comparison to its local schedule:

$$g_i(\mathcal{S}) = \frac{C^i_{loc}}{C_i(\mathcal{S})}.$$

The Maximal Minimal Gain problem, denoted as MaxMinGain, takes the same input as MOSP. It builds a schedule of all the tasks of all the organizations on the m machines of the organizations, in order to maximize the minimum gain among the organizations. The returned schedule is thus $\mathcal{S} = \arg\max_{\mathcal{S}} \min_{i \in \{1,\dots,N\}} g_i(\mathcal{S})$

Problem MaxMinGain can be solved in polynomial time when all the tasks have the same length. Moreover, in this case, it is possible to find a schedule $\mathcal{S}$ which is optimal for MaxMinGain and which is optimal for problem $(P||C_{\max})$: the global makespan is minimized while the minimal gain of an organization is maximized. The algorithm is very simple: it consists in scheduling the tasks greedily, by increasing local makespans.

When tasks can have different lengths, MaxMinGain is strongly NP-hard and hard to approximate. We also show that there is no algorithm which is optimal for MaxMinGain and which has an approximation smaller than 2 for MOSP. Naturally, this implies that no algorithm can be optimal for MaxMinGain and have an approximation ratio smaller than 2 for $(P||C_{\max})$.

As seen in the previous section, a list scheduling by increasing local makespan is optimal for MaxMinGain when tasks all have the same length. However, in the general case, such a schedule can break the rationality constraint. We complete our results with a heuristic that aims at returning a schedule as close as possible from a list schedule by increasing local makespan but respecting the rationality constraint. On tested instances, this heuristic returns a schedule with an average makespan below 1.07 times the optimal makespan and an average minimum gain above 0.92 times the optimal one.

References

Cohen J., D. Cordeiro, D. Trystram, F. Wagner, 2010, “Analysis of Multi-Organization Scheduling Algorithms", *Euro-Par*, Vol. 2, pp. 367-379.

Cohen J., D Cordeiro, D. Trystram, F. Wagner, 2011, “Multi-organization scheduling approximation algorithms", *Concurrency and Computation: Practice and Experience*, Vol. 23, pp. 2220-2234.

Cordeiro D., P. Dutot, G. Mounié, D. Trystram, 2011, “Tight Analysis of Relaxed Multi-organization Scheduling Algorithms", *IPDPS*, pp. 1177-1186.

Dutot P., F. Pascual, K. Rzadca, D. Trystram, 2011, “Approximation Algorithms for the Multi-organization Scheduling Problem", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, pp. 1888-1895.

Hall L. A., Shmoys D. B., 1989, “Approximation Schemes for Constrained Scheduling Problems", *FOCS*, pp. 134-139.

Ooshita F., T. Izumi, T. Izumi, 2009, “A Generalized Multi-Organization Scheduling on Unrelated Parallel Machines", *PDCAT*, pp. 26-33.

Ooshita F., T. Izumi, T. Izumi, 2012, “The Price of Multi-Organization Constraint in Unrelated Parallel Machine Scheduling", *Parallel Process Letters*, Vol. 22.

Pascual F., K. Rzadca, D. Trystram, 2011, “Cooperation in Multi-organization Scheduling", *Euro-Par*, pp. 224-233.

On the complexity of the crossdock truck-scheduling problem

Q. Fabry[1,3], A. Agnetis[2], L. Berghman[1] and C. Briand[3]

[1] Université de Toulouse - Toulouse Business School,
20 BD Lascrosses – BP 7010, 31068 Toulouse Cedex 7, France
`l.berghman@tbs-education.fr`
[2] Università degli Studi di Siena, DIISM, Siena, Italy
`agnetis@diism.unisi.it`
[3] LAAS-CNRS, Université de Toulouse, UPS, Toulouse, France
`{quentin.fabry,cyril.briand}@laas.fr`

Keywords: crossdocking, truck scheduling, complexity.

1 Introduction

Crossdocking is a warehouse management concept in which items delivered to a warehouse by inbound trucks are immediately sorted out, reorganized based on customer demands and loaded into outbound trucks for delivery to customers, without requiring excessive inventory at the warehouse (J. van Belle *et al.* 2012). If any item is held in storage, it is usually for a brief period of time that is generally less than 24 hours. Advantages of crossdocking can accrue from faster deliveries, lower inventory costs, and a reduction of the warehouse space requirement (U.M. Apte and S. Viswanathan 2000, N. Boysen *et al.* 2010). Compared to traditional warehousing, the storage as well as the length of the stay of a product in the warehouse is limited, which requires an appropriate coordination of inbound and outbound trucks (N. Boysen 2010, W. Yu and P.J. Egbelu 2008).

The crossdock truck-scheduling problem (CTSP), which decides on the succession of truck processing at the dock doors, is especially important to ensure a rapid turnover and on-time deliveries. The problem studied concerns the operational level: trucks are allocated to the different docks so as to minimize the storage usage during the product transfer. The internal organization of the warehouse (scanning, sorting, transporting) is not explicitly taken into consideration. We also do not model the resources that may be needed to load or unload the trucks, which implies the assumption that these resources are available in sufficient quantities to ensure the correct execution of an arbitrary docking schedule. In this abstract, we present some new complexity results that refer to a situation in which the number of docks (or doors) at the terminal is small, namely one or two. This situation has been indeed addressed in the literature, e.g. (A. Chiarello *et al.* 2018). However most authors focus on tardiness objectives, while we focus on minimizing overall soujourn time of the pallets, which is especially meaningful for perishable goods or for reducing stock holding costs.

This abstract is structured as follows: Section 2 formalizes the problem and introduces some basic notations, Section 3 addresses the complexity of the crossdocking truck scheduling problem in various scenarios (complexity proofs are not provided for the sake of conciseness), then a few concluding remarks are provided.

2 Detailed problem statement

We consider a crossdocking warehouse where *inbound trucks* $i \in I$ need to be unloaded and *outbound trucks* $o \in O$ need to be loaded (where I is the set of all inbound trucks and O is the set of all outbound trucks). The warehouse features n docks that can be used both for loading and unloading. The unloading and loading processing times of trucks $i \in I$ and $o \in O$ are referred to as p_i and p_o, respectively. Similarly, let W_i (respectively, W_o) denote the number of *pallets* to be unloaded from i (respectively, to be loaded on o). We let w_{io} denote the number of pallets that must be transferred from i to o. It is sometime convenient to visualize an instance of the problem through a bipartite graph $G_T = (I, O, P)$ called *transfer graph.* In G_T, the two node sets correspond to inbound and outbound trucks respectively, and there is an arc (i, o) if

$w_{io} > 0$. The arc set P expresses start-start precedence constraints, i.e., if $(i,o) \in P$, truck $o \in O$ cannot start being loaded before truck $i \in I$ starts being unloaded. In this paper we consider two scenarios:

(i) There is no relationship between the number of pallets that need to be loaded/unloaded and the processing time of a truck. In this case, for any two trucks h and k, in general $W_h/p_h \neq W_k/p_k$. We say that in this scenario processing times are *unrelated*;

(ii) The loading/unloading time of a truck is proportional to the number of pallets that must be loaded/unloaded. For simplicity, in this case we assume that the processing times are expressed in terms of number of pallets being moved, i.e.,

$$p_i = W_i = \sum_{o \in O} w_{io}, \forall i \in I \tag{1}$$

and

$$p_o = W_o = \sum_{i \in I} w_{io}, \forall o \in O \tag{2}$$

We say that in this scenario processing times are *correlated*. Notice that, in this case,

$$\sum_{o \in O} p_o = \sum_{i \in I} p_i. \tag{3}$$

It is assumed that there is sufficient workforce to load/unload all docked trucks at the same time. Hence, a truck assigned to a dock does not wait for the availability of a material handler.

Products can be transshipped directly from an inbound to an outbound truck if the outbound truck is placed at a dock. Otherwise, the products are temporarily stored and will be loaded later on. The problem is to determine time-consistent start times s_i and s_o of unload and load tasks $i \in I$ and $o \in O$ so as to minimize the *total time spent in the warehouse by all pallets* (total flow time). For each pallet which has to be transferred from i to o such a flow time equals $s_o - s_i$. Therefore, the total flow time is

$$\sum_{(i,o) \in P} w_{io}(s_o - s_i). \tag{4}$$

In what follows, $CTSP(n,U)$ denotes the problem with n gates and unrelated processing times, while $CTSP(n,C)$ denotes the problem with n gates and correlated processing times.

Due to (1) and (2), it is easy to show that problem $CTSP(n,C)$ consists in finding the feasible schedule that minimizes

$$\sum_{o \in O} p_o s_o - \sum_{i \in I} p_i s_i. \tag{5}$$

3 Complexity results

Let us first consider the problem CTSP when $n = 1$, i.e., the crossdocking platform has a single gate, and let us start with the special case in which the transfer graph G_T is complete, i.e., it is a "1-biclique" (Figure 1). This means that $w_{io} > 0$ for each $i \in I$ and $o \in O$, i.e., each inbound truck has at least one pallet that must be transferred to each outbound truck.

Let us consider the unrelated problem CTSP(1,U) in the "1-biclique" case. Since G_T is complete, in any feasible schedule *all* inbound trucks must be consecutively scheduled, before all outbound trucks. So, the problem consists of deciding in which order they should be scheduled. The following property holds.

Theorem 1. *When G_T is a biclique, $CTSP(1,U)$ is solved by first scheduling all inbound trucks in nonincreasing order of the ratio p_i/W_i, then all outbound trucks in nondecreasing order of the ratio p_o/W_o.*

$\square$

Note that such an optimal sequence can be obtained in $O(n \log n)$. Concerning problem $CTSP(1,C)$, recalling (1) and (2), Theorem 1 implies that, when G_T is a biclique, $CTSP(1,C)$ is solved by scheduling all inbound trucks before all outbound trucks, in any order. Theorem 1 easily extends to the case in which G_T consists of k disjoint bicliques (e.g., see Figure 2 with $k = 3$).

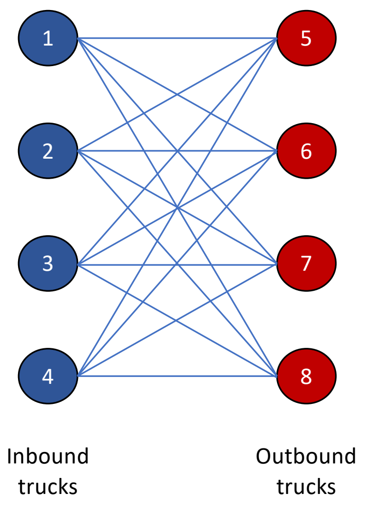


Fig. 1. G_T in the case of 1-biclique.

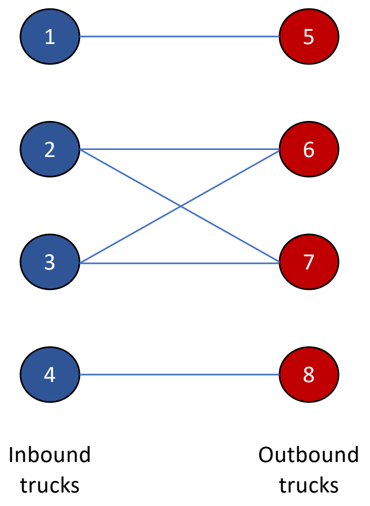


Fig. 2. G_T in the case of 3-biclique.

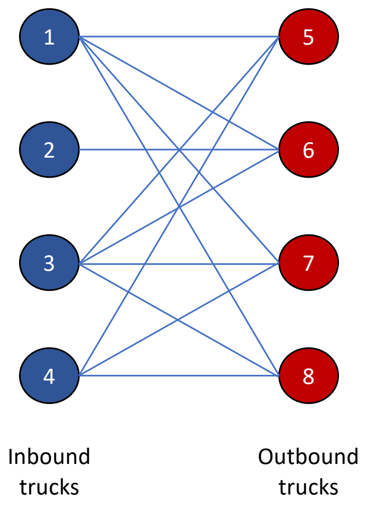


Fig. 3. G_T in the general case.

Corollary 1. *When G_T is a collection of bicliques, $CTSP(1, U)$ is solved by sequencing the trucks involved in each biclique consecutively as dictated by Theorem 1, and then sequencing the bicliques in any order.* □

Let us now turn to problem $CTSP(1, U)$ when G_T has a general structure (see Figure 3), which can be stated in decision form as follows.

"Given a positive integer H, is there a truck sequence at the dock such that the total flow time does not exceed H?"

The following result holds.

Theorem 2. *CTSP(1,U) is NP-complete.*

Proof. Reduction from OPTIMAL LINEAR ARRANGEMENT. □

The complexity of $CTSP(1, C)$ when G_T has a general structure remains open.
Let us now turn to $CTSP(2, C)$, i.e., the case in which there are two gates and processing times are correlated. The following result holds:

Theorem 3. *CTSP(2,C) is NP-complete even when G_T is a 1-biclique.*

Proof. Reduction from PARTITION. □

4 Conclusion

In conclusion, we summarize our findings in the following table (where NPC stands for NP-Complete). Note that the case where G_T has a bi-clique structure is significant to determines the frontier between the polynomial and NP-complete cases. Moreover, as it is always possible (by removing arcs) to transform a general G_T graph in order to give it a bi-clique structure, having efficient methods to solve the biclique case can give good lower bounds for the general case.

n	C , 1-biclique	C	U, 1-biclique	U
1	$O(n)$	open	$O(n \log n)$ (Th. 1)	NPC (Th.2)
2	NPC (Th.3)	NPC (Th.3)	NPC (Th.3)	NPC (Th.3)

References

U.M. Apte and S. Viswanathan, Effective cross docking for improving distribution efficiencies, International Journal of Logistics: Research and Applications, 3 (3), 291–302.

N. Boysen, Truck scheduling at zero-inventory cross docking terminals, Computers & Operations Research, 37, 32–41.

N. Boysen and M. Fliedner and A. Scholl, Scheduling inbound and outbound trucks at cross docking terminals, OR Spectrum, 32, 135–161.

Chiarello, A., Gaudioso, M., Sammarra, M., Truck synchronization at single door crossdocking terminals, OR Spectrum 40(2), 395–447 (2018).

M. E. Dyer and L. A. Wolsey, Formulating the single machine sequencing problem with release dates as a mixed integer problem, Discrete Applied Mathematics, 26, 255–270.

Garey, M.R., R.L. Graham, D.S. Johnson, D.E.Knuth, Complexity results for bandwidth minimization, *SIAM Journal on Applied Mathematics*, 34, 477–495.

J.K. Lenstra and A.H.G. Rinnooy Kan and P. Brucker, Complexity of machine scheduling problems, Annals of Discrete Mathematics, 1, 343–362.

M. Lombardi and M. Milano, A min-flow algorithm for Minimal Critical Set detection in Resource Constrained Project Scheduling, Artificial Intelligence, 182-183, 58–67.

J. van Belle and P. Valckenaers and D. Cattrysse, Cross docking: State of the art, Omega,40 (6), 827–846.

W. Yu and P.J. Egbelu, Scheduling of inbound and outbound trucks in cross docking systems with temporary storage, *International Journal of Production Economics*, 184, 377–396.

Linear inequalities for neighborhood based dominance properties for the common due-date scheduling problem

Anne-Elisabeth Falq[1], Pierre Fouilhoux[1] and Safia Kedad-Sidhoum[2]

[1] Sorbonne Université, CNRS, LIP6, 4 place Jussieu, 75005 Paris, France
anne-elisabeth.falq@lip6.fr, pierre.fouilhoux@lip6.fr
[2] CNAM, CEDRIC, 292 rue Saint Martin, 75003 Paris, France safia.kedad_sidhoum@cnam.fr

Keywords: just-in-time scheduling, dominance properties, integer linear programming.

1 The common due-date problem

We consider a set of n tasks J that have to be processed non-preemptively on a single machine around a common due-date d. Given for each task $j \in J$ a processing time p_j and a unitary earliness (resp. tardiness) penalty α_j (resp. β_j), the problem denoted $1 \,|\,|\, \sum \alpha_j \, [d{-}C_j]^+ {+}\, \beta_j \, [C_j{-}d]^+$, aims at finding a feasible schedule that minimizes the sum of earliness-tardiness penalties.

When $d \geq \sum p_j$, the due date is said **unrestrictive**, and the problem is NP-hard, even if penalties are symmetric, *i.e.* $\alpha_j = \beta_j$ for all $j \in J$ (Hall and Posner 1991). In the general case, the problem is NP-hard, even if the task penalties are equal, *i.e.* $\alpha_j = \beta_j$ for all $j \in J$ (Hoogeveen and van de Velde 1991). In both cases to the dynamic programming algorithms proposed in (Hall and Posner 1991, Hoogeveen and van de Velde 1991). A heuristic method together with a benchmark is provided in (Biskup and Feldmann 2001). These instances are efficiently solved by an exact method proposed in (F. Sourd 2009).

In this work, we focus on the problem with an unrestrictive due-date. We propose a compact integer linear program modeling it. To improve the efficiency of this formulation, we propose a new type of linear inequalities translating some neighborhood based dominance properties. Moreover, for sake of brevity, we assume that the α-ratios α_j/p_j for $j \in J$ are different, as well as the β-ratios β_j/p_j. Nevertheless, the following results are still true without this assumption.

2 A compact linear formulation based on structural dominance properties

In a given schedule, a task is **early** (resp. **tardy**), if it completes before or at d (resp. after d), and a task is **on-time** if it completes exactly at time d. A schedule having an on-time task is said **V-shaped**, if early (resp. tardy) tasks are ordered by increasing α-ratios (resp. decreasing β-ratios). A schedule is called a **block** if it presents no idle time.

For the unrestrictive case, V-shaped blocks having an on-time task are dominant, which means that there exists an optimal solution within this set of schedules (Hall and Posner 1991). Using this dominance property, a schedule can be completely described by the partition between early and tardy tasks.
Indeed, if the set of early tasks E is given, the set of tardy tasks $T = J \setminus E$ is also fixed, and the earliness e_u (resp. the tardiness t_u) of any task $u \in J$, can be deduced as follows:

$$e_u = \begin{cases} p\big(A(u) \cap E\big) & \text{if } u \in E \\ 0 & \text{otherwise} \end{cases} \qquad t_u = \begin{cases} p\big(B(u) \cap T\big) & \text{if } u \in T \\ 0 & \text{otherwise} \end{cases}$$

where $p(S) = \sum\limits_{j \in S} p_j$ for any $S \subseteq J$, $A(u) = \Big\{ j \in J \mid \frac{\alpha_j}{p_j} > \frac{\alpha_u}{p_u} \Big\}$ and $B(u) = \Big\{ j \in J \mid \frac{\beta_j}{p_j} > \frac{\beta_u}{p_u} \Big\}$.

Note that, for each task u, the sets $A(u)$ and $B(u)$ are defined from the instance, so they can be pre-computed. We introduce, for each $u \in J$, $\bar{A}(u) = J \setminus \big(A(u) \cup \{u\}\big)$ and $\bar{B}(u) = J \setminus \big(B(u) \cup \{u\}\big)$.

Let us consider a boolean variable δ_j for each $j \in J$ indicating if task j is early. *i.e.* a vector $\delta \in \{0,1\}^J$ encodes the partition $\big(E = \{j \in J \mid \delta_j = 1\},\ T = \{j \in J \mid \delta_j = 0\}\big)$. Although these variables are sufficient to encode solutions, additional boolean variables are introduced to replace quadratic terms appearing in the earliness and tardiness expression. Since these terms are only products of boolean variables, we use the classical linearization from (R. Fortet 1959) : for each couple in $J^< = \big\{(i,j) \in J^2 \mid i < j\big\}$, we add a new boolean variable $X_{i,j}$ and the four following inequalities coupling it with variables δ_i and δ_j.

$$\forall (i,j) \in J^<, \quad X_{i,j} \geqslant \delta_i - \delta_j \tag{1}$$
$$\forall (i,j) \in J^<, \quad X_{i,j} \geqslant \delta_j - \delta_i \tag{2}$$
$$\forall (i,j) \in J^<, \quad X_{i,j} \leqslant \delta_i + \delta_j \tag{3}$$
$$\forall (i,j) \in J^<, \quad X_{i,j} \leqslant 2 - (\delta_i + \delta_j) \tag{4}$$

If $(\delta, X) \in \{0,1\}^J \times \{0,1\}^{J^<}$ satisfies inequalities $(1)-(4)$, then $X_{i,j}$ indicates if $\delta_i \neq \delta_j$, and more importantly $\delta_i \delta_j = \delta_i + \delta_j - X_{i,j}$ and $(1-\delta_i)(1-\delta_j) = 2 - \delta_i - \delta_j - X_{i,j}$. The objective function reduces then to the following linear function:

$$f(\delta, X) = \sum_{u \in J} \alpha_u \left(\sum_{j \in A(u)} p_j \, \frac{\delta_j + \delta_u - X_{j,u}}{2} \right) + \beta_u \left((1-\delta_u)\, p_u + \sum_{j \in B(u)} p_j \, \frac{2 - \delta_j - \delta_u - X_{j,u}}{2} \right)$$

By introducing the polyhedron $P = \Big\{(\delta, X) \in [0,1]^J \times [0,1]^{J^<} \mid (1)-(4)\Big\}$, and denoting $int(P)$ its integer points, the problem can be formulated as a linear integer program (A-E. Falq, P. Fouilhoux and S. Kedad-Sidhoum 2019):

$$(F) \quad \min_{(\delta, X) \in int(P)} f(\delta, X)$$

Since it has exactly $n + n(n-1)/2$ boolean variables and $4n(n-1)/2$ inequalities, (F) is a compact formulation. Note that no linear inequalities are needed to ensure the task non-overlapping since it is handled through the encoding.

3 Linear inequalities for neighborhood based dominance properties

It is common, in local search procedures, to slightly change a solution $\mathcal{S}$ to obtain a new one $\mathcal{S}'$, called a **neighbor** of $\mathcal{S}$. If the neighbor $\mathcal{S}'$ is better, (*i.e.* if it has a smaller total penalty in our case), we say that $\mathcal{S}$ is dominated (by $\mathcal{S}'$), it follows that $\mathcal{S}$ cannot be optimal.

This simple observation leads to a dominance property for any **neighborhood** $\mathcal{N}$ which associates to a solution the set of its neighbors. A solution $\mathcal{S}$ is said **$\mathcal{N}$-dominated** if there exists $\mathcal{S}' \in \mathcal{N}(\mathcal{S})$ which is strictly better than $\mathcal{S}$. Hence solutions which are not $\mathcal{N}$-dominated are dominant.

Here, as a schedule is encoded by a partition (E, T) between early and tardy tasks, we consider two operations providing a neighbor (E', T'):

- the **insertion** operation, which consists in inserting an early task on the tardy side *i.e.* $E' = E \setminus \{u\}$ and $T' = T \cup \{u\}$ for some $u \in E$, or conversely in inserting a tardy task on the early side *i.e.* $E' = E \cup \{u\}$ and $T' = T \setminus \{u\}$ for some $u \in T$,
- the **swap** operation, which consists in inserting an early task on the tardy side while a tardy task is inserted on the early side *i.e.* $E' = E \setminus \{u\} \cup \{v\}$ and $T' = T \setminus \{v\} \cup \{u\}$ for some $(u,v) \in E \times T$.

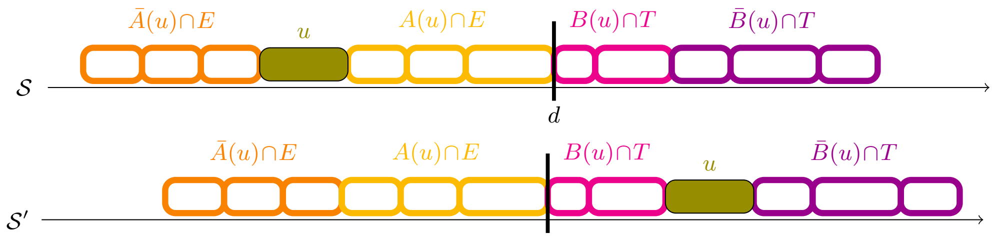


Fig. 1. Insertion of an early task u on the tardy side of a schedule

Figure 1 illustrates the insertion of an early task on the tardy side. Let us fix a task $u \in J$. The top part of the scheme shows the general form of an arbitrary schedule $\mathcal{S}$ in which u is an early task. The bottom part shows the general form of the neighbor $\mathcal{S}'$ of $\mathcal{S}$ obtained by inserting of u on the tardy side. Considering solutions as schedules (rather than partitions) allows to easily express the penalty variation between $\mathcal{S}$ and $\mathcal{S}'$ as follows:

$$-\alpha_u\, p\big(\underline{A(u)\cap E}\big) + \beta_u \Big(p\big(\underline{B(u)\cap T}\big) + p_u\Big) - p_u\, \alpha\big(\underline{\bar{A}(u)\cap E}\big) + p_u\, \beta\big(\underline{\bar{B}(u)\cap T}\big)$$

Using this the penalty variation expression, we produce a linear inequality

• which cuts **all** schedules in which u is early and which are dominated by the schedule obtained by inserting u on the tardy side,

• which is valid for any other schedule, in particular for all optimal schedules since they are non-dominated.

Two elements allow us to produce such an inequality. First, assuming that u is early in the schedule encoded by a vector (δ, X) is equivalent to assume that the linear term $1-\delta_u$ equal zero. Secondly, if (δ, X) encodes a schedule where u is early, the penalty variation induced by the insertion of u on the tardy side, denoted $\Delta_u^{av}(\delta)$ is linear in δ:

$$\Delta_u^{av}(\delta) = -\alpha_u \sum_{i\in A(u)} p_i\, \delta_i \;+\; \beta_u \sum_{i\in B(u)} p_i (1-\delta_i) + \beta_u p_u \;+\; p_u \Big(\sum_{i\in \bar{B}(u)} \beta_i\, (1-\delta_i) - \sum_{i\in \bar{A}(u)} \alpha_i\, \delta_i \Big)$$

and bounded by a constant:

$$\forall \delta \in \{0,1\}^J,\ -\Delta_u^{av}(\delta) \leqslant M_u^{av} \ \text{ where } \ M_u^{av} = \alpha_u\, p\big(A(u)\big) - \beta_u p_u + p_u\, \alpha\big(\bar{A}(u)\big)$$

We finally deduce the following inequality, which translates the dominance of the set of schedules non-dominated by the insertion of u:

$$\Delta_u^{av}(\delta) \geqslant -M_u^{av}\, (1-\delta_u) \quad (5_u)$$

Following the same approach, we produce a similar inequality (6_u) cutting exactly the schedules in which u is tardy, and dominated by inserting u on the early side. We also produce an inequality $(7_{u,v})$, for given $v \neq u$, cutting exactly the schedules in which u is early, v is tardy, and dominated by swapping u and v.

Note that inequalities of family (5), (6) and (7) are not standard reinforcement inequalities. Classically, valid inequalities are added to cut extreme points which are not integer and then do not encode a feasible solution, since they correspond to a too optimistic value. On the contrary, these dominance inequalities cut some integer points which encode feasible solutions because they correspond to dominated, and then non-optimal, schedules.

4 Exact resolution and rounding heuristic

Let us introduce the polyhedron reinforced by the previous dominance inequalities $P' = \Big\{(\delta, X) \in [0,1]^J \times [0,1]^{J^<} \mid (1)-(4), \forall u \in J, (5_u), (6_u),\ \forall (u,v) \in J^<, (7_{u,v}), (7_{v,u})\Big\}$, and the associated formulation : $(F') \quad \min\limits_{(\delta,X) \in int(P')} f(\delta, X)$.

Theoretically, we know that both formulations (F) and (F') give the same value.To compare them from a practical point of view, we implement them using a linear solver (CPLEX version 12.6.3.0), and test them on the benchmark proposed by (Biskup and Feldmann 2001). Under a time limit of one hour, formulation (F) using all CPLEX features allows to exactly solve instances up with [50] tasks, while formulation (F') without any CPLEX features allows to exactly solve instances up with [150] tasks.

Although designed for exact solving, (F) (resp. (F')) can be used to obtain a lower bound, by solving its linear relaxation denoted $\bar{F}$ (resp. $\bar{F}'$), and to obtain an upper bound together with a feasible schedule, by rounding the fractional solution of $\bar{F}$ (resp. $\bar{F}'$).

A first rounding procedure consists in rounding vector δ and then fixing X accordingly so that we obtain $\hat{x} \in int(P)$. We then obtain a feasible schedule and an upper bound UB1 for (F) (resp. UB1' for (F')). Note that $\hat{x}$ can violate some dominance inequalities, then $\hat{x} \notin P'$ (that implies in particular that CPLEX does not accept $\hat{x}$ as an incumbent solution). So we add a **repairing phase**, which consists in applying swap and insert operations as long as it is possible, *i.e.* while an insert inequality or a swap inequality is violated, meaning that an insert or a swap operation strictly improves the solution. We finally obtain a non-dominated schedule and a better upper bound UB2 (resp. UB2').

Note that this repairing phase can also be applied to the heuristic solution provided by the Biskup and Feldmann algorithm (Biskup and Feldmann 2001), which possibly transforms their upper bound UB3 in a better upper bound UB4. We compare experimentally these four upper bounds and show that UB2, UB2' and UB4 are very strong : they are exact for 45 over 50 instances (of size up to 150), and the average gap to the optimal value for the 5 other instances is less than 0,1%.

References

D. Biskup and M. Feldmann, 2001, "Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates", *Computers and operations research*, 28:787–801

A-E. Falq, P. Fouilhoux and S. Kedad-Sidhoum, 2019, Mixed integer formulations using natural variables for single machine scheduling around a common due date. *CoRR*, abs/1901.06880, 2019.

R. Fortet, 1959, "L'Algèbre de Boole et ses applications en Recherche Opérationnelle", *Cahiers du Centre d'Études en Recherche Opérationnelle*, 4:5

N.G. Hall and M.E. Posner, 1991, "Earliness-tardiness scheduling problems, I: weighted deviation of completion times about a common due date", *Operations Research*, 39(5):836–846.

J.A. Hoogeveen and S.L. van de Velde, 1991, "Scheduling around a small common due date", *European Journal of Operational Research*, Vol 55:237–242,

F. Sourd, 2009, "New exact algorithms for one-machine earliness-tardiness scheduling", *INFORMS Journal on Computing*, 21(1):167–175

An acceleration procedure for several objective functions in the permutation flow shop scheduling problem

Victor Fernandez-Viagas[1], Jose M. Molina-Pariente[1], Carla Talens[1] and Jose M. Framinan[1]

Industrial Management, School of Engineering, University of Seville, Spain
vfernandezviagas,jmolina1,cartafa,framinan@us.es

Keywords: Scheduling, Flowshop, flow shop, accelerations, speed-up, total completion time, total tardiness, total earliness, just-in-time,critical path.

1 Introduction

In a flowshop layout, n jobs have to be processed on m machines following all jobs the same route of machines. The flowshop scheduling problem involves the search of the best order to process the jobs in each machine. Traditionally, a common simplification, denoted as Permutation Flowshop Scheduling Problem (PFSP), is adopted to avoid an extensive use of manpower or machines in the shop, where the order of jobs does not change between machines. This particular problem is one of the most studied optimization problems in Operations Research (Fernandez-Viagas *et. al.* 2017), probably due to the following reasons: this flowshop layout is very common in real manufacturing scenarios (Vakharia and Wemmerlov 1990); many job shops can be simplified to a reduced flow shop under several constraints (Storer *et. al.* 1992); and many models and solution procedures for different constraints and layouts have their origins in the flowshop scheduling problem. The PFSP is denoted by $Fm|prmu|\gamma$, where γ is the goal to be solved. Due to the NP-hard nature of the problem, many approximate algorithms have been proposed in the literature for the traditional problem and/or related constrained PFSP. Without any doubt, one of the key factor for the efficiency of these approaches is the use of methods to accelerate the calculation of the objective functions or local search methods. In this regards, Taillard (1990) proposed the first speed-up procedure for $Fm|prmu|C_{max}$ (denoted as Taillard's accelerations) in the literature. These accelerations have been incorporated in hundreds of papers and its use is nowadays mandatory to obtain an efficient approximate algorithm for the $Fm|prmu|C_{max}$ problem. Unfortunately, they can be only applied in a very few amount of objectives and/or constraints, and the search for more efficient accelerations is still open in the literature. Thereby, in the race for finding accelerations for approximate algorithms in other related problems, different speed-up procedures have been proposed in the literature taking into account their specific problem properties. In this paper, we propose a new speed-up procedure for $Fm|prmu|\sum C_j$, $Fm|prmu|\sum T_j$, and $Fm|prmu|\sum E_j + \sum T_j$, using specific properties based on the critical path, which clearly outperforms previous proposals.

2 Literature review

A number of speed-up procedures have been proposed to accelerate approximate algorithms in the related flowshop problems. To the best of our knowlegde, the well-known accelerations proposed by Taillard (1990) was the first proposal published in the literature for flowshop layouts. Using the specific properties of the $Fm|prmu|C_{max}$ problem, the author develops a very efficient procedure to accelerate the evaluation of approximated algorithms in this problem. More specifically, using this procedure the complexity of insertion local search methods can be reduced from $n^3 \cdot m$ to $n^2 \cdot m$. Due to the repercussion and excellent results of this paper, several other authors have been adapted them to related PFSP. Thereby, Mercado and Bard (1998) adapt the Taillard's accelerations for the PFSP with sequence-dependent and anticipatory setup times and makespan minimisation, i.e. $Fm|s_{ijk}, prmu|C_{max}$. Naderi and Ruiz (2010) adapted them for the distributed permutation flowshop scheduling problem to minimise makespan, $DF|prmu|C_{max}$. For the blocking PFSP and makespan minimisation ($Fm|blocking|C_{max}$), they have been first proposed by Wang *et. al.* (2010). For the no-idle case (idle times no allowed on machines), they are adapted by Fatih Tasgetiren *et. al.* (2013) to minimise the makespan and by Pan and Ruiz (2010) for the mixed no-idle permutation flowshop scheduling problem with makespan minimisation. In the non-permutation case also for makespan, they are extended by Benavides and Ritt (2016). In the PFSP with time lags constraints, Wang *et. al.* (2018) adapt the Taillard's accelerations to minimise the makespan. Regarding other objective functions, these accelerations have been only adapted in some very particular cases. For the no-idle case, they have successfully adapted by Fatih Tasgetiren *et. al.* (2013) for total completion time ($Fm|prmu, no-idle|\sum C_j$) and total tardiness ($Fm|prmu, no-idle|\sum T_j$), respectively. Fernandez-Viagas and Framinan (2015) adapted the accelerations to a different objective function, the $Fm|prmu|\epsilon(C_{max}/T_{max})$ problem, but allowing some infeasible solutions that should be repaired. They adapt the accelerations for $Fm|prmu|\epsilon(C_{max}/T_{max})$. In addition, they incorporated a bounded local search based on problem properties, also to reduce the number of positions where the jobs are inserted.

Other accelerations have been also proposed in the PFSP literature, when the completion time of each job is needed to evaluate the objective function. Thereby, Framinan and Leisten (2008) propose accelerations (denoted as FL's accelerations) to reduce the evaluation of insertion local search in the PFSP with minimisation of total tardiness ($Fm|prmu|\sum T_j$). Note that these accelerations can be easily extended to interchange local search, or other related PFSP as e.g.: $Fm|prmu|\sum C_j$, $Fm|prmu|\sum E_j + \sum T_j$, and $DF|prmu|\sum C_j$. Finally, some methodologies -reducing the number of moves in local searches- have also been proposed in the literature, denoted as bounded local searches. Basically, they reduce these moves by evaluating some lower or upper bounds and comparing them against the so-far best solution found by the algorithm. This is the case of the aforementioned bounded local search proposed by Fernandez-Viagas and Framinan (2015) in $Fm|prmu|\epsilon(C_{max}/T_{max})$ which bound the feasible positions where insert the jobs. Fernandez-Viagas and Framinan (2014) also use such as a procedure avoiding the insertion or interchange of jobs in factories for the distributed PFSP ($DF|prmu|C_{max}$). Finally, a lower bound is used in Pagnozzi and Stützle (2017) to discard the evaluation of some insertions in the PFSP with weighted total tardiness ($Fm|prmu|\sum w_j T_j$).

3 New speed up procedure

In this section, we propose a new speed up procedure, denoted as FMF accelerations, for insertion-based local search methods in the PFSP. The procedure highly reduces the CPU time of this type of local search methods by avoiding the calculation of a number of operations. Basically, it is based in the fact that, when inserting job σ in position j, only the completion times of the jobs between the critical path and job π_{j+1} need to be calculated to obtain any objective function in the $Fm|prmu|-$ problem because: jobs over the critical path do not influence in the completion times on the critical path; and the completion times of jobs before $j+1$ are known from previous iteration. In Figure 1, we present an example of a sequence composed of four jobs (1,2,4,5) where a new job 3 wants to be inserted in position 3. The critical path is shown in solid gray and the only operations whose completion times must be evaluated are with diagonal green lines. As a consequence only the completion times of the jobs between the critical path and job π_{j-1} need to be calculated to obtain any objective function in the $Fm|prmu|-$ problem.

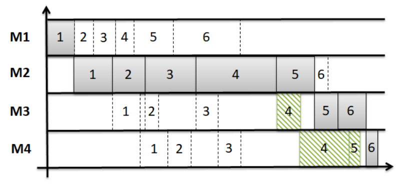


Fig. 1. Example of the new speed-up

With this in mind, the detailed procedure of the proposed accelerations can be explained as follows. Firstly, we calculate e_{ij}, q_{ij}, f_{ij} for sequence Π (i.e. without job σ) according to the following equations:

$$e_{ij} = max\{e_{i,j-1}, e_{i-1,j}\} + p_{i\pi_j}, i = 1 \ldots m, j = 1 \ldots k-1 \tag{1}$$

$$q_{ij} = max\{q_{i+1,j}, q_{i,j+1}\} + p_{i\pi_j}, i = m \ldots 1, j = k-1 \ldots 1 \tag{2}$$

$$f_{ij} = \max\{e_{i,j-1}, f_{i-1,j}\} + p_{i\sigma}, i = 1 \ldots m, j = 1 \ldots k \tag{3}$$

In addition we introduce cp_{ij} as a variable to reproduce the critical path after the insertion of the new job. cp_{ij} is then equals to 1 if there is no idle time between the operation O_{i,π_j} and $O_{i,\pi_{j+1}}$, and 0 otherwise (i.e. when there is no idle time between O_{i,π_j} and O_{i+1,π_j}). Next, job σ is tested in each position j of sequence Π. In each of these positions, we determine the machine $i^{'}$ where the forward and backward critical paths join, i.e. $i^{'} = max_{i=1 \ldots m}\{f_{ij} + q_{ij}\}$. Then, we calculate the value of the objective function for all previous insertion. There, the variable cp_{ij} is used to reproduced the critical path. If $cp_{ij} = 1$, the completion time of job in position $j+2$ (which corresponds to job pi_{j+1}) is the actual load of machine i (denoted $Load_i$) plus the processing time $p_{i\pi_{j+1}}$ and the completion time of this job in the other machines is updated. Otherwise, the critical path moves to a higher machine and the completion time of job in position $j+1$ can directly be obtained adding the processing time to the load of previous machine.

4 Computational Results and Conclusions

In this section, we compare the proposed FMF accelerations against previous accelerations proposed in the literature. Three different experimentations have been carried out by implementing the proposed accelerations in the following three different objective functions: Experimentation #1: Total completion time ($Fm|prmu|\sum C_j$); Experimentation #2: Total tardiness ($Fm|prmu|\sum T_j$); and Experimentation #3: Total earliness and tardiness ($Fm|prmu|\sum E_j + \sum T_j$).

The computational results have been developed on two well-known sets of extensive instances with and without due dates, respectively. Results show the excellent performance of the proposed accelerations, regardless the objective function and the benchmark. More specifically, the proposed accelerations clearly outperform each other accelerations proposed in the literature so far. Thereby, in the minimisation of total earliness and tardiness, the

CPU times are reduced in average 75.5% against the method without accelerations and 42.9% against the best so far accelerations found in the literature. For some instances the reduction is around a 90% of the computational time (as compared not using accelerations). In the total tardiness case, the average reduction of computational time is 63.5% against the 44.6% and 45.9% of the PS's and FL's accelerations respectively. Similarly, the reduction in the total completion time is 50.8% against the 37.7% found by the FL's accelerations.

Acknowledgements

This research has been funded by the Spanish Ministry of Science and Innovation, under the project "PROMISE" with reference DPI2016-80750-P.

References

Taillard, E., 1990, "Some efficient heuristic methods for the flow shop sequencing problem", *European Journal of Operational Research*, Vol. 47(1), pp. 65-74.

Vakharia A.J., Wemmerlov U., 1990, "Designing a cellular manufacturing system: a materials flow approach based on operation sequences", *IIE Transactions*, Vol. 22.

Rios-Mercado R.Z. and Bard J.F., 1998, "Heuristics for the flow line problem with setup costs", *European Journal of Operational Research*, Vol. 110, pp. 76-98.

Benavides A.J. and Ritt M., 2016, "Two simple and effective heuristics for minimizing the makespan in non-permutation flow shops", *Computers and Operations Research*, Vol. 66, pp. 160-169.

Naderi B. and Ruiz R., 2010, "The distributed permutation flowshop scheduling problem", *Computers & Operations Research*, Vol. 37, pp. 754-768.

Framinan J.M. and Leisten R., 2008, "Total tardiness minimization in permutation flow shops: A simple approach based on a variable greedy algorithm", *International Journal of Production Research*, Vol. 46, pp. 6479-6498.

Pan Q.K. and Ruiz R., 2010, "An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem", *Omega*, Vol. 44, pp. 41-50.

Fernandez-Viagas V. and Framinan J.M., 2015, "Efficient non-population-based algorithms for the permutation flowshop scheduling problem with makespan minimisation subject to a maximum tardiness", *Computers & Operations Research*, Vol. 64, pp. 86-96.

Fernandez-Viagas V. and Framinan J.M., 2014, "A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem", *International Journal of Production Research*, Vol. 53, pp. 1111-1123.

Pagnozzi F. and Stützle T., 2017, 'Speeding up local search for the insert neighborhood in the weighted tardiness permutation flowshop problem", *Optimization Letters*, Vol. 11(7), pp. 1283-1292.

Storer R.H., Wu S.D. and Vaccari R., 1992, "New search spaces for sequencing problems with application to job shop scheduling", *Management Science*, Vol. 38, pp. 1495-1509.

Fernandez-Viagas V., Ruiz R. and Framinan J.M., 2017, "A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation", *European Journal of Operational Research*, Vol. 257(3), pp.707-721.

Wang L. and Pan Q.-K. and Suganthan P.N. and Wang W.H. and Wang Y.M., 2010, "A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems", *Computers and Operations Research*, Vol. 37(3), pp.509-520.

Fatih Tasgetiren M. and Pan Q.K. and Suganthan P.N. and Buyukdagli O., 2013, "A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem", *Computers and Operations Research*, Vol. 40(7), pp.1729-1743.

Wang B. and Huang K. and Li T., 2018, "Permutation flowshop scheduling with time lag constraints and makespan criterion", *Computers Industrial Engineering*, Vol. 120, pp. 1-14.

Scheduling problems with processing time dependent profit: applications and a nice polynomial case

Florian Fontan, Nadia Brauner, Pierre Lemaire

Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, F-38000 Grenoble, France
`{firstname.lastname}@grenoble-inp.fr`

Keywords: scheduling, controllable processing time, polynomial algorithm, b-matching

We study the complexity of scheduling problems where jobs have a variable processing time: one can *decide* the processing time of each job. The profit for a job then depends on its allocated processing time. Detailed results and proofs for this problem can be found in the thesis of Fontan (2019). This abstract presents an insight of that document.

In our experience, the problem originates from astrophysics and the search for exoplanets (Lagrange *et. al.* 2016). Astrophysicists want to schedule observations on a telescope and, for each possible target (star), there exist time-windows when it is visible, a required duration for its observation, and an interest for observing it; the objective is to maximize the total interest of the schedule. This primary version of the problem has been described and solved by Catusse *et. al.* (2016); but it appears that shortening an observation would be worth doing if that makes room for another one. More generally an observation remains relevant even if its processing (observation) time is slightly less than the required value, with an accordingly downgraded interest. Such a situation can be modeled by processing time dependent profits. The general problem is NP-complete, and an efficient practical solution algorithm has been proposed Fontan (2019). In the following, we present the model for processing time dependent profit (Section 1), limited to the case where time-windows only differ by their deadlines. Then in Section 2, we focus on a special polynomial case to show a proof technique for those problems.

1 Processing time dependent profit

We consider a scheduling problem with n jobs and m identical parallel machines; each job T_j has a deadline d_j and a profit function $w_j(p_j)$ that depends on the decided processing time p_j. Figure 1 shows three examples of profit functions.

A schedule is feasible if it satisfies the following conditions:

- every machine processes only one job at a time,
- a scheduled job T_j must start after 0 and end before its deadline d_j,
- preemption is not allowed.

The objective is to find a feasible schedule that maximizes the total profit:

$$\max \sum_{j=1}^{n} w_j(p_j)$$

with the convention that $p_j = 0$ if T_j is not scheduled.

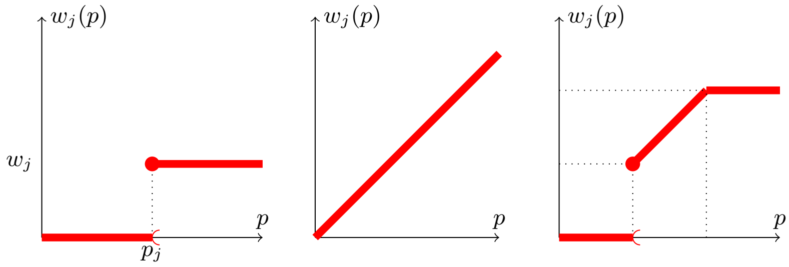
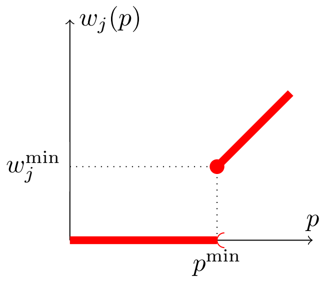


Fig. 1. Examples of profit functions of: (a) a classical scheduling problem ($w_j(p) = w_j$ if $p \geq p_j$, 0 otherwise); (b) a basic (linear profit) problem; (c) the star observation problem

2 Focus on a polynomial case solved with maximum weight b-matching

In this section, we focus on a specific variant with a common deadline, *i.e.* $d_j{=}d$ for all jobs T_j, and the following profit function:

$$w_j(p) = \begin{cases} 0 & \text{if } p < p^{\min} \\ w_j^{\min} + b_j(p - p^{\min}) & \text{if } p \geq p^{\min} \end{cases}$$



We exhibit a polynomial algorithm for this special processing time dependent profit maximization scheduling problem with parallel machines. This algorithm uses as subproblem the maximum weight b-matching which can be solved in polynomial time (Schrijver 2002): Given a graph $G(V, E)$, a demand/supply b_v for each vertex $v \in V$, and a weight/cost c_e for each edge $e \in E$, a b-matching of G is a vector $x \in \mathbb{N}^E$ such that $\sum_{u,(uv)\in E} x_{(uv)} \leq b_v$ for all $v \in V$. The weight of a b-matching $x \in \mathbb{N}^E$ is defined as $\sum_{e\in E} c_e x_e$. The maximum weight b-matching problem is then the problem of finding a b-matching of maximum weight in G.

A set of solutions is dominant if it contains at least one optimal solution. Our algorithm consists in solving a polynomial number of maximum weight b-matching problems which rely on the dominant set described below.

We consider the case $d = qp^{\min} + r$, with $q, r \in \mathbb{N}$, $q \geq 2$, $0 < r < p^{\min}$. However, with a similar reasoning, the results can be adapted for the case $d = qp^{\min}$, $q \in \mathbb{N}$, $q \geq 2$. The case $d < 2p^{\min}$ is trivial.

Lemma 1. *Let $\mathcal{U}$ be the set of solutions such that for all $S \in \mathcal{U}$:*

- *for all $T_j \in S$, $p_j \geq p^{\min}$;*
- *on each machine, there exists at most one job $T_j \in S$ such that $p_j \neq p^{\min}$;*
- *there exists at most one job $T_j \in S$ such that $p_j \notin \{d, p^{\min}, p^{\min} + r\}$. Such a job is called a* special *job.*

Then, $\mathcal{U}$ is dominant.

Figure 2 illustrates the structure of the solutions of $\mathcal{U}$. The horizontal axis corresponds to the time, each line represents a machine and each striped rectangle and its length

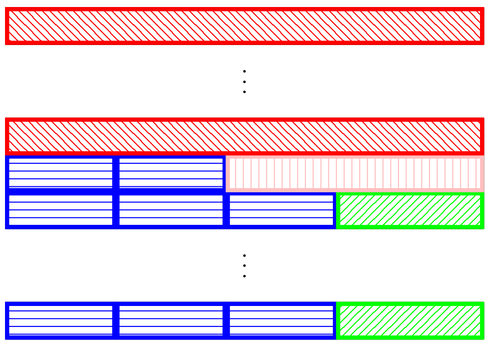


Fig. 2. Illustration of the structure of the solutions in $\mathcal{U}$

represent a scheduled job and its processing time. The long jobs on the top machines are those of length d; the smallest jobs are those of length $p^{\min}$; the jobs at the end of the bottom machines are those of length $p^{\min} + r$; the job with vertical stripes is a special job.

Let $S \notin \mathcal{U}$ be an optimal solution. The idea of the proof of Lemma 1, that we do not detail here, is to build another solution $S' \in \mathcal{U}$ from S without degrading its value.

Theorem 1. *The processing time dependent profit scheduling problem with a common deadline for all jobs and a profit function as decribed at the beginning of this section can be solved in polynomial time.*

Proof. Following Lemma 1, we only focus on solutions of $\mathcal{U}$. Thus, we can partition the jobs of a solution $S \in \mathcal{U}$ into 4 subsets depending on their processing time (and one more for the non scheduled jobs):

$$\begin{aligned}
U_1(S) &= \{T_j \in S, \quad p_j = d\} \\
U_2(S) &= \{T_j \in S, \quad T_j \text{ is a special job}\} \\
U_3(S) &= \{T_j \in S, \quad p_j = p^{\min} + r\} \\
U_4(S) &= \{T_j \in S, \quad p_j = p^{\min}\} \\
U_0(S) &= \{T_j \notin S\}
\end{aligned}$$

In addition, we define $n_k(S)$ the cardinality of $U_k(S)$ and $t(S)$ the number of jobs scheduled on the same machine as the job of U_2:

$$\forall k \in \{0, \dots, 4\}, \quad n_k(S) = |U_k(S)|$$

$$t(S) = \begin{cases} 0, & \text{if } U_2(S) = \emptyset \\ \frac{d - p_j}{p^{\min}}, & \text{if } U_2(S) = \{T_j\} \end{cases}$$

Note that, if $U_2(S) = \{T_j\}$, then $p_j = d - t(S)p^{\min}$.

We now infer the following relations. For all $S \in \mathcal{U}$:

$$0 \leq n_1(S) \leq m \qquad 0 \leq t(S) \leq n - 1$$

$$\begin{aligned}
n_2(S) &= \begin{cases} 0, \text{ if } t(S) = 0 \\ 1, \text{ otherwise} \end{cases} \\
n_3(S) &= m - n_1(S) - n_2(S) \\
n_4(S) &= (q-1)n_3(S) + t(S) \qquad \text{(remember that } q = \lfloor d/p^{\min} \rfloor\text{)} \\
n_0(S) &= n - n_1(S) - n_2(S) - n_3(S) - n_4(S)
\end{aligned}$$

Therefore, if $n_1(S)$ and $t(S)$ are fixed, we can determine $n_2(S)$, $n_3(S)$, $n_4(S)$ and $n_0(S)$. Thus, the optimal value is the best optimal value of the $O(n^2)$ problems with those parameters fixed:

$$\mathrm{OPT} = \max_{(n_1,t)\in(\{0,\dots,m\}\times\{0,\dots,n-1\})} \mathrm{OPT}(n_1,t)$$

Now, we show that for all $(n_1,t) \in \{1,\dots,m\} \times \{0,\dots,n-1\}$, a maximum weight b-matching model can compute $\mathrm{OPT}(n_1,t)$. We create n nodes T_j, $j = 1,\dots,n$ with supply 1, and 5 nodes U_i, $i = 0,\dots,4$ with demand n_i such that $n_2 = 0$ if $t \leq 1$, 1 otherwise; $n_3 = m - n_1 - n_2$, $n_4 = (q-1)n_3 + t$, $n_0 = n - n_1 - n_2 - n_3 - n_4$. Then for all $j = 1,\dots,n$, for all $i = 0,\dots,4$, we add the arc (T_j, U_i) and cost:

$$c_{ji} = \begin{cases} w_j(d) & i = 1 \\ w_j(d - tp^{\min}) & i = 2 \\ w_j(p^{\min} + r) & i = 3 \\ w_j^{\min} & i = 4 \\ 0 & i = 0 \end{cases}$$

A part of the graph including only one job is represented in Figure 3. Edges are only between a node corresponding to a job and a node corresponding to a set U_k. If edge (T_j, U_k) is used in the b-matching solution, then job T_j will be scheduled according to the corresponding set in the corresponding solution of the scheduling problem. For example, if $U_k = U_1$, then $p_j(S) = d$. Hence, the obtained solution thus corresponds to a schedule S with $n_1(S) = n_1$, etc.

The size of the graph is polynomial compared to the size of the instance. Furthermore, the number of problems that we have to solve is $O(n^2)$. Therefore, the problem can be solved in polynomial time.

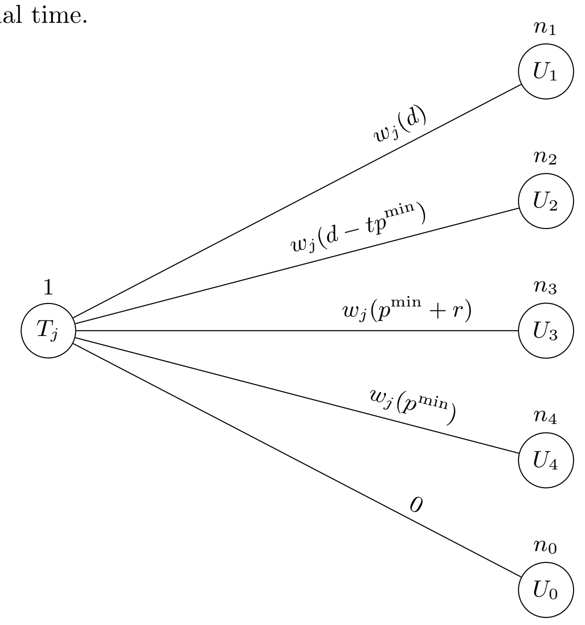


Fig. 3. A part of the graph showing only one job given as input of the maximum weight b-matching problem

References

Anne-Marie Lagrange, Pascal Rubini, Nadia Brauner, Hadrien Cambazard, Nicolas Catusse, Pierre Lemaire, and Laurence Baude. SPOT: an optimization software for dynamic observation programming. In *SPIE 9910, Observatory Operations: Strategies, Processes, and Systems VI, 991033 (July 18, 2016)*, Edinburgh, United Kingdom, July 2016.

Nicolas Catusse, Hadrien Cambazard, Nadia Brauner, Pierre Lemaire, Bernard Penz, Anne-Marie Lagrange, and Pascal Rubini. A Branch-And-Price Algorithm for Scheduling Observations on a Telescope. In *Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 3060–3066. AAAI Press, 2016.

Florian Fontan. *Theoretical and practical contributions to star observation scheduling problems.* PhD Thesis, Grenoble 2019.

Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2002.

Planning problem in Healthcare domain*

Olivier Gérard[1 2], Laure Brisoux Devendeville[1] and Corinne Lucet[1]

[1] MIS Laboratory (EA 4290), University Picardie Jules Verne, France
{olivier.gerard, laure.devendeville, corinne.lucet}@u-picardie.fr
[2] Evolucare Technologies, France
o.gerard@evolucare.com

Keywords: Planning, healthcare, optimization, 0-1 linear programming

1 Introduction

The complexity of planning in healthcare domain is an issue that is increasingly being highlighted by hospitals. Many healthcare problems belong to the family of Resource Constrained Project Scheduling Problems (RCPSP) that are NP-Hard (Garey M.R. and Johnson D.S. 1979) (Baptiste P. *et al.* 2006). The RCPSP problem consists in finding the best assignment of resources and start times to a set of activities. Scheduling problems have been the subject of many studies for decades in various fields (Anthony R.N. 1965) (Blazewicz J. *et al.* 2019), and they are of increasing interest in healthcare domain (Shnits B. *et al.* 2019). Through better patient care and better management of staff time schedules, health facilities want to reduce their costs while improving patient care. There is a rich literature on the variety and the description of these problems (Hall R.W. *et al.* 2012). Nowadays, schedules are mostly designed by hand, a difficult and time-consuming task that can be challenged by kinds of unexpected events. The structure of the problems that might be encountered differs according to the institutions, their size and the number of resources taken into account. The institutions' needs are various, and the criteria for evaluating a schedule may also change from one institution to another or from one department to another within the same institution. In this paper we present a 0-1 linear programming model able to cope with various real-world healthcare scenarios.

The rest of this article is structured as follows. In section 2, we describe and formalize our scheduling problem. In section 3 we present some instances and the corresponding results obtained by the CPLEX solver. In section 4, we conclude with some remarks and perspectives.

2 0-1 Linear programming model

The horizon H is decomposed into timeslots. We have a finite set of resources R. Each resource $r \in R$ is characterised by a set of properties Π_r that determines which roles a resource will be able to hold in an appointment. To each resource $r \in R$ is also associated a set of timeslot t such that $Available_t^r = 1$ if resource r is available at timeslot t. For example, an orthopedic surgeon who is available the first hour over $H =< t_1, t_2, t_3, t_4 >$ will be represented as ressource r with properties *orthopedic surgeon* and *orthopaedist* and with the set of available timeslots $< 1, 1, 0, 0 >$. He will be able to perform surgical operations and medical consultations. A is a set of appointments, such that each appointment $a \in A$ is characterized by its duration $duration_a$, a feasibility interval ES_a and LS_a, $qtreq_a^\pi$ the amount of resources with property π required by a. $Essential_a$ and $Emergency_a$ are two coefficients used to respectivly quantify the importance and the urgency of appointment

* This project is supported by LORH project (CIFRE N° 2018/0425 between Evolucare and MIS Laboratory)

a. They both occur as penalties in the objective function. Each appointment a is also defined by a set of resources R_a having one of the properties required by a and Π_a the set of properties required by a. $PreAssigned_a$ is a set of couples $(resource, property)$ pre-assigned to a. A_r is a set of appointments in which a resource r can participate.

We define decision variables x and y, with $x_a^{r,\pi} = 1$ if resource r with the property π is assigned to appointment a and $y_a^t = 1$ if appointment a starts at the timeslot t. Now, we are going to present the hard constraints mentioned above.

A resource may have multiple properties, and thus be able to perform multiple roles. However, resource r can only be allocated to appointment a with exactly one of its properties π if a is scheduled:

$$\forall\, a \in A, \forall\, r \in R_a, \sum_{\pi \in \Pi_r} x_a^{r,\pi} \leq 1 \tag{1}$$

If resource r does not have property π, it cannot be allocated to appointment a with this property π:

$$\forall\, r \in R, \sum_{\pi \in \Pi \setminus \Pi_r} \sum_{a} x_a^{r,\pi} = 0 \tag{2}$$

If resource r does not have any of the properties π required by appointment a, it cannot be allocated to a:

$$\forall\, a \in A, \sum_{r \in R \setminus R_a} \sum_{\pi \in \Pi} x_a^{r,\pi} = 0 \tag{3}$$

Each appointment a must be planned into a feasibilty interval determined by ES_a and LS_a:

$$\forall a \in A,\ ES_a \times \sum_{t \in H} y_a^t \leq t \times \sum_{t \in H} y_a^t \leq LS_a \tag{4}$$

If appointment a is planned, it is necessary to allocate the required quantity of resources with property π:

$$\forall\, a \in A, \forall\, \pi \in \Pi_a, \sum_{r \in R_a} x_a^{r,\pi} = qtreq_a^{\pi} \times \sum_{t \in [ES_a; LS_a]} y_a^t \tag{5}$$

Each resource r, allocated to appointment a, must be available for the complete duration of a:

$$\forall\, a \in A, \forall\, r \in R_a, \forall\, t \in [ES_a; LS_a],$$
$$duration_a \times \sum_{\pi \in \Pi_a} x_a^{r,\pi} - y_a^t \times \sum_{t'=t}^{t+duration_a-1} dispo_t^r \leq (1 - y_a^t) \times H \tag{6}$$

An appointment a is at most scheduled once:

$$\forall\, a, \in A \sum_{t \in [ES_a; LS_a]} y_a^t \leq 1 \tag{7}$$

Resources are not allocated if the appointment a is not scheduled:

$$\forall\, a \in A, |R| \times |\Pi| \times \sum_{t\in[ES_a;LS_a]} y_a^t \geq \sum_{r\in R_a} \sum_{\pi\in\Pi_a} x_a^{r,\pi} \tag{8}$$

Resource r cannot be allocated to different appointments on same timeslot t:

$$\forall\, r \in R, \forall\, a \in A_r, \forall\, b \in A_r - \{a\}, \forall\, t \in [\max(ES_a; ES_b); \min(LS_a; LS_b)],$$

$$\sum_{\pi\in\Pi_r} x_a^{r,\pi} + \sum_{t'=t-duration_b+1}^{t} y_a^{t'} + \sum_{\pi\in\Pi_r} x_b^{r,\pi} + \sum_{t'=t-duration_b+1}^{t} y_b^{t'} \leq 3 \tag{9}$$

In most cases, an appointment is associated to a specific resource. The following constraint ensures that the resources r with their property π in $PreAssigned_a$ are allocated to appointment a:

$$\forall a \in A, \forall(r,\pi) \in PreAssigned_a,\ x_a^{r,\pi} = \sum_{t\in[ES_a;LS_a]} y_a^t \tag{10}$$

The quality of a solution is evaluated by an objective function f that computes the sum of the unplanned appointments $a \in A$, weighted by the importance factor $Essential_a$ and the sum of the differences between the start date of an appointment a and ES_a, weighted by the emergency factor $Emergency_a$. The purpose is to find a valid solution while minimizing the objective function defined in equation 11.

$$f = \sum_{a\in A}(1 - \sum_{t\in[ES_a;LS_a]} y_a^t) \times Essential_a + \sum_{a\in A} \sum_{t\in[ES_a;LS_a]} y_a^t \times \frac{t - ES_a}{LS_a - ES_a} \times Emergency_a \tag{11}$$

3 Experimentations and results

We generated instances from four different scenarios with the help of various planners from different health care facilities in France who face daily concrete problems.

Table 1. Description of the scenarios.

Scenario	**Number of resources**		**Resources av.**	**Appointments dur.**	**Number of appointments**		**Horizon**
					Per patient	Total	
SurgDep	π_1 = patient	16	100%	3 - 7 timeslots	1	16	23
	π_2 = surgeon	4	83%				
	π_3 = room	4	83%				
Admission	π_1 = patient	8	75%	1 - 2 timeslots	9	72	120
	π_2 = specialist	4	75%				
RehabCenter	π_1 = patient	24	100%	4 timeslots	4	96	20
	π_2 = doctor	12	100%				
	π_3 = physiotherapist	6	100%				
	π_4 = ergotherapist	6	100%				
CardioRehab	π_1 = patient	16	77%	2 timeslots	8	128	104
	π_2 = specialist	10	77%				

The characteristics of the scenarios are described in Table 1. For each scenario, we give the number of resources per property $\pi \in \Pi$, the rate of resources availability, the appointments duration, the number of appointments and the number of timeslots making up the horizon.

From each of these four scenarios, we generated three instances, starting with neither important nor urgent appointments and then increasing the number of essential (Ess) and urgent appointments (Em). We implemented the model under CPLEX and we ran tests on an Intel i5-8350U processor. We limited the computation time to two hours, and we reported the results in Table 2.

Table 2. Results of CPLEX

Instance name	Time	Objective value
$SurgDep\_Ess_0Em_0$	797	0
$SurgDep\_Ess_1Em_1$	2737	5.8160
$SurgDep\_Ess_2Em_2$	2085	19.4235
$CardioRehab\_Ess_0Em_0$	>7200	0-4
$CardioRehab\_Ess_1Em_1$	>7200	12.7059-16.7059
$CardioRehab\_Ess_2Em_2$	>7200	18.8015-23.0589
$Admission\_Ess_0Em_0$	246	0
$Admission\_Ess_1Em_1$	659	4.5098
$Admission\_Ess_2Em_2$	671	9.1734
$RehabCenter\_Ess_0Em_0$	1117	0
$RehabCenter\_Ess_1Em_1$	3015	14.8852
$RehabCenter\_Ess_2Em_2$	6956	23.7496

The objective value column corresponds to the objective function in equation 11. If CPLEX was unable to find an optimal solution within the time limit, we reported the upper and lower bound. The time column is the running time in seconds that CPLEX needs to work out the optimal solution. For all scenarios other than the *CardioRehab* scenario, CPLEX found an optimal solution before the computation time limit. We noticed that increasing the number of important and urgent appointments implied an increase in computing time, except for the scenario *SurgDep*. The calculation time also increases with the number of appointments involved in the scenario.

4 Conclusion & perspectives

In this paper we have described and formalized concrete scheduling problems. We proposed a 0-1 linear programming model able to solve various scenarios in healthcare. We implemented this model under CPLEX and generated some instances in order to test it. The optimality has been reached for most instances and this model has proven to be effective on various scheduling issues in the medical domain. This will be a reliable benchmark to compare different approaches addressing these problems. We plan to develop a genetic algorithm to solve larger instances.

References

Anthony R. N., 1965, "Planning and Control Systems : a framework for analysis", Division of Research, Graduate School of Business Administration, Harvard University.

Baptiste P., P. Laborie, C. Le Pape, C. and W.Nuijten, 2006, "Constraint-Based Scheduling and Planning.", In Foundations of artificial intelligence (Vol. 2, pp. 761-799), Elsevier.

Blazewicz J. , K.H. Ecker, E. Pesch, G. Schmidt, M. Sterna and J. Weglarz, 2019, "Handbook on Scheduling", Springer International Publishing, International Handbooks on Information Systems.

Garey M.R. and D.S. Johnson, 1979, "Computers and Intractability: A Guide to the Theory of NP-Completeness", Series of Books in the Mathematical Sciences, W. H. Freeman.

Hall R. W., 2012, "Handbook of healthcare system scheduling". Springer Science+ Business Media, LLC.

Shnits B., I. Bendavid and Y.N. Marmor, 2019, "An appointment scheduling policy for healthcare systems with parallel servers and pre-determined quality of service", Omega (pp. 102095), Elsevier.

Solving the Multi-mode Resource Investment Problem with Constraint Programming

Patrick Gerhards[1]

Helmut Schmidt University Hamburg, Germany
patrick.gerhards@hsu-hh.de

Keywords: Project Scheduling, Multi-mode Resource Investment Problem, Constraint Programming

1 Introduction

When dealing with project scheduling problems, often a fixed deadline for the project completion time is imposed. The main concern of the project manager is to plan the use of resources in such a way, that the project finishes on time and the project costs are minimised. In this work, we will investigate how we can solve a problem of this kind - the multi-mode resource investment problem (MRIP) - using mixed-integer programming (MIP) and constraint programming (CP) techniques. Since the efficiency of constraint programming solvers increased significantly in recent years, we want to study if they are a suitable procedure when solving this problem type.

2 Multi-mode Resource Investment Problem

The MRIP is a project scheduling problem and an extension of the resource investment problem (RIP). The RIP, also known as resource availability cost problem (RACP), was introduced by Möhring (1984) and has many practical application cases (e.g. software development or construction projects). For the multi-mode variant, Hsu and Kim (2005) presented a priority rule heuristic and Qi et al. (2015) applied a modified version of the particle swarm optimisation metaheuristic to the MRIP. Kreter et al. (2018) tested MIP as well as CP formulations of the RIP (and some of its extensions) and showed that CP techniques work especially well. They solved many of the open instances to optimality.

Next, we give a formal definition of the MRIP. An instance of the MRIP consists of a set of activities $A = \{0, \ldots, n+1\}$ with precedence relations $E \subset A \times A$ among them. The activities are nonpreemptable and for each activity i, there is a set of modes M_i. Depending on the chosen mode $m \in M_i$, the activity processing duration d_{im} can vary. There are two types of resources that are required by the activities in the MRIP: the renewable resources in $\mathcal{R}$ replenish after each period and are useful to model workers or machines. Non-renewables resources $\mathcal{R}^n$ are consumed by activity execution and do not replenish. The amount of required resource units of activity i depends on the mode m and the resource $k \in \mathcal{R}$ ($k \in \mathcal{R}^n$) and is denoted by r_{imk} (r^n_{imk}). For the renewable resources, the peak resource consumption in all periods needs to be lower than or equal to the resource capacity a_k allocated to the project. The renewable resource costs are computed by multiplying the capacity with the resource unit cost factor c_k. Similar, the non-renewable costs are also the product of the resource unit cost factor c^n_k and the total non-renewable resource consumption a^n_k. In the MRIP, there is a also a deadline D given that restricts the project completion. Using forward and backward calculation (Kelley 1963), we can compute bounds on the earliest start (EST_i) and latest finish (LFT_i) times of the activities. The goal is to find a precedence feasible schedule and a mode assignment that minimises the resource costs.

We present a mathematical model for the MRIP using so-called *pulse* variables x_{imt}. It is an adaption of a model for the resource constrained project scheduling problem (Artigues 2017). In preliminary experiments, the model based on *pulse* variables achieved better results than the models with *on-off* or *step* variables. For each activity $i \in A$, mode $m \in M_i$ and period $t \in [EST_i, LFT_i - d_{im}]$ we introduce the binary decision variable x_{imt} that is equal to 1 if and only if activity i is processed in mode m and starts in period t.

Let us now show a model that can be used as a mixed-integer program to solve the MRIP.

$$\min \sum_{k \in \mathcal{R}} c_k \cdot a_k + \sum_{k \in \mathcal{R}^n} c_k^n \cdot a_k^n \tag{1}$$

$$s.t. \sum_{m \in M_i} \sum_{t=ES_i}^{LF_i - d_{im}} x_{imt} = 1 \quad \forall i \in A \tag{2}$$

$$\sum_{m \in M_i} \sum_{t=ES_i}^{LF_i - d_{im}} x_{imt}(t + d_{im}) \leq \sum_{m \in M_j} \sum_{t=ES_j}^{LF_j - d_{jm}} x_{jmt} \cdot t \qquad \forall (i,j) \in E \tag{3}$$

$$\sum_{i \in A} \sum_{m \in M_i} \sum_{t=ES_i}^{LF_i - d_{im}} x_{imt} \cdot r_{imk}^n \leq a_k^n \quad \forall k \in \mathcal{R}^n \tag{4}$$

$$\sum_{i \in A} \sum_{m \in M_i} \sum_{q=\max(ES_i, t-d_{im}+1)}^{\min(t, LF_i - d_{im})} x_{imq} \cdot r_{imk} \leq a_k \quad \forall k \in \mathcal{R}, t = 0, \dots, D \tag{5}$$

$$a_k \geq 0, \quad a_k^n \geq 0 \quad \forall k \in \mathcal{R}^n \tag{6}$$

$$x_{imt} \in \{0,1\} \quad \forall i \in A, \forall m \in M_i, t = ES_i, \dots, LS_i \tag{7}$$

In the objective function (1), we minimise the sum of the resource costs. The renewable part is the product of the peak resource usage a_k and the given resource cost factor c_k. Similarly, we multiply the amount of consumed non-renewable resource units a_k^n with its resource cost factor c_k^n to get the non-renewable resource costs. The constraints (2) ensure that each activity is executed in exactly one mode and exactly one start time is chosen. We model the precedence constraints in a aggregated way and they are displayed in (3): if $(i,j) \in E$, then the finish period of activity i (left side of the inequality) has to be lower than or equal to the start period of activity j (right side). In constraint (4), we compute the non-renewable resource consumption for each non-renewable resource and (5) shows the renewable resource consumption in each time period $t \in [0, D]$. Lastly, (6) - (7) depict the decision variables.

3 Constraint Programming Model

Next, we present a constraint programming model. We use the software IBM ILOG CPLEX CP Optimizer (cf. Laborie et al. (2018)) to model and solve the MRIP using CP-based techniques. The modelling language of CPLEX CP Optimizer offers the use of so-called *interval variables*. They can be used to model the start and finish time of an activity and with the keyword *size*, it is possible to specify the length of the interval (i.e., the difference between the finish and start time). With *optional*, we can declare that an activity can be left unperformed (useful in the context of different modes) and `presenceOf` shows, if an interval variable is performed or not. Let us introduce the decision variables used in our model: With *act*[*i*] we define a interval variable for each activity $i \in A$

(see (14)). Since activities can be performed in multiple modes, we introduce in (15) an optional interval variable $mode[i,m]$ for each mode with a specific duration d_{im}. Among interval variables, we can use time expressions such as `endBeforeStart` to model the precedence restrictions as seen in (10). With the `alternative` expression, we can link the *act* and *mode* interval variables. The constraint $\texttt{alternative}(b, \{b_1, \ldots, b_n\})$ ensures that if interval variable b is present, then exactly one of the interval variables in $\{b_1, \ldots, b_n\}$ is also present and their start and end times coincide. This expression is used in (9) to ensure that we choose exactly one processing mode for each activity. To model the peak resource consumption, we use real valued decision variables a_k and a_k^n (see (13)). We make use of a `cumulative` function named $renewUsage_k$ in (12) to represent the renewable resources. There, we sum up over the resource consumptions of the present interval variables with the so called $\texttt{pulse}(a,h)$ expression (that adds the amount h between the start and end time of interval variable a). The non-renewable resources are modelled by summing up the respective resource consumptions of all present mode interval variables in (11). Finally, in the objective function (8), we sum up the resource costs for the peak resource usage of the renewable and non-renewable resources.

$$\min \sum_{k \in \mathcal{R}} c_k \cdot a_k + \sum_{k \in \mathcal{R}^n} c_k^n \cdot a_k^n \tag{8}$$

$$s.t. \quad \texttt{alternative}(act[i], \{mode[i,m] : m \in M_i\}) \qquad \forall i \in A \tag{9}$$

$$\texttt{endBeforeStart}(act[i], act[j]) \qquad \forall (i,j) \in E \tag{10}$$

$$\sum_{i \in A} \sum_{m \in M_i} \texttt{presenceOf}(mode[i,m]) \cdot r_{imk} \leq a_k^n \qquad \forall k \in \mathcal{R}^n \tag{11}$$

$$renewUsage_k = \sum_{i \in A} \sum_{m \in M_i} \texttt{pulse}(mode[i,m], r_{imk}) \leq a_k \qquad \forall k \in \mathcal{R} \tag{12}$$

$$a_k \geq 0 \quad \forall k \in \mathcal{R} \quad a_k^n \geq 0 \qquad \forall k \in \mathcal{R}^n \tag{13}$$

$$\texttt{interval } act[i] \qquad \forall i \in A \tag{14}$$

$$\texttt{interval } mode[i,m] \textit{ optional size } d_{im} \qquad \forall i \in A, \forall m \in M_i \tag{15}$$

The CPLEX CP Optimizer software features an automatic search that is complete and tunes its parameters automatically. It uses propagation of the temporal network, filtering algorithms for the cumulative resource constraints and large neighborhood search techniques to solve complex scheduling problems (Laborie et al. 2018).

4 Computational Experiments

In order to test the performance of the two approaches presented above, we used the benchmark instances of the RIPLib dataset[1]. It features MRIP instances with 30, 50 and 100 activities, 3 or 6 modes per activity and up to 8 renewable resources. In total, 4 950 instances were used in our experiments. We used version 12.9.0 of the CPLEX CP Optimizer solver to solve the CP model depicted above and Gurobi 9.0.0 to solve the MIP model presented before. The solvers were executed on an Intel Xeon Silver 4214 CPU running at 2.20 GHz and the thread count for each solver was restricted to 1. As a stopping criterion, we used time limits of 60, 600 and 3 600 seconds. In Table 1 we depict the portion of instances that were solved to optimality by the MIP or CP solver. Surprisingly, the CP method solved almost twice as many instances in 60 seconds than the MIP solver in 3 600 seconds. With the 1 hour time limit, CP solved almost one third of the instances

[1] https://riplib.hsu-hh.de/

Table 1. Percentage of instances solved to optimalitiy

	Max runtime in seconds		
Method	60	600	3 600
MIP	0.9 %	4.3 %	9.3 %
CP	14.6 %	22.6 %	28.4 %

to optimality. It shows clearly, that CP outperforms the MIP approach on these instances. Further results will be presented at the conference due to space limitations and can also be found in (Gerhards 2020).

References

Artigues, C., 2017, "On the strength of time-indexed formulations for the resource-constrained project scheduling problem", *Operations Research Letters* , Vol. 45, No. 2, pp. 154-159.

Gerhards, P., 2020, "The multi-mode resource investment problem: a benchmark library and a computational study of lower and upper bounds", *OR Spectrum*, Vol. 42, No. 4, pp. 901-903.

Hsu, C. C., D. S. Kim, 2005, "A new heuristic for the multi-mode resource investment problem", *Journal of the Operational Research Society*, Vol. 56 No. 4, pp. 406-413.

Kelley, J. E., 1963, "The critical-path method: Resources planning and scheduling", *Industrial Scheduling*, Vol. 13, no. 1, pp. 347-365.

Kreter, S., Schutt, A., Stuckey, P. J., Zimmermann, J., 2018, "Mixed-integer linear programming and constraint programming formulations for solving resource availability cost problems", *European Journal of Operational Research*, Vol. 266, No. 2, pp. 472-486.

Laborie, P., J. Rogerie, P. Shaw, P. Vilím, 2018, "IBM ILOG CP optimizer for scheduling", *Constraints*, Vol. 23, No. 2, pp. 210-250.

Möhring, R. H., 1984, "Minimizing costs of resource requirements in project networks subject to a fixed completion time", *Operations Research*, Vol. 32, No. 1, pp. 89-120.

Qi, J. J., Y. J. Liu, P. Jiang, B. Guo, 2015, "Schedule generation scheme for solving multi-mode resource availability cost problem by modified particle swarm optimization", *Journal of Scheduling*, Vol. 18, No. 3 pp. 285-298.

Multi-Scenario Scheduling with Rejection Option to Minimize the Makespan Criterion

Gilenson M and Shabtay D

Department of industrial engineering and management,
Ben-Gurion university of the Negev, Beer-Sheva, Israel
e-mail: gilenson@post.bgu.ac.il, dvirs@bgu.ac.il

Keywords: single-machine scheduling, multi-scenario scheduling, complexity classification, makespan, scheduling with rejection.

1. Introduction and problem definition

We study a set of single-machine scheduling problems, where job-processing times are uncertain at the time-point at which the scheduler has to make his scheduling decisions. We assume that there is a finite set of different scenarios that can affect the processing environment, and thus processing times are scenario-dependent. Scheduling problems with scenario-dependent processing times are mainly studied in the literature under the assumption that all jobs have to be scheduled in shop (see, e.g., Daniels and Kouvelis 1995; Yang and Yu 2002; Aloulou and Croce 2008; Mastrolilli et al. 2013; Choi and Chung 2016; and Kasperski and Zielinski 2016), exposing the manufacturer to a great deal of uncertainty (risk). The common way to control risk in the multi-scenario scheduling literature is to find a robust schedule, which minimizes the maximal value of the scheduling criterion between all scenarios (see, e.g., Daniels and Kouvelis 1995; Yang and Yu 2002; Aloulou and Croce 2008; Mastrolilli et al. 2013; and Kasperski and Zielinski 2016). A different approach to control risk is to reject the processing of some jobs by either outsourcing them or rejecting them altogether.

Although scheduling with rejection and multi-scenario scheduling are two solid fields in the scheduling literature, only Choi and Chung, (2016) consider these two approaches simultaneously. We aim to extend the relevant literature on multi-scenario scheduling with rejection in order to provide the manufacturer tools to coordinate outsourcing with scheduling decision in an uncertain processing environment. In this paper, we focus on single-machine problems with the objective of minimizing the makespan.

The set of problems we study is formally defined as follows: we are given a set $\boldsymbol{J} = \{J_1, J_2, \ldots, J_n\}$ of n independent, non-preemptive jobs that are available for processing at time zero. There is a set of q different scenarios, each of which defines a different possible set of job processing times (we consider both cases where q is a constant and an arbitrary value). By $p_j^{(i)}$ we denote the processing time of job J_j $(j = 1, \ldots, n)$ on the single machine under scenario i $(i = 1, \ldots, q)$. Moreover, by e_j we denote the cost of rejecting job J_j (e.g., outsourcing its processing to a subcontractor).

A solution π=(τ, $\sigma(\boldsymbol{A})$) is defined by (i) a partition $\tau = \boldsymbol{A} \cup \widehat{\boldsymbol{A}}$ of set $\boldsymbol{J}$ into two disjoint subsets $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$ referring to the set of accepted and rejected jobs, respectively; and by (ii) a schedule $\sigma(\boldsymbol{A})$ of the accepted jobs (jobs in set $\boldsymbol{A}$) on the machine. Given a solution let

$$RC = RC(\widehat{\boldsymbol{A}}) = \sum_{J_j \in \widehat{A}} e_j$$

be the total rejection cost. Moreover, let $C_j^{(i)}$ be the completion time of any job $J_j \in \boldsymbol{A}$ on the single machine under scenario i $(i = 1, \ldots, q)$. For a given scheduling criterion G, let $G^{(i)}$ be its value under scenario i $(i = 1, \ldots, q)$, and let $F^{(i)} = G^{(i)} + RC$ for $i = 1, \ldots, q$. We measure the quality of a solution by the following set of q different solution values

$$\mathbf{SV} = \left\{F^{(1)}, \ldots, F^{(q)}\right\}.$$

In this paper we consider the case where G is the makespan criterion, accordingly we have that

$$G^{(i)} = C_{max}^{(i)}(\boldsymbol{A}) = \max_{J_j \in A} \left\{C_j^{(i)}\right\}.$$

As we measure the quality of any solution by q different solution values, many different problem variations can be considered (see, e.g., Gilenson et al. 2018). In this paper, we focus on the following problem variations, which are commonly analysed in the multi-criteria literature:

- Problem Variation 1 (PV_1): Given a set of non-negative parameters, $\theta^{(i)}$ $(i = 1,\ldots,q)$, find a solution, π, that minimizes the linear combination of $F^{(1)},\ldots,F^{(q)}$, i.e., that minimizes $\sum_{i=1}^{q} \theta^{(i)} F^{(i)}$.
- Problem Variation 2 (PV_2): Find a solution, π, that minimizes $F^{(1)}$ subject to $F^{(1)} \leq K_i\ for\ i = 2,\ldots,q$, where K_i is a given upper bound on the value of $F^{(i)}$.
- Problem Variation 3 (PV_3): Identify a single Pareto-optimal solution (also known as a non-dominated or efficient solution) for each Pareto-optimal point, where a solution π is called Pareto-optimal with respect to $F^{(1)},\ldots,F^{(q)}$ if there is no other solution π', such that $F^{(i)}(\pi') \leq F^{(i)}(\pi)$ for $i = 1,\ldots,q$, with at least one of these inequalities being strict. The corresponding Pareto-optimal point is given by $\left(F^{(1)}(\pi),\ldots,F^{(q)}(\pi)\right)$.

We use the standard three-field notation $\alpha|\beta|\gamma$ introduced in Graham et al. (1979) to describe our scheduling problems. The α field describes the machine environment. If $\alpha = 1$, it implies that the scheduling is done on a single-machine. The β field defines the job-processing characteristics and constraints. When considering a multi-scenario scheduling problem we include the set of scenario-dependent parameters in this field. If processing times are scenario-dependent, we include $p_j^{(i)}$ in this field. We also include the rej entry in the β field for cases where rejection is allowed. The scheduling criteria appear in the γ field.

2. Brief literature review

The single-scenario variant of the makespan minimization problem with rejection, i.e., the $1|rej|C_{max}(\boldsymbol{A}) + RC$ problem, was studied by De et al. (1990). They observed that the objective function can be reformulated as

$$C_{max}(\boldsymbol{A}) + RC = \sum_{J_j\in\boldsymbol{A}} p_j + \sum_{J_j\in\widehat{\boldsymbol{A}}} e_j.$$

Therefore, if job J_j is included in $\boldsymbol{A}$, it contributes p_j to the objective function value, and if job J_j is included in $\widehat{\boldsymbol{A}}$, it contributes e_j to the objective function value. Accordingly, they concluded that the following lemma holds:

Lemma 1: The $1|rej|C_{max}(\boldsymbol{A}) + RC$ problem is optimally solvable in $O(n)$ time by applying the following rule for $j = 1,\ldots,n$: If $p_j \leq e_j$ then assign job J_j to set $\boldsymbol{A}$. Otherwise, assign J_j to set $\widehat{\boldsymbol{A}}$.

It follows from the above lemma that the optimal objective value of the $1|rej|C_{max}(\boldsymbol{A}) + RC$ problem is in fact $\sum_{j=1}^{n} min\{p_j, e_j\}$.

To the best of our knowledge, only Choi and Chung (2016) studied a multi-scenario scheduling problem with rejection to minimize the makespan. They studied the $1\left|rej, p_j^{(i)}\right| max_{i\in\{1,\ldots,q\}}\left\{G^{(i)} - G_{opt}^{(i)}\right\}$ problem, where $G_{opt}^{(i)}$ is the optimal (minimal) solution value under scenario i $(i = 1,\ldots,q)$ and

$$G^{(i)} = \sum_{J_j\in\boldsymbol{A}} p_j^{(i)} + \sum_{J_j\in\widehat{\boldsymbol{A}}} e_j.$$

They proved that (i) the problem is ordinary NP-hard even when $q = 2$; (ii) that when q is constant then the problem reduces to the min-max Shortest Path problem with q scenarios and thus admits an FPTAS (Fully Polynomial Approximation Scheme); (iii) that if q is arbitrary then the problem becomes strongly NP-hard; and that (iv) the special case where $p_j^{(i)} = p_j + \alpha^{(i)}$ ($\alpha^{(i)}$ is a scenario-dependent constant that is common to all jobs) is solvable in $O(n\log n)$ time. Moreover, they designed a 2-approximation algorithm for the general problem (with arbitrary q) which is based on LP relaxation.

3. Our results

Consider first PV1, i.e., consider the $1\left|rej, p_j^{(i)}\right|\sum_{i=1}^{q} \theta^{(i)} (C_{max}^{(i)}(\boldsymbol{A}) + RC)$ problem. The fact that

$$\sum_{i=1}^{q} \theta^{(i)} (C_{max}^{(i)}(\boldsymbol{A}) + RC) = \sum_{i=1}^{q} \theta^{(i)} \left(\sum_{J_j\in\boldsymbol{A}} p_j^{(i)} + \sum_{J_j\in\widehat{\boldsymbol{A}}} e_j\right) = \sum_{J_j\in\boldsymbol{A}} \sum_{i=1}^{q} \theta^{(i)} p_j^{(i)} + \sum_{J_j\in\widehat{\boldsymbol{A}}} e_j \sum_{i=1}^{q} \theta^{(i)} = \sum_{J_j\in\boldsymbol{A}} p_j + \sum_{J_j\in\widehat{\boldsymbol{A}}} e_j,$$

where $p_j = \sum_{i=1}^{q} \theta^{(i)} p_j^{(i)}$ for $j = 1,\ldots,n$, implies that the following lemma holds:

Lemma 2: Any instance of the $1\left|rej, p_j^{(i)}\right|\sum_{i=1}^{q}\theta^{(i)}\,(C_{max}^{(i)}(\boldsymbol{A}) + RC)$ problem reduces, in $O(nq)$ time, to an equivalent instance of the $1|rej|C_{max}(\boldsymbol{A}) + RC$ problem by setting $p_j = \sum_{i=1}^{q}\theta^{(i)}\,p_j^{(i)}$ for $j = 1,\ldots,n$.

The following corollary is now straightforward from the results in Lemmas 1 and 2:

Corollary 1: The $1\left|rej, p_j^{(i)}\right|\sum_{i=1}^{q}\theta^{(i)}\,(C_{max}^{(i)}(\boldsymbol{A}) + RC)$ problem is solvable in $O(nq)$ time by applying the following rule for $j = 1,\ldots,n$: If $p_j = \sum_{i=1}^{q}\theta^{(i)}\,p_j^{(i)} \leq e_j$, then assign job J_j to set $\boldsymbol{A}$. Otherwise, assign J_j to set $\widehat{\boldsymbol{A}}$.

We then consider PV2. We show that PV_2 is equivalent to the Multi-dimensional 0-1 Knapsack problem, where the problem parameters may take both negative and positive values. The fact that the less general Multi-dimensional 0-1 Knapsack problem, with non-negative value of parameters, is ordinary NP-hard for any constant number of dimensions, and is strongly NP-hard when q is arbitrary (see, Garey and Johnson, 1979) leads to the following theorem:

Theorem 1: PV_2 and PV_3 are at least ordinary NP-hard for any constant value of q and are strongly NP-hard when q is arbitrary.

Although there is a pseudo-polynomial time algorithm for special cases of the Multi-dimensional 0-1 Knapsack problem with both positive and negative parameters, when the number of dimensions is constant (e.g., for Multi-dimensional 0-1 Knapsack problem with only non-negative parameters, and for Subset Sum problem with both positive and negative parameters), we did not find an evidence for the existence of such an algorithm for our equivalent problem. Therefore, we still had to tackle the question whether PV2 and PV_3 are strongly or ordinary NP-hard when q is constant.

We answer this question by showing that PV_2 and PV_3 are ordinary NP-hard for any constant value of q. We obtain this result by reducing each of the problems (PV_2 and PV_3) to a Multi-Criteria Shortest Path problem, which is solvable in pseudo polynomial time (Hassin 1992, Garroppo et al. 2010).. Therefore, the following theorem holds:

Theorem 2: PV_2 and PV_3 are ordinary NP-hard for any constant value of q.

We then show that our complexity results in Theorems 1 and 2 for PV_2 and PV_3 are also applicable for the absolute robustness problem variation (that is, the problem of finding a solution that minimizes the maximal objective value under all possible scenarios, i.e., that minimizes $\max_{i\in\{1,\ldots,q\}}\{F^{(i)}\}$), leading for the following result as well:

Theorem 3: The absolute robustness problem variation is ordinary NP-hard for any constant value of q and is strongly NP-hard when q is arbitrary.

Finally, we consider a special case of PV_3, where for each job $J_j \in \boldsymbol{J}$ there are only two scenarios of processing times. We provide a fast O($n$log$n$) time algorithm for finding the set of all supported solutions (where a solution is called supported if there exists a set of non-negative $\theta^{(i)}$ parameters ($i = 1,2,\ldots,q$), such that this solution is optimal for PV_1). We note that the set of all supported solutions is a subset of the Pareto-optimal set of solutions.

References

Aloulou M A. and Della Croce F., 2008, "Complexity of single machine scheduling problems under scenario-based uncertainty", Operations Research Letters, Vol. 36(3), pp. 338-342.

Choi B C. and Chung K., 2016, "Min-max regret version of a scheduling problem with outsourcing decisions under processing time uncertainty", European Journal of Operational Research, Vol. 252(2), pp. 367-375.

Daniels R L. and Kouvelis P., 1995, "Robust scheduling to hedge against processing time uncertainty in single-stage production", Management Science, Vol. 41(2), pp. 363-376.

De P., Ghosh J B., and Wells C E., 1991, "Optimal delivery time quotation and order sequencing", Decision Sciences, Vol. 22(2), pp. 379-390.

Garroppo R G., Giordano S., and Tavanti L., 2010, "A Survey on multi-constrained optimal path computation: exact and approximate algorithms", Computer Networks, Vol. 54, pp. 3081-3107.

Garey M R. and Johnson D S., 1979, "Computers and intractability: a guide to the theory of NP-completeness".

Gilenson M., Naseraldin H. and Yedidsion L., 2018, "An approximation scheme for the bi-scenario sum of completion times trade-off problem", Journal of Scheduling, Vol. 22(3), pp. 289-304.

Graham R L., Lawler E L., Lenstra J K. and Kan A R., 1979, "Optimization and approximation in deterministic sequencing and scheduling: a survey", Annals of Discrete Mathematics, Vol. 5, pp. 287-326.

Hassin R., 1992, "Approximation schemes for the restricted shortest path problem", Mathematics of Operations Research, Vol. 17(1), pp. 36-42.

Kasperski A. and Zieliński P., 2016, "Single machine scheduling problems with uncertain parameters and the OWA criterion", Journal of Scheduling, Vol. 19(2), pp. 177-190.

Mastrolilli M., Mutsanas N. and Svensson O., 2013, "Single machine scheduling with scenarios", Theoretical Computer Science, Vol. 477, pp. 57-66.
Yang J. and Yu G., 2002, "On the robust single machine scheduling problem", Journal of Combinatorial Optimization, Vol. 6(1), pp. 17-33.

A Continuous-Time Model for the Multi-Site Resource-Constrained Project Scheduling Problem

Mario Gnägi and Norbert Trautmann

Department of Business Administration, University of Bern, 3012 Bern, Switzerland
mario.gnaegi@pqm.unibe.ch, norbert.trautmann@pqm.unibe.ch

Keywords: Resource-Constrained Project Scheduling, Multiple Sites, Mathematical Programming.

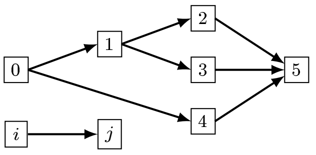
1 Introduction

In the Resource-Constrained Project Scheduling Problem (RCPSP), it is assumed that all project activities are executed at a single site, and consequently all resources are located at this site. In the multi-site RCPSP, which is a novel extension of the single-site RCPSP, it is possible to consider the execution of the activities at several alternative sites. Furthermore, some units of the various renewable resource types are assumed to be permanently located at a single site, whereas the other units can be moved between the sites. Hence, the multi-site RCPSP includes the management of a so-called resource pooling because some resource units can be shared among the different sites. Finally, the spatial distance between the sites gives rise to two different types of transportation times that must be considered in the project schedule. First, while a resource unit is moved between two sites, i.e., during the transportation time, it cannot be allocated to the execution of an activity. Second, if two activities that are interrelated by a precedence relationship are executed at different sites, then a minimum time lag between the completion of the first activity and the start of the second activity must be taken into account, which corresponds to the time required for transporting the first activity's output between the respective sites.

To the best of our knowledge, the multi-site RCPSP has only been treated by Laurent et al. (2017), who provide a binary linear programming (BLP) model. The model belongs to the class of discrete-time models, i.e., the planning horizon is divided into a set of equally-long periods, and it is assumed that an activity can be completed at the end of such a period only. Laurent et al. (2017) report results of a computational analysis performed with CPLEX on a set of small-sized self-generated instances, where the number of activities was varied between 5 and 30. It turned out that within a prescribed time limit of $3'600$ seconds of CPU time, none of the instances with 30 activities could be solved to optimality; therefore, Laurent et al. (2017) additionally propose four different metaheuristics.

Subject of this paper is a novel continuous-time mixed-binary linear programming (MBLP) model for the multi-site RCPSP. Besides an illustrative example, we provide new computational results for instances with 30 activities and a varying number of sites. We have tested the novel continuous-time model and the discrete-time model proposed by Laurent et al. (2017) on a set of 960 instances also generated by Laurent et al. (2017). It has turned out that when using the novel continuous-time model, feasible solutions are devised for all instances; a large number of these instances can even be solved to optimality, and the MIP gap for the remaining instances is relatively small. Moreover, when using the novel continuous-time model, for a considerable number of instances a feasible solution is devised which has a better objective function value than the best solution devised by the discrete-time model and all the metaheuristics presented in Laurent et al. (2017).

The remainder of this paper is organized as follows. In Section 2, we illustrate the multi-site RCPSP by an example. In Section 3, we explain the types of decision variables used



Activity i	0	1	2	3	4	5
p_i	0	2	2	3	2	0
r_{i1}	0	1	1	1	2	0
r_{i2}	0	1	0	0	0	0

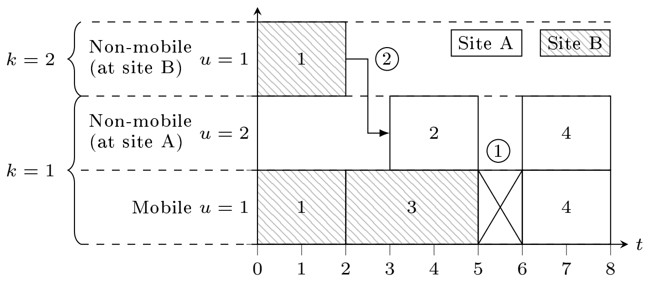


Fig. 1. Illustrative example: duration, resource requirements and activity-on-node network (left); optimal solution (right)

in the novel continuous-time model. In Section 4, we report our computational results. In Section 5, we provide some conclusions and give an outlook for future research.

2 Illustrative Example

In this section, we illustrate the multi-site RCPSP by means of an example project comprising $n = 4$ real activities $\{1, \ldots, 4\}$. Two resource types $k = 1$ and $k = 2$ are required for executing the activities. The project start and the project completion are represented by the fictitious activities 0 and $n+1$; both fictitious activities have a duration of 0 and do not require any unit of any resource type. The activity-on-node network of the illustrative example is depicted on the left-hand side of Figure 1. Each node of this network corresponds to exactly one activity of the project and vice versa, and an arc from a node i to a node j of the network indicates that a precedence relationship is prescribed between the completion of activity i and the start of activity j. The table below provides the duration p_i and the requirements r_{ik} for the two resource types $k = 1$ and $k = 2$ for each activity $i \in \{0, 1, \ldots, 5\}$. Furthermore, we assume that there are two sites A and B; a transport between these sites takes 1 unit of time in any direction. Moreover, we assume that there are two units of resource type $k = 1$; one of these units, e.g. unit $u = 1$, is mobile, i.e., it can be moved between sites A and B, and the other unit $u = 2$ is non-mobile, i.e., it is permanently located at site A. There is one unit of resource type $k = 2$, and this unit is non-mobile and permanently located at site B.

On the right-hand side of Figure 1, an optimal solution for our illustrative example is visualized. For each activity, rectangles indicate to which resource units it is assigned, and during which periods it is executed. First, there is a transportation time between the completion of activity $i = 3$ and the start of activity $j = 4$, because the mobile resource unit $u = 1$ of resource type $k = 1$, which is allocated to their execution, needs to be transported from site B to site A. Second, there is a transportation time between the completion of activity $i = 1$ and the start of activity $j = 2$, because these two activities are precedence-interrelated, but activity $i = 1$ is executed at site B, and activity $j = 2$ is executed at site A, i.e., the output of activity $i = 1$ must be transported from site B to site A.

3 Decision Variables Used in the Novel Continuous-Time Model

In this section, we explain the types of decision variables used in the novel continuous-time model by means of our illustrative example; for a detailed presentation of the model, we refer to Gnägi and Trautmann (2019). The notation used for the sets and parameters is as follows. The set V comprises all activities including the fictitious ones representing the

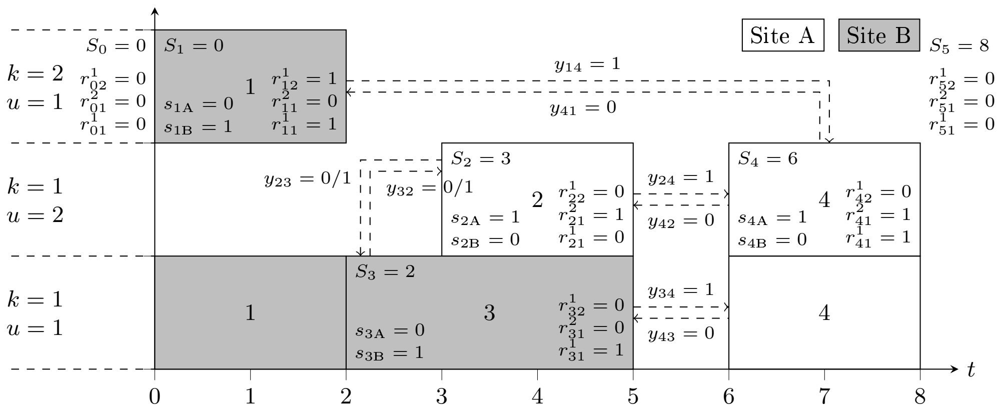


Fig. 2. Types of decision variables used in the continuous-time model

project start and the project completion; we also use a set $\dot{V}$ comprising all real activities only. Furthermore, the set L comprises the alternative sites, and the set R comprises the various resource types; for each resource type $k \in R$, we denote the available number of units as R_k.

The model employs a set of continuous start-time variables S_i $(i \in V)$, together with the following three sets of binary variables:

– site-selection variables s_{il} indicate whether an activity $i \in \dot{V}$ is executed at site $l \in L$,
– resource-assignment variables r_{ik}^u indicate the assignment of an activity $i \in V$ to the various units $u \in \{1, \ldots, R_k\}$ of resource type $k \in R$, and
– sequencing variables y_{ij} indicate the sequences between all pairs of activities $(i,j) \in \dot{V} \times \dot{V}$ $(i \neq j$ and $(i,j) \notin TE)$, where TE denotes the set of pairs of activities that cannot be executed in parallel due to the prescribed precedence relationships.

We illustrate these types of decision variables in Figure 2 by means of the example project introduced in Section 2. By convenience, the project starts at time 0, i.e., $S_0 := 0$.

4 Computational Results

In this section, we report the results of our experimental performance analysis. We tested the performance of both the discrete-time BLP model presented in Laurent et al. (2017), hereafter referred to as LDGN17, and the novel continuous-time MBLP model, hereafter referred to as GT20, on 960 test instances from the set MSj30 for the multi-site RCPSP, each of which consisting of $n = 30$ activities with $|L| = 2$ or $|L| = 3$ sites; these instances have been generated by Laurent et al. (2017) by adapting the well-known single-site RCPSP instances j30 from the PSPLIB (cf. Kolisch and Sprecher 1996). We implemented both models in Python 3.6, and we used the Gurobi Optimizer 8.1 as solver. For each test instance, we prescribed a maximum computation time of 300 seconds, and we limited the maximum number of threads to four. We did not change the default values for any other solver setting.

Table 1 summarizes the computational results. We report the results for all tested instances, but also for the subsets of instances for which both models devised at least a feasible solution. # Feas and # Opt correspond to the number of instances for which a

Table 1. Computational results

# Sites	Subset	Model	# Feas	# Opt	Gap$^{\text{LB}}$ (%)	Gap$^{\text{BKS}}$ (%)	# BKS$^+$	# Vars
2	All	LDGN17	453	268	30.64	13.34	59	8,826
		GT20	**480**	**316**	**13.17**	**0.56**	**121**	**3,187**
2	Feas	LDGN17	**453**	268	30.64	13.34	59	8,873
		GT20	**453**	**313**	**11.23**	**0.57**	**108**	**3,237**
3	All	LDGN17	443	214	45.91	19.75	61	8,858
		GT20	**480**	**277**	**21.09**	**1.56**	**149**	**3,217**
3	Feas	LDGN17	**443**	214	45.91	19.75	61	8,906
		GT20	**443**	**274**	**17.85**	**1.40**	**138**	**3,269**

feasible solution and to the number of instances for which a proven optimal solution, respectively, has been found within the prescribed maximum computation time. Gap$^{\text{LB}}$ (%) corresponds to the average relative gap between the objective function value OFV and the lower bound LB obtained by the solver (calculated as $(OFV - LB)/LB$), and Gap$^{\text{BKS}}$ (%) corresponds to the average relative gap between OFV and the best known solution BKS that has been reported by Laurent et al. (2017) for their proposed metaheuristics (calculated as $(OFV - BKS)/BKS$). # BKS$^+$ corresponds to the number of instances for which a new best solution has been found. Finally, # Vars corresponds to the average number of variables used in the tested models (before Gurobi's preprocessing).

5 Conclusions and Outlook

We studied the multi-site RCPSP, which extends the single-site RCPSP by considering alternative sites for the activities' execution, the management of resource pooling among the sites, and the arising transportation times between the sites. For the studied problem, we have developed a novel continuous-time MBLP model, which turned out to outperform the only known model from the literature with respect to various performance measures on a set of standard test instances.

A promising direction for future research is to apply the novel continuous-time model for a detailed analysis of the advantages of resource pooling in project management.

References

Laurent, A., Deroussi, L., Grangeon, N., and Norre, S., 2017, "A new extension of the RCPSP in a multi-site context: Mathematical model and metaheuristics.", *Computers & Industrial Engineering*, Vol. 112, pp. 634–644.

Gnägi, M., and Trautmann, N., 2019, "A continuous-time mixed-binary linear programming formulation for the multi-site resource-constrained project scheduling problem.", In: Wang, M., Li, J., Tsung, F., and Yeung, A. (eds.): *Proceedings of the 2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Macau, pp. 382–365.

Kolisch, R., and Sprecher, A., 1996, "PSPLIB-a project scheduling problem library.", *European Journal of Operational Research*, Vol. 96(1), pp. 205–216.

Non-dominated sorting genetic algorithm for a bi-objective flexible flow shop problem. A Case Study.

Ibeth Gratz Rodríguez[1], Jose-Fernando Jimenez[1], Eliana María González-Neira[1], Nicolás Eduardo Puerto Ordóñez[2], Yenny Alexandra Paredes Astudillo[3], Juan Pablo Caballero-Villalobos[1]

[1]Industrial Engineering Department, Pontificia Universidad Javeriana, Colombia
E-mail: grattz.i, j-jimenez, eliana.gonzalez, juan.caballero@javeriana.edu.co

[2]Industrial Engineering Department, Universidad de los Andes, Colombia
E-mail: ne.puerto10@uniandes.edu.co

[3]Institucion Universitaria Politecnico Grancolombiano, Colombia
E-mail: yennyparedes@javeriana.edu.co

Keywords: Scheduling, flexible flow shop, genetic algorithm, non-dominated sorting genetic algorithm, total weighted tardiness, total setup cost.

1. Introduction

Production scheduling is one of the most complex tasks for manufacturing industries. It demands to allocate several productions orders or jobs within the machines aiming to optimize a set of KPIs, such as operating, financial and/or sustainable indicators, among many others. Certainly, the scheduling of this jobs could focus in fulfilling a single objective. However, for the last decades, industries have explored the inclusion of multiple objectives for balancing the best possible overall solution. This paper proposes Non-dominated genetic algorithm, for solving a flexible flow shop problem (FFSP) that minimized the total weighted tardiness and total setup cost.

For solving this problem, this paper focuses in the developing a metaheuristic for a case study regarding the production process of a Colombian Soap Factory. The current research considers the following factors: the total resources available, features related to the setup times of the machines, processing times of the products and the due-date requirements of each customer order. Additionally, it is included an extra factor related with the costs that can be reduced as a consequence of the sequencing of jobs.

The manufacturing environment of the Soap Factory is a FFSP with a set of 19 stages (S), each with a single machine but one that contains two unrelated parallel machines. All jobs have the same processing route starting in stage 1 and finishing in stage 19. Nevertheless, due to customizations, some jobs are allowed to skip some stages, depending on its characteristics. Once each product completes the required operation on each stage, it must be added to the queue of the next stage, where it must wait in a buffer of unlimited capacity to be processed. Finally, there are sequence dependent setup times and the processing times are fixed (including the transportation times between stages).

FFSPs have been studied by multiple authors. Yu et al. (2018) studied a FFSP in which there are unrelated parallel machines. Every machine has special characteristics that allow the processing of certain types of products. In order to find a solution of this problem, the use of a genetic algorithm is proposed, in which the effect of three different mutation operators is studied to increase diversification in every iteration. The objective of this study was to minimize the total tardiness. On the other hand, Rabiee et al. (2014) and Ahonen and de Alvarenga (2017) attempt to minimize the makespan in a FFSP. The first authors implemented a hybrid algorithm using an imperialist competitive algorithm, a simulated annealing, a variable neighborhood search and a genetic algorithm; while the second authors performed a comparison between a simulated annealing and a tabu search to find the solution to the stated problem.

Regarding the objective function, it can be said that a variety of studies focus on a single objective analysis; even though, real implications of scheduling problems generally involve more than one objective (Torkashvand et al., 2017). The above, due to the diversity of jobs that are found in the real industry, the processing requirements of each product, as well as the flexibility in terms of delivery to each customer. Consequently, Lassig et al. (2017) minimized both the tardiness and earliness by applying a multiobjective genetic algorithm, in an FFSP. Alternatively, Talbi (2009) showed that algorithms such as SPEA2 and NSGA2 are proper to approximate the pareto optimal set solutions with an acceptable computational complexity and are relatively simple in terms of its implementation.

Finally, in terms of the inclusion of cost analysis in scheduling problems, Yu and Seif (2016) established a genetic algorithm to provide a solution to a flow shop scheduling problem in which the target is to minimize the total tardiness and the total maintenance costs. Rohaninejad et al. (2015) proposed a tabu search algorithm to minimize the sum of the total tardiness cost, the extra time cost and the setup cost. Nevertheless, this research is made for a job shop scheduling problem. It is also found that the analysis of costs in the objective functions of scheduling problems is rarely studied in the literature, even less for a FFSP like the one proposed for sequencing the jobs in JLS Soap Factory.

To the best of our knowledge, there is no evidence of the study of a multiobjective FFSP that considers: the obtention of pareto solutions of total weighted tardiness and total setups costs, the allowing of skipping stages and the consideration of sequence dependent setup times.

2. Proposed solution approach

To solve the stated FFSP an NSGA2 algorithm (Deb et al., 2002) is proposed, using a crossover operator and three mutation operators, which pursue the diversification in the search of solutions throughout the solution space (Peng et al., 2018). Moreover, the application of a complete factorial design is proposed to determine the parameters of the metaheuristic application.

Initially, a population of size N is randomly generated. The population N will serve as parents in the first generation of the algorithm. Each individual of the population is then evaluated in the two objective functions that are going to be minimized. This process is performed in order to compare the solutions and find those that are non-dominated. Once the set of non-dominated solutions are found, they are assigned to a first F1 front. Subsequently, the solutions that were not classified in the F1 front, are compared with each other once again. Then, a second set of non-dominated points are found. This set is again classified in a second F2 front. This procedure is repeated successively until all the Fn fronts are found, and each of the points generated by the individuals of population N are classified.

Additionally, in order to favor the diversity of the solutions found, the density of points surrounding each of the solutions classified in the F fronts is estimated. This procedure is accomplished by calculating the average distance of two points that surround the evaluated solution side by side.

By completing this process, it is possible to order the entire population N of initial parents, first by the non-dominance of the resulting solutions and then, by the density of the points that surround each of the solutions. Afterwards, the selection of the chromosomes that will be crossed is done through a binary tournament. Then, a two-point crossover and three mutation operators named as inverse, insert and swap operation, are used to enhance the diversity of the solutions found.

Finally, the population of chromosomes chosen to be crossed and mutated, and the chromosomes originated from the previous operations, are gathered in a set P, which will be ordered according to the non-dominance of their solutions. This process allows to eliminate the worst individuals until obtaining a population of size N again. The same procedure is performed until the number of iterations defined as a parameter have reached its limit.

A diversity metric is applied to determine the quality of the combination of parameters in terms of the distribution of points along the pareto approach. The equation for the calculation of diversity is shown below:

$$Diversity = \frac{\left(dext1 + dext2 + \sum_{i=1}^{N-1}\left|di - \bar{d}\right|\right)}{\left(dext1 + dext2 + \bar{d}(N-1)\right)} \tag{1}$$

Where $dext1$ and $dext2$ correspond to the euclidean distances of the end points of the generated front, di corresponds to the euclidean distance between two consecutive points and $\bar{d}$ is the average of the euclidean distances of all the points found (Deb et al., 2002). Therefore, it is expected for di to be as close as possible to $\bar{d}$ so as to avoid the concentration of solutions in a single region of the approximated pareto front.

Four parameters are defined in order to analyze their influence in the quality of the solutions found by the algorithm, and consequently, in the result of the diversity metric. These parameters are probability of crossover (*Pcross*), probability of mutation (*Pmut*), initial population N (*P_ini*) and number of iterations (*Iter*).

To estimate the parameters of the algorithm, a value of *Pcross* equals to 0.5 is fixed. As a consequence, this study benefits diversification rather than intensification in the search throughout the solution space. Two levels for each parameter are established for the factorial design, and 30 randomly generated replicates of each combination are completed. *Table 1* shows the factors and their respective levels defined for the experiment.

Table 1. Factors and levels of the factorial design.

	Parameters		
Levels	**P_ini**	**Pmut**	**Iter**
Low	50	0.5	50
High	100	0.9	100

The diversity for each set of solutions generated by the combination of parameters is the response variable of the experiment.

3. Results and discussion

According to the results, only the effect of *Pmut* is statistically significant (P-value <0.05), and there are not interactions between parameters. As a result, in order to favor small values for the diversity metric, a combination of [100, 0.5, 0.5, 100] in the parameters *P_ini*, *Pcross*, *Pmut*, and *Iter* respectively, is chosen to be applied for all instances created to evaluate the metaheuristic. Figure 1 shows the main effects plot and the interactions between factors plot.

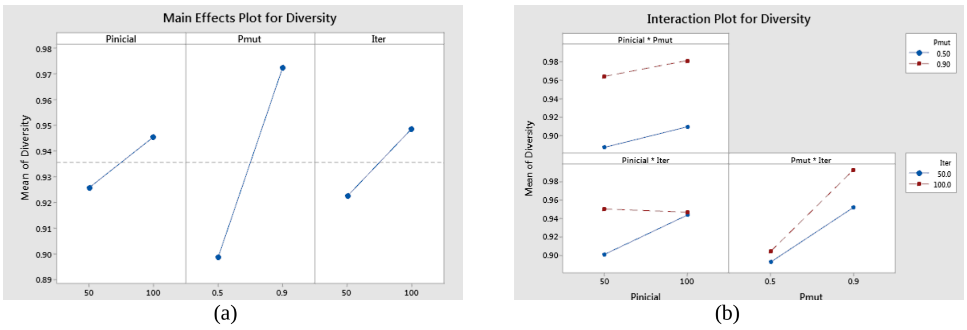


Figure 1. (a) Main effects plot for diversity metric. (b) Interaction plot of the parameters studied in the experiment.

The proposed approach was evaluated in thirty instances in order to assess the approximation of the pareto front obtained with the application of the NSGA2 algorithm. Additionally, the metaheuristic is compared with two priority rules: longest processing time of the jobs (LPT) and shortest processing time of the jobs (SPT). For each of the instances evaluated, it is found that the proposed NSGA2 algorithm can significantly improve the results obtained from the use of LPT and SPT priority rules in both the total setup cost and total weighted tardiness. Finally, it is found that each objective function presents a decreasing trend in each generation of the population, which shows the ability of the metaheuristic to find non-dominated solutions in each iteration.

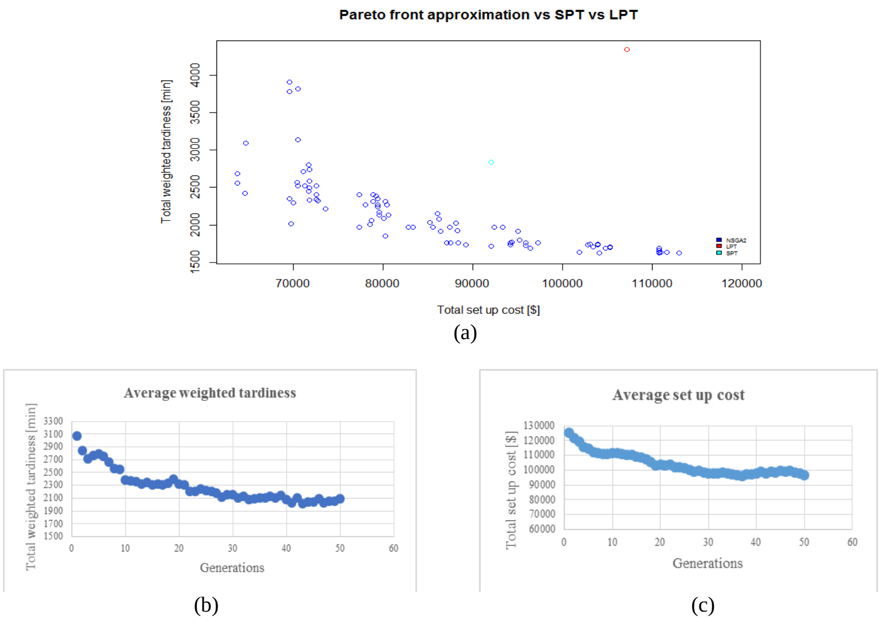


Figure 3. a) Pareto front Approximation, b) total weighted tardiness evolution and c) total setup cost evolution.

References

Ahonen, H. and de Alvarenga, A.G., 2017. "Scheduling flexible flow shop with recirculation and machine sequence-dependent processing times: formulation and solution procedures". The International Journal of Advanced Manufacturing Technology, Vol. 89(1-4), pp.765-777.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T.A.M.T., 2002. "A fast and elitist multiobjective genetic algorithm: NSGA-II". IEEE transactions on evolutionary computation, Vol. 6(2), pp.182-197.

Lassig, L., Mazzer, F., Nicolich, M. and Poloni, C., 2017. "Hybrid flow shop management: multi objective optimisation". Procedia CIRP, Vol. 62, pp.147-152.

Peng, K., Wen, L., Li, R., Gao, L. and Li, X., 2018. "An effective hybrid algorithm for permutation flow shop scheduling problem with setup time". Procedia CIRP, Vol. 72, pp.1288-1292.

Rabiee, M., Rad, R.S., Mazinani, M. and Shafaei, R., 2014. "An intelligent hybrid meta-heuristic for solving a case of no-wait two-stage flexible flow shop scheduling problem with unrelated parallel machines". The International Journal of Advanced Manufacturing Technology, Vol. 71(5-8), pp.1229-1245.

Rohaninejad, M., Kheirkhah, A.S., Vahedi Nouri, B. and Fattahi, P., 2015. "Two hybrid tabu search–firefly algorithms for the capacitated job shop scheduling problem with sequence-dependent setup cost". International Journal of Computer Integrated Manufacturing, Vol. 28(5), pp.470-487.

Talbi, E.G., 2009. Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Torkashvand, M., Naderi, B. and Hosseini, S.A., 2017. "Modelling and scheduling multi-objective flow shop problems with interfering jobs". Applied Soft Computing, Vol. 54, pp.221-228.

Yu, A.J. and Seif, J., 2016. "Minimizing tardiness and maintenance costs in flow shop scheduling by a lower-bound-based GA". Computers & Industrial Engineering, Vol. 97, pp.26-40.

Yu, C., Semeraro, Q. and Matta, A., 2018. "A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility". Computers & Operations Research, Vol. 100, pp.211-229.

An analytical model for budget allocation in risk prevention and risk protection

Xin Guan[1] and Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
xin.guan@ugent.be, mario.vanhoucke@ugent.be
[2] Technology and Operations Management Area, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, UK

Keywords: project risk response, budget allocation, risk prevention, risk protection, analytical model.

1 Introduction

In project risk management (PRM), the underlying risks that would affect the project objectives will be identified and evaluated, and those risks above a certain level are considered intolerable, which need to be mitigated in the process of risk response. Since projects are always executed within a limited budget, it is imperative for project managers (PMs) to reduce project risks in a cost-efficient way. Therefore, the purpose of this study is to propose a method to reasonably allocate budget for mitigating the project risks. Based on the proposed method, the optimal budget allocation decision can be obtained and the effects of the characteristics of the project risk and the risk response on the optimal budget allocation are investigated according to the analytical solutions.

2 Budget allocation in project risk response

In the current practice, project risk is often reduced through prevention policies enforced to reduce the occurrence likelihood of the risk or contingency measures taken to alleviate the negative impacts after the risk occurs. In this study, taking measures before the risk occurs aiming at preventing the risk from happening is defined as *risk prevention*. Implementing actions after the risk occurs aiming at alleviating the negative impacts result from the occurrence of the risk is recognized as *risk protection*. Despite that both risk prevention and risk protection can contribute to the risk reduction in terms of either risk probability or risk impact, any of them entails some cost, and the cost may be different. Thus, to reduce the risk to a certain level, determining the budget allocated for risk prevention and/or risk protection is of great practical need for effective project risk response.

Although budget allocation problems have become a central issue in project risk response, few attempts have been made to solve it. Most of the existing studies related to project risk response focused on the selection of risk response actions (Ben-David and Raz 2001, Fan *et. al.* 2008, Sherali *et. al.* 2011, Dey 2012, Zhang and Fan 2014) rather than the budget allocation problem. They typically assumed the costs and effects of risk response actions are known. This assumption overlooks the complex relations between the costs and effects of risk response actions, so that the generated decision support may be not enough for practical application.

To the best of our knowledge, only two studies have focused on the budget allocation problem in project risk response. Sherali *et. al.* (2008) modelled the possible progressive consequences following the safety risk as an event tree and built an optimization model to lower the losses of these consequences by allocating resources among safety measures. Sato and Hirao (2013) proposed an optimization method to reduce the failure probabilities of project activities and maximize the risk-based project value by allocating the available budget among project activities. Although these two studies deal with the budget allocation problem in project risk response, risk prevention in combination with risk protection are not considered simultaneously. In addition, the characteristics of project risks and risk response strategy are ignored, which may lead to an inferior performance or a misuse of budget resources in project risk response. Therefore, this study will first analyse the costs and effects of risk response with the consideration of the risk and risk response characteristics. Then, an optimization model to obtain an optimal budget allocation is constructed. Finally, some results are drawn from the analytical solutions.

3 Methodology

3.1 Risk response analysis

Since both risk prevention and risk protection can be used to mitigate the risk, the total cost for risk response consists of the cost used for risk prevention and the cost used for risk protection. Generally, the cost for achieving a certain reduction in risk probability or loss depends on the characteristics of the risk and the risk response strategy. Thus, this study models the cost for risk prevention or protection as linear and non-linear functions of these characteristics.

Eqs. (1) - (2) present the linear functions of risk preventive cost and risk protective cost, respectively.

$$q_l = a(P_0 - P), \quad a > 1 \tag{1}$$

$$r_l = b(L_0 - L), \quad 0 < b < 1 \tag{2}$$

with q_l (r_l) the risk preventive (protective) cost in linear relation, P_0 (L_0) the initial risk probability (loss) referred as the risk characteristics, a (b) the unit preventive (protective) cost recognized as the risk response characteristics and P (L) the ex-post probability (loss).

The non-linear functions of the costs for risk prevention and risk protection are shown in Eqs. (3) - (4) (Fan *et. al.* 2008).

$$q_n = \int_{P_0}^{P} c_P dP = a\ln\left(\frac{P_0 - \varepsilon}{P - \varepsilon}\right), \quad a > 0 \tag{3}$$

$$r_n = \int_{L_0}^{L} c_L dL = b\ln\left(\frac{L_0 - \delta}{L - \delta}\right), \quad b > 0 \tag{4}$$

with q_n (r_n) the risk preventive (protective) cost in non-linear relation, ε (δ) the minimum risk probability (loss) after risk prevention (protection) regarded as the risk characteristics, and c_P (c_L) the marginal cost for risk prevention (protection), denoted as $c_P = -\frac{a}{P-\varepsilon}$ and $c_L = -\frac{b}{L-\delta}$.

3.2 Model formulation

Based on the functions of the costs proposed above, a mathematical model for the budget allocation problem (BAP) in project risk response, which aims at minimizing the total cost (Z), is built as below.

$$\textbf{BAP} \quad \text{Minimize } Z = q + r \tag{5}$$

subject to

$$P \cdot L = R \tag{6}$$

$$q, r \geq 0 \tag{7}$$

Constraint (6) ensures the risk can be reduced to an accepted level R. Herein, the accepted level is defined as a percentage of the initial expected loss, namely $R = \mu P_0 L_0$. Thus, μ can be regarded as the requirement for risk response. A small μ implies a strict risk response requirement, which means a large amount of expected loss needs to be reduced. q and r denote the cost for risk prevention and risk protection, respectively. In the linear relation, q (r) equals to q_l (r_l). Otherwise, q (r) equals to q_n (r_n).

4 Results

We plan to analyse the optimality conditions of the model, propose some propositions and proofs, and conduct some computational results, which is still work in process. So far, we have obtained some preliminary results as presented in the followings.

The results with linear relations show that only one risk response strategy, either risk prevention or risk protection, but never both, is required for risk response. The preference for risk prevention or risk protection depends on the values of aP_0 and bL_0, but has no relation with μ, which indicates the decision for using risk prevention or risk protection in project risk response is affected by the characteristics of the risk and risk response. With respect to the budget allocated for risk prevention and risk protection, the results report that the optimal budget allocated to risk prevention has no relation to the initial risk loss and the risk protection characteristics, but increases with the initial risk probability and the unit risk preventive cost, and decreases with the risk response requirement. Similarly, the optimal budget allocated for risk protection increases as the initial risk loss increases, the unit protective cost increases or the risk response requirement becomes loose.

The results with non-linear relations suggest that risk prevention, risk protection or a combination of them can be the optimal option for risk response. Risk prevention is preferred at a high initial probability and a low initial loss, while risk protection is preferred at a low probability and a high loss. When the initial risk probability and loss are taken medium values, both risk prevention and risk protection are required. Regarding to budget amount allocated to risk prevention and protection, the result shows that, if only risk prevention is opted, the budget allocated for it depends on the initial probability, the risk response requirement and the unit preventive cost. Specifically, more budget is required at a strict risk response requirement, a low initial probability or a high unit preventive cost. Similarly, when only risk protection is preferred, more budget is required at a strict response requirement, a low initial loss or a high unit protective cost. The result that a low initial probability (loss) leads to more budget is a little different from that in the linear relations. If both risk prevention and risk protection are selected for risk response, the effects of the risk characteristics, risk response characteristics and the risk response requirements all have effects on the optimal budget allocation, and an interaction among these effects are observed.

To conclude, the characteristics of the project risk and risk response can affect the optimal budget allocation decision, and their effects vary with the relations between the costs and the effects of risk response strategies.

References

Ben-David I., T. Raz, 2001, “An integrated approach for risk response development in project planning", *Journal of the Operational Research Society*, Vol. 52, pp. 14-25.

Dey P. K., 2012, “Project risk management using multiple criteria decision-making technique and decision tree analysis: A case study of Indian oil refinery", *Production Planning & Control*, Vol. 23, pp. 903-921.

Fan M., N. P. Lin and C. Sheu, 2008, “Choosing a project risk-handling strategy: An analytical model", *International Journal of Production Economics*, Vol. 112, pp. 700-713.

Sato T., M. Hirao, 2013, “Optimum budget allocation method for projects with critical risks", *International Journal of Project Management*, Vol. 31, pp. 126-135.

Sherali H. D., E. Dalkiran and T. S. Glickman, 2011, “Selecting optimal alternatives and risk reduction strategies in decision trees", *Operations Research*, Vol. 59, pp. 631-647.

Sherali H. D., J. Desai and T. S. Glickman, 2008, “Optimal allocation of risk-reduction resources in event trees", *Management Science*, Vol. 54, pp. 1313-1321.

Zhang Y., Z. P. Fan, 2014, “An optimization method for selecting project risk response strategies", *International Journal of Project Management*, Vol. 32, pp. 412-422.

Embedded vision systems buffer minimization with energy consumption constraint

Khadija HADJ SALEM[1], Tifenn RAULT[1] and Alexis ROBBES[1]

Université de Tours, LIFAT EA 6300, CNRS, ROOT ERL CNRS 7002, 64 avenue Jean Portalis, 37200 Tours
{khadija.hadj-salem, tifenn.rault, alexis.robbes}@univ-tours.fr

Keywords: Embedded vision systems, Scheduling, Tool Switching Problem, Buffers, Linear Integer Programming, Constraint Programming.

1 Introduction

Designing embedded vision systems involves various optimization problems as scheduling image processing. The limited resources of the devices imply to reduce drastically the computation time, energy consumption, and memory cost. In (Hadj Salem *et. al.* 2018), an image processing scheduling problem is modelized as a variant of the Job Sequencing and tool Switching Problem (SSP). Furthermore, the first results were presented about minimizing the makespan and minimizing the number of switches. However, the buffer minimization is only mentioned and remains unstudied. The SSP is a $\mathcal{NP}$-hard problems that arise from computer and manufacturing systems, see (Catanzaro *et. al.* 2015). This problem involves sequencing a finite set of jobs on a single machine and loading a restricted subset of tools to a magazine with a limited capacity such that the total number of tool switches is kept to a minimum. In our context, the magazine size is the available memory and the number of switches correspond to the energy consumption.

In this paper, we tackle buffer minimization as a new extension of the SSP. This dimensioning problem is called *Prefetch-Constrained Minimum Buffers Problem* (P-C-MBP). Our study provides the first results, an ILP modelization, and a CP approach. We compare these two methods on instances from the literature of SSP and real-world ones. The preliminary results are quite promising and show that the CP performs better than the ILP.

2 The P-C-MBP Statement

The P-C-MBP involves scheduling a number of output tiles (jobs) on a single machine, under a limited number of prefetches (number of tool switches), such that the resulting number of buffers (magazine slots) is kept to a minimum. This can be formalized as follows: let a P-C-MBP instance be represented by a 4-tuple, $I = (X, Y, \mathcal{R}_y, N)$ where:

- $\mathcal{X} = \{1, \dots, X\}$ is the set of X input tiles to be prefetched from the external memory to the internal buffers;
- $\mathcal{Y} = \{1, \dots, Y\}$ is a set of Y independent non-preemptive output tiles (also called tasks) to be computed;
- $(\mathcal{R}_y)_{y\in\mathcal{Y}}$ is the X-dimensional column vector, where $\mathcal{R}_y \subseteq \mathcal{X}$, which defines the set of required input tiles (called prerequisites or tools). These $\mathcal{R}_y$ tiles have to be prefetched from the external memory and must be present in the buffers during the whole corresponding computation step.
  In the same way, we denote by $(\mathcal{R}_x)_{x\in\mathcal{X}}$ the Y-dimensional row vector, where $\mathcal{R}_x \subseteq \mathcal{Y}$, which defines the set of used output tiles for each input tile x.

– N is an integer non-negative number of prefetches, i.e., each input tile can be loaded more than one time. It is assumed that $X < \sum_{y \in \mathcal{Y}} |\mathcal{R}_y|$, otherwise the problem is trivial.

The solution to such an instance is a sequence $y_1, \ldots, y_Y$ determining the order in which the output tiles are computed (executed), and a sequence $(x, z)_1, \ldots, (x, z)_N$ of prefetch configurations determining which input tiles are prefetched (loaded) in which buffers (magazine slots) at a certain time.

The function objective of P-C-MBP is to minimize the number of buffers, denoted by Z.

We proved that the P-C-BMP problem is $\mathcal{NP}$-hard by showing that if an algorithm exists for solving the P-C-BMP, it can be called iteratively a polynomial number of times to solve the SSP problem. As a consequence, if P-C-BMP was polynomial, so would SSP.

3 A position-based Integer Linear Programming

Due to the relation between the P-C-BMP and the SSP, we can draw inspiration from the existing SSP's ILPs. In fact, (Catanzaro *et. al.* 2015) proposed and compared several ILPs with different categories. In our study, we focused on a position-based ILP, which is very similar to the standard one given by (Tang and Denardo 1987), to which strengthening inequalities are added.

For all $y \in \mathcal{Y},\ j \in \mathcal{Y}$ and $x \in \mathcal{X}$, we define three sets of binary variables: c_{yj} is equal to 1 if output tile y is computed at position j and 0 otherwise. Similarly, e_{xj} equal to 1 if input tile x exists in buffer at position j (is already loaded) and 0 otherwise. Finally, p_{xj} is equal to 1if input tile x has just been prefetched at position j and 0 if it was already loaded. Then, a valid formulation for the P-C-BMP is the following:

$$\min\ Z$$

Subject to

$$\sum_{j \in \mathcal{Y}} c_{yj} = 1, \forall\ y \in \mathcal{Y} \quad (1)$$

$$\sum_{y \in \mathcal{Y}} c_{yj} = 1, \forall\ j \in \mathcal{Y} \quad (2)$$

$$\sum_{x \in \mathcal{R}_y} e_{xj} \geq |\mathcal{R}_y| * c_{yj}, \forall\ y \in \mathcal{Y},\ j \in \mathcal{Y} \quad (3)$$

$$p_{x1} = e_{x1}, \forall\ x \in \mathcal{X} \quad (4)$$

$$p_{xj} \leq e_{xj}, \forall\ x \in \mathcal{X},\ j \in \mathcal{Y} \backslash \{1\} \quad (5)$$

$$p_{xj} \leq 1 - e_{xj-1}, \forall\ x \in \mathcal{X},\ j \in \mathcal{Y} \backslash \{1\} \quad (6)$$

$$p_{xj} \geq e_{xj} - e_{xj-1}, \forall\ x \in \mathcal{X},\ j \in \mathcal{Y} \backslash \{1\} \quad (7)$$

$$\sum_{x \in \mathcal{X}} \sum_{j \in \mathcal{Y}} p_{xj} \leq N \quad (8)$$

$$\sum_{x \in \mathcal{X}} \sum_{j \in \mathcal{Y}} p_{xj} \geq |\mathcal{X}| \quad (9)$$

$$\sum_{j \in \mathcal{Y}} p_{xj} \geq 1, \forall\ x \in \mathcal{X} \quad (10)$$

$$\sum_{j \in \mathcal{Y}} p_{xj} \leq |\mathcal{R}_x|, \forall\ x \in \mathcal{X} \quad (11)$$

$$Z \geq \sum_{x \in \mathcal{X}} e_{xj}, \forall\ j \in \mathcal{Y} \quad (12)$$

$$Z \geq \max_{y \in \mathcal{Y}} |\mathcal{R}_y|, \forall\ y \in \mathcal{Y} \quad (13)$$

$$Z \leq |\mathcal{X}| \quad (14)$$

$$c_{yj} \in \{0,1\}, \forall\ y \in \mathcal{Y},\ j \in \mathcal{Y} \quad (15)$$

$$e_{xj}, p_{xj} \in \{0,1\}, \forall\ x \in \mathcal{X},\ j \in \mathcal{Y} \quad (16)$$

The objective function minimizes the number of buffers defined by Z, under constraints (1) — (16). Constraints (1) — (2) are a set of standard assignment constraints. Constraints (3) are the requirement constraints that define the relation between prefetches and computations steps. Similarly, constraints (4) — (7) define the relation between the loading and prefetching of input tiles. Constraint sets (8) — (9) and (10) — (11) give bounds on prefetches number and on prefetches number for each input tile $x, \forall x \in \mathcal{X}$, respectively. In the same way, constraints (12) compute a lower bound of Z in relation to the prefetches number for each computation position $j, \forall j \in \mathcal{Y}$, while constraints (13) — (14) give an upper/lower bounds on buffers number. Finally, constraints (15) — (16) set the ranges of the variables.

To strengthen the ILP given before, we provide two valid inequalities (17) — (18), in which constraint (17) ensures that the first computation must take place during the first half of the possible position, while constraints (18) gives a lower bound on the number of times in which input tiles must exist in the buffer.

$$\sum_{j=1}^{|\mathcal{Y}|/2} c_{1j} = 1 \qquad (17) \qquad\qquad \sum_{x\in\mathcal{X}}\sum_{j\in\mathcal{Y}} e_{xj} \geq \sum_{y\in\mathcal{Y}} |\mathcal{R}_y| \qquad (18)$$

4 A Constraint Programming Approach

To the best of our knowledge, it appears that the *constraint programming* paradigm (CP) has been little considered in the SSP's literature. Thus, we introduce here a new CP model for the P-C-MBP using concepts present in IBM ILOG CP Optimizer.

We first define two sets of variables as follows:

- I_y: the interval variable for each output tile (task) $y \in \mathcal{Y}$.
- $I_{x,j}$: the interval variable for each input tile (prerequisite) $x \in \mathcal{X}$, and for each interval number $j \in \{1, \ldots, j_{\max}\}$, where $j_{\max} = N - |\mathcal{X}| + 1$. If $j = 1$ this variable is mandatory, while for $j > 0$ the interval is optional. This means that an input tile x is loaded at least once, and possibly more.

We also use the following *Cumul functions*:

- $\texttt{nbXScheduled} = \sum_{x\in\mathcal{X}, j\in\{1,\ldots,j_{\max}\}} \texttt{StepAtStart}(I_{x,j}, 1);$
- $\texttt{nbOverlap} = \sum_{x\in\mathcal{X}, j\in\{1,\ldots,j_{\max}\}} \texttt{Pulse}(I_{x,j}, 1);$
- $\texttt{necessaryTiles}\ [y] = \sum_{x\in\mathcal{R}_y, j\in\{1,\ldots,j_{\max}\}} \texttt{Pulse}(I_{x,j}, 1), \forall y \in \mathcal{Y}.$

The objective is to minimize the buffers number Z, subject to the following constraints:

$$\texttt{noOverlap}(\mathtt{I_y}) \quad \forall\, y \in \mathcal{Y} \qquad (19)$$

$$\texttt{nbOverlap}(\mathtt{I_{x,j}}) \quad \forall\, x \in \mathcal{X}, \forall j \in \{1, .., j_{\max}\} \qquad (20)$$

$$\texttt{nbXScheduled} \leq N \qquad (21)$$

$$\texttt{nbOverlap} \leq Z \qquad (22)$$

$$\texttt{alwaysIn}(\texttt{necessaryTiles}[\mathrm{y}], \mathtt{I_y}, |\mathcal{R}_\mathrm{y}|, |\mathcal{R}_\mathrm{y}|) \quad \forall y \in \mathcal{Y} \qquad (23)$$

5 Computational Experiments

To evaluate the different approaches, we present the first numerical results. All experiments were performed on an Intel Core i5 processor, a 2.67 GHz machine, equipped with 4 GB of RAM and operating system Windows. The ILP is solved by CPLEX and the CP approach is implemented using IBM ILOG CP Optimizer. The CPU time limit for each run on each problem instance is 300 seconds. The number of prefetches N is set to $|\mathcal{X}|$, which is its smallest possible value.

Experiments were made using two kinds of data-sets possessing different characteristics. We first considered a collection of 16 data-sets for the well-known SSP, downloadable at http://www.unet.edu.ve/∼jedgar/ToSP/ToSP.htm. We then considered a set of 10 benchmarks from real-life non-linear image processing kernels for MMOpt software already used in (Hadj Salem *et. al.* 2018). These instances are reduced by using a dominance property, which removes each output tile that needs a subset of input tiles, which is already used at least once by another output tile.

In our preliminary tests on SSP data-sets (instances goes from size 9×10 to 40×60), both ILP and CP models were able to solve within the time limit optimally, only instances with (X, Y) under $(15, 10)$ input/output tiles. In contrast, they give similar results for the other instances, except for few ones for which the CP is a bit better.
In the case of bigger instances of MMOpt (greater than 64×64 input/output tiles), Table 1 gives the detailed numerical results of ILP and CP models. In this table, the **Gap(%)** is computed as follow: $100 * (Z_{\text{ILP}} - Z_{\text{CP}})/Z_{\text{CP}}$.

Table 1. Numerical results for ILP & CP using MMOpt instances

Instance	$\lvert\mathcal{X}\rvert$	$\lvert\mathcal{Y}\rvert$	$Z_{\textbf{ILP}}$	**time**$_{\textbf{ILP}}$	$Z_{\textbf{CP}}$	**time**$_{\textbf{CP}}$	**Gap(%)**
test_2D_0	256	64	**4**	101.39	**4**	0.09	**0**
test_2D_1	64	256	**1**	2,5	**1**	0,11	**0**
test_polaire_0	146	90	146	300	68	300	114.7
test_polaire_1	80	60	80	–	46	–	73.91
test_polaire_2	244	82	244	–	142	–	71.83
test_fisheye_0	176	103	176	–	109	–	61.46
test_fisheye_1	224	103	224	–	139	–	61.15
test_fisheye_2	360	103	360	–	230	–	56.52
test_fd_0	429	300	429	–	349	–	22.92
test_fd_1	2272	206	2272	–	2081	–	9.17

As illustrated in Table 1, we can see that both ILP and CP models can solve only the first two instances. For the rest, the CP model is definitely better than ILP and provides relatively good upper bounds.

Overall, the ILP model fails to find good bounds, specifically in the case of MMOpt instances. On the other hand, the CP model seems to be able to handle well this kind of problem. Nonetheless, extensive tests, some improvements, and comparison with other mat-heuristics should be considered.

References

Catanzaro D., L. Gouveia and M. Labbé, 2015, "Improved integer linear programming formulations for the job Sequencing and tool Switching Problem", *European Journal of Operational Research*, Vol. 244, pp. 766–777.

Hadj Salem K., Y. Kieffer and M. Mancini, 2018, "Meeting the Challenges of Optimized Memory Management in Embedded Vision Systems Using Operations Research", *Recent Advances in Computational Optimization: Results of the Workshop on Computational Optimization WCO 2016*, pp. 177–205.

Tang C.S., E.V. Denardo, 1987, "Models arising from a flexible manufacturing machine, Part I: Minimization of the number of tool switches", *Operations Research*, pp. 767–777.

Multi-project scheduling problems with shared multi-skill resource constraints

Meya HAROUNE[1,2], Cheikh DHIB[2], Emmanuel NERON[1], Ameur SOUKHAL[1],Hafedh MOHAMED BABOU[2] and Mohamedade NANNE[2]

[1] Laboratoire d'Informatique Fondamentale et Appliquée de Tours LIFAT EA 6300 ROOT ERL-CNRS 7002, France(2)
meya.haroune@etu.univ-tours.fr,{emmanuel.neron, ameur.soukhal}@univ-tours.fr
[2] Unité de recherche Documents Numériques et Interaction de l'Université de NouakchottAl-Asriya DNI, Mauritanie
dhib.cheikh@iscae.mr, hafedh.mohamed-babou@esp.mr, farouk@una.mr

Keywords: Operational research, Project scheduling, Personnel constraints, Skills management, ILP, Heuristics, Tabu search.

1 Introduction

Companies are increasingly concerned by the organization of their projects, taking into account skills of their teams, in order to answer to customer requests. Thus, in this context, several project managers can compete for renewable resources (human or machines) to carry out their projects and to avoid delay that may be expensive (Dhib *et. al.* (2016)).

In this paper, we study a problem of multi-project scheduling where several project managers manage one or more projects. Projects share the same resources. Shared resources are human teams available with limited capacities and various skills. Our aim is to propose to the project managers a schedule of their activities within a fixed time horizon. This schedule must take into account the completion time of each phase of the project to reduce the weighted cost of realization of the projects (payment of the penalties). Our goal is to study a relevant industrial model, by integrating the particularities and the characteristics of persons and activities, as realistic as possible.

Project scheduling problems have been widely studied and the dedicated literature on this subject is very important. The most well known are RCPSP (for Resource Constrained Project Scheduling Problem) Artigues *et. al.* (2008), the MM-RCPSP (for MulimodeProjects Scheduling Problem) Bianco *et. al.* (1998), MSPSP (for Multi-skill Project Scheduling Problem)Bellenguez-Morineau (2006). Some other works focused on scheduling in a multi-project context. Concerning the studied problem, it has been introduced by Néron *et. al.* (2011), that corresponds to a real case of multi-project planning. This problem can be encountered in IT companies.

In the next section, the studied problem and the used notations are formally presented.

2 Problem description

Multiple projects are running simultaneously and share common human resources. Each project $K_{l,l=1...,L}$ consists of a set of independent and preemptive activities. An activity J_i is characterized by an estimated load p_i expressed in day. There are release dates r_i, due dates d_i and late penalties w_i expressed in week.

Each person M_j, has a quota per project $Q_{j,l}$, determining his/her maximum rate of participation to the project K_l. The periods of availability per week of each person, are known and are divided between the different projects in proportion to the rate of his/her intervention. Each resource has a skill level for performing an activity. Thus, the processing time (nominal load) of an activity is computed according to the efficiency of the person in charge of its processing. It is defined for activity J_i and person M_j as follows: $p_{i,j} = (2 - v_{i,j})p_i$, where $v_{i,j}$ is the given efficiency level of resource M_j in carrying out the activity J_i, $0 \leq v_{i,j} \leq 1$. Durations and availabilities are measured by a half of a day as time unit.

An activity is assigned to a single person throughout its realization, even if it is spread over several weeks. At a time t, a person M_j can work only on one activity. The number of activities on which a person may be assigned during the same week can not exceed the given value b_j. So, this limitation reduces the number of context changes and thus increases the efficiency of the person. However, a minimum (resp. maximum) load C_i^{min} (resp. C_i^{max}) is defined for each activity J_i to frame the quantities of its realization per week. One solution to this problem is to check whether there is a possible schedule within the fixed time horizon of persons to different project activities. The schedule of different activities within the week is not considered here. We are interested in determining the time spent per week and per person on each activity while respecting the constraints mentioned above and minimizing the total weighted tardiness, $Z = \sum w_i T_i$, where T_i is the number of weeks of delay of activity J_i.

The originality of this model is the consideration of the notion of skill and the minimum/maximum activities loads. This problem is $\mathcal{NP}$-hard since the activity scheduling problem on uniform parallel machines noted $Qm \mid r_i, pmtn \mid \sum w_i T_i$ known as $\mathcal{NP}$-hard, is a special case of the studied problem (Brucker *et. al.* (1997)).

Example: Let's consider two projects (K_1, K_2) to be planned over a three-week horizon. Two persons (M_1, M_2) work on these projects. The data are given in Table 1. In this table we particularly specify the project to which the activities belongs, resources that are able to perform the activity and the efficiencies of the resources to perform an activity. Table 2 presents the characteristics of the resources, represented by their weekly availability and their quotas per project.

Table 1. Characteristics of activities

		J_1	J_2	J_3	J_4
Interval $\{r_i, d_i\}$		{1,3}	{2,3}	{2,2}	{1,2}
Load (p_i)		8	4	7	6
C_i^{max}		6	6	5	8
Penalty(w_i)		10	20	15	10
Project		K_1	K_1	K_2	K_2
Skill		$\{M_1,M_2\}$	$\{M_2\}$	$\{M_1\}$	$\{M_1,M_2\}$
Efficiencie $v_{i,j}$	M_1	1.0	-	0.5	0.0
	M_2	0.5	1.0	-	1.0

Table 2. Characteristics of resources

	M_1	M_2	
W_1	8	6	Availability
W_2	10	10	
W_3	5	8	
K_1	50%	70%	Quotas
K_2	60%	60%	

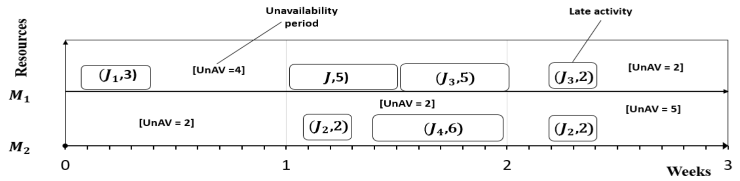


Fig. 1. Example of feasible solution

Figure 1 presents a feasible solution. Only the activity J_3 is late, i.e. a one-week delay ($T_3 = 1$). Hence, the total cost due to this delay is 15.

3 Heuristics approaches

To optimally solve the studied problem, an integer linear programming is proposed by Meya *et. al.* (2019). This ILP solves small and medium sized instances up to 40 activities over 8 weeks in reasonable time. To solve large size instances two heuristics are developed: a local search algorithm and a tabu search algorithm.

3.1 2-phases heuristics

The 2-phase algorithm operates in 2 phases: With the first phase, we seek to build an initial solution, i.e. a possible assignment of operators to the different activities of each project. For this purpose, we use a combination of a priority rule (to determine which activity should be considered first) and algorithm developed to solve bin packing problem (to determine which person should execute the current activity). Different priority rules have been tested: weighted earliest due dates WEDD; shortest (resp. longest) processing time SPT (LPT) where the efficiency coefficients to perform activity is taken into account. To assign activities to persons, two known algorithms are considered: First-fit algorithm (the first operator found with availability large enough to process the activity is chosen); Best-fit algorithm (the operator with the smallest large enough availability on project of the activity is chosen). The preliminary results show that the combination of WEDD and Best-fit algorithm outperforms all other heuristics. In the second phase, with a given assignment, we should determine a compatible schedule of each person over time. In this second phase, we should solve $\mathcal{NP}$-hard problem. So, heuristic based on a maximum flow with minimum cost (MF-MC) is proposed. The graph is defined by three levels of nodes plus two dummy nodes. Note that this graph is built by person (see figure 2 where 3 activities belong to 2 projects are assigned to one operator).

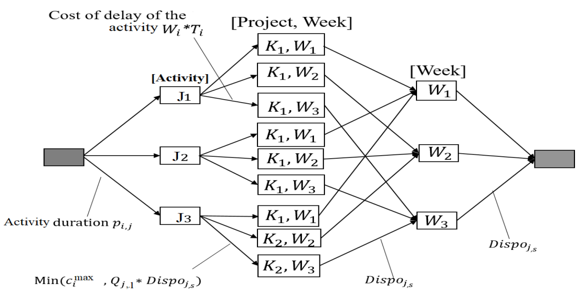


Fig. 2. Max Flow-Min Cost: 3 activities assigned to one person

In order to minimize the total weighted tardiness, we add costs on the edges between 'Activity' node and node 'project, week'. This cost represents the cost of delay of the activity multiplied by the index of the week.

3.2 Local search algorithm

Two neighborhood functions have been developed to improve the assignment of activities. The first function $V1$ tries to reassign late activities to other person, where the second function $V2$ tries to free up time for a person performing a late activity (an activity on-time can be reassigned to another person). To determine a compatible schedule of each person over time, we recall MF-MC algorithm.

3.3 Tabu search algorithm

The basic idea of our developed tabu search is described as follow. Initial assignment of activities to operators is given by the best 2-phases heuristic. Two neighbourhood operators are used: the first one is $V1$. According to the second neighbourhood operator, denoted $V3$, late or incomplete activity is switched with another activity. Swapping is only possible if the execution time of these two activities overlaps. The best solution is kept as a starting point for the next iteration. At each iteration, the number of neighbors generated depends on the quality of the solution selected to be the current solution. If each time the current solution is improved, the neighborhood size is expanded. Otherwise, if after a certain number of iterations, the found best solution is not improved, a diversification strategy is applied. This strategy consists in replacing a current solution with another one generated by applying one of the priority rules previously introduced. We then obtain a new assignment of activities to operators. The tabu search is therefore restarted with this new assignment.

3.4 Experimental results

The generated instances are inspired by a practical case: 2 and 3 projects with 60 and 90 activities per project and between 10 and 15 persons. These projects run simultaneously over a horizon of 12 and 14 weeks. The experiments were performed using Cplex 12.8.0 solver where the calculation time is limited to 3600sec.

Tables 4 and 5 summarize the obtained results where the first column represents the percentage of instances for which heuristics found a feasible solution (reminder that each project should be performed during a given period); The second column gives the percentage of the instances that are optimally solved by heuristics; The third column shows the average deviation of instances not solved to optimal and the last column gives the execution time in seconds. All these values correspond to 34 instances for which ILP found feasible solutions within 3600sec.

Table 3. Local search approach (LS)

Method	% Feas	% Opti	% GAP	Total GAP	Time (m)
InitAssignment	36	22,4	58,4	45	0,06
LS with V1	69	46,2	29	15,3	1,2
LS with V2	64,4	47	23,3	14,5	1,8
LS with V1 & V2	86,3	64,5	22,5	7	3,1

Table 4. Tabu search algorithm (TS)

Method	% Feas	% Opti	% GAP	Total GAP	Time (m)
InitAssignment	36	22,4	58,4	45	0,06
TS with V1	82	53	22,1	13,4	1,5
TS with V3	64,5	59,5	20,1	11,7	3,6
TS with V1 & V3	92,8	73	15	**4,6**	5,2

We can see in both tables that the neighborhood operator V1 is more efficient compared to the other neighborhood operators V2 and V3. We note that TS with V1 and V3 outperforms all other heuristics. Over the 34 instances, the mean deviation is at most 4,6% (see total Gap column where 73% of instances are optimally solved). Over the 27% of the remaining instances, the mean deviation from the optimal solution given by TS with V1 and V3 is at most 15%.

References

Brucker P., B. Jurisch and A. Krämer, 1997, "Complexity of scheduling problems with multipurpose machines", *Annals of Operations Research*, Vol. 70, pp. 57-73.

Dhib Ch., A. Soukhal and E. Néron, 2016, "Solving Methods for a Preemptive Project Staffing and Scheduling problem", *Handbook on Project Management and Scheduling*, Vol. 1&2, Springer.

Artigues C., S. Demmasey and E. Néron, 2008, "Resource-constrained project scheduling problem: Models, Algorithms, Extension and Applications", *Control systems, robotics and manufacturing series*, Wiley.

Bianco, P. Dell'Olmo and M.G. Speranza, 1998, "Heuristics for multi-mode scheduling problems with dedicated resources", *European Journal of Operational Research*, Vol. 107(2), 260-271.

Bellenguez-Morineau 0., 2006, "Méthodes de résolution pour un problème de gestion de projet avec prise en compte de compétences", Thèse de doctorat de l'université de Tours.

Néron E., A. Soukhal, R. Semur, F. Tercinet and S. Bouamer, 2011, "planification de projets -modèle et méthode de résolution", ROADEF(2011).

Haroune M., Dhib Ch., Néron E., A. Soukhal, H. Babou, and F. Nanne, 2019, "Ordonnancement multi-projets à contraintes de ressourcespartagées multi-compétences", ROADEF(2019).

Solving large, long-horizon resource constrained multi project scheduling problems with genetic algorithms

Brendan Hill[1], Adam Scholz[1], Lachlan Brown[1], and Dr Ana Novak[2]

[1] School of Mathematics and Statistics, University of Melbourne, Australia
brendan.hill@gmail.com, adam.scholz@unimelb.edu.au,
l.brown25@student.unimelb.edu.au
[2] Joint and Operations Analysis Division, Defence Science and Technology Group, Australia
ana.novak@dst.defence.gov.au

Keywords: RCMPSP, priority rules, genetic algorithms, large-scale/long-horizon.

1 Introduction

In this paper we adapt the *priority rule* based approach to the *Resource Constrained Multi-Project Scheduling Problem* (RCMPSP) (Villafáñez, F.A. *et. al.* 2018) for large-scale, long-horizon scheduling problems, combining the expressiveness of the priority rule framework with genetic algorithms (GA) to overcome the limitations of current methods. An efficient scheduling algorithm is described which runs in approximately linear time, enabling the use of sampling-intensive search methods in the priority rule space.

Although the algorithm can be applied to any long horizon scheduling problem, we evaluate the method in the context of a simulation of recruits flowing through military training schools. Recruit training can take years, and ensuring sufficient supply of the right people at the right time requires planning over a long horizon. Integral to this planning effort is an understanding of the yearly capacity of each training facility in progressing students through the relevant training syllabus.

We find that the the problem can be usefully framed as a RCMPSP where lessons are activities, and infrastructure and instructor specifications are resource requirements, and so the long range optimisation of training activities can be defined in project scheduling terms. Previous work in this area by the research team involved randomly sampling priorities and heuristic based optimisation of training activities along with ongoing work with Mixed Integer Linear Programming (MILP). The RCMPSP is NP-hard as it is a generalisation of the Resource Constrained Project Scheduling Problem (RCPSP) (Blazewicz, J. *et. al.* 1983). Current approaches tend to focus on heuristic methods, in particular, priority rule based methods such as found in Vázquez, E.P.*et. al.* (2013); Kurtulus IB. and Davis, E.W. (1982). Here, we adopt this representation and apply search methods over the space of possible priority rules to optimise over long-horizon objective, and find that a *random key genetic algorithm* ("RKGA") method outperforms baseline methods considered.

2 Problem formulation

A training school defines a syllabus for each student consisting of lessons of varying durations, dependencies on previous lessons and resource requirements (both infrastructure and instructor based). We define each individual student as a *project* and each of their lessons as an *activity*, possibly dependent on earlier activities, and specifying local *resource requirements* which must be satisfied in accordance with the global resource constraints of the training facility. We model over $Y = 20$ years where $\varDelta = 20$ students per year are introduced, spread evenly throughout the year, defining the *release time* of each initial activity in each project.

The aim is to maximize average yearly throughput of students over the horizon, i.e. average number of *projects completed per year*. As such we adopt as our primary objective measure z_1 the number of projects completed within the horizon $\mathcal{G}(h)$ for some feasible schedule h. Additionally we consider the degree of partial completion of remaining projects $\overline{\mathcal{G}}(h)$ and hence the average remaining progress, z_2 is added. Since $z_1 \in \mathbb{N}$ and necessarily $z_2 \in [0, 1)$ this forms a lexicographical ordering within the multi-objective function:

$$\max_h z(h) = z_1 + z_2 = |\mathcal{G}(h)| + \frac{1}{|\overline{\mathcal{G}}(h)|} \sum_{p \in \overline{\mathcal{G}}(h)} \frac{\#\text{ Activities completed in project } p}{\text{Total activities in project } p} \qquad (1)$$

The modelling involves 20 projects per year over 20 years, each with 500 activities resulting in 200000 activities in total. We assumed 250 active training days per year with 10 hours periods of 30 minute blocks for a total of 100000 discrete time steps. Activities which could not be scheduled within the horizon were discarded.

3 The standard Priority Rule Scheduling Algorithm (PRSA)

The high volume of activities to be scheduled, combined with the need to apply sampling intensive methods, necessitates the implementation of a computationally efficient algorithm. We have adapted the *priority rule scheduling algorithm* (PRSA) from Vázquez, E.P.*et. al.* (2013) which accepts as input a set of activities $\mathcal{A}$ with durations and resource constraints, and a priority rule $\phi : \mathcal{A} \to \mathbb{R}$. Activities with no dependencies are added to the priority queue $\mathcal{Q}$ first. Then, activities are initialized when one or more dependencies are scheduled, and queued when all dependencies have been added to the schedule. Among the queued activities, we iteratively select the highest priority activity $a^* = argmax\{\phi(a) : a \in \mathcal{Q}\}$ to be scheduled. We then identify the earliest time a^* can be added to the schedule in accordance with global resource constraints, the release time of the activity and completion time of all dependencies (*). If a time cannot be found within the horizon, a^* and all subsequent dependencies are discarded. This continues until $\mathcal{Q} = \emptyset$.

A feature unique to our application is that many activities are defined by the same lesson, with the same resource requirements. This means that the earliest schedulable time for each lesson can be tracked and updated, avoiding the need to search the entire horizon in step (*) - a significant computational saving. The generation of a single schedule with 200000 activities for a fixed ϕ took on average 80ms on an i7 machine with 32GB RAM. A review of the runtime performance of extreme problem sizes up to 50M activities suggested that the algorithm had approximately linear time complexity in the number of activities.

4 Searching the priority rule space

For this PRSA algorithm, a priority rule ϕ must define the priority value for each activity, specifying the sequence in which activities are added to the schedule. Given that many activities relate to the same underlying lesson type, we define the priority rule in terms of these lessons.

4.1 Representations of priority rules space

We considered two representations of the priority rule space, firstly the direct representation, where $\alpha_i \in \mathbb{R}$ defines the priority value of lesson i and hence all related activities:

$$\phi = [\alpha_1, \ldots, \alpha_n]$$

Secondly we developed a feature-based representation of the activities, where the feature vector ζ_i defines the duration, resource requirements and other properties of lesson i. Then the actual priority rule was the weighted linear combinations β of the lesson features,where β_j defines the weighting of feature j:

$$\phi = \beta^T[\zeta_1, \dots, \zeta_n]$$

Let $h_{\phi,x} = PRSA(\phi, x)$ be the schedule resulting from applying $PRSA$ to problem instance x with priority rule ϕ. Now, the problem is identifying $\phi^* = argmax_\phi z(h_{\phi,x})$.

4.2 Searching with Random Key Genetic Algorithm and baseline methods

We develop a stochastic optimizer based on the *Random Key Genetic Algorithm* (RKGA) (Gonçalves, J.F. and Resende, M.G. 2011) as follows. The priority rule ϕ is taken to be the chromosome, with a random key representation $\phi_i \in [0, 1)$, where the ranks within ϕ defined the relative priorities of activities to schedule. The k-point crossover method (Umbarkar, A.J. and Sheth, P.D. 2015) with $\gamma = 0.05$ point-wise mutation probability in each chromosome element in each generation were used, with a population size of 50. The population was evolved using GA until the function valuation limit was exceeded to provide a fair comparison with other sampling methods.

For comparison, we also applied three baseline methods:

– **NOPRIORITY**: Fixes $\phi := [0, \dots, 0]$ to remove all prioritization of activities and default to tie-breaking rules
– **RANDOM**: Samples uniformly at random $\phi \in [0, 1]^n$ up to function evaluation limit
– **N/M**: Deterministic search with Nelder-Mead optimisation over $\phi \in \mathbb{R}^n$

The RANDOM, N/M and RKGA methods were each evaluated for both the direct representation, and the feature based representation ("**FEAT-***").

5 Preliminary experimentation

We generated 100 random problem instances which had comparable complexity to real world training facility problems in terms of numbers of activities, resources, durations and interdependencies between activities. Each search method was applied to each problem instance with a function evaluation limit of 10000. Early indications showed that RKGA consistently outperformed other methods and so a multi-start, longer-running version of this was implemented to identify the "**BESTKNOWN**" solution for each problem instance, in the absence of an exact solution or approximation bound. Then, the objective value of each search method for each instance, relative to the BESTKNOWN, was extracted as the performance measure, and the distribution of this measure over all problem instances was computed. The results are shown in Figure 1.

6 Discussion and future work

While all methods outperformed NOPRIORITY, we note that RKGA found the best solution in 92% of cases, obtaining on average 96.6% of best known objective value within the function evaluation limit, and showed significant potential for further gains with a greater limit. We observed that Nelder-Mead frequently converged on a local minimum significantly far from the best known, and may be unsuited given the high dimensionality.

Contrary to our expectations the feature based methods ("FEAT-*") performed consistently worse on average than their direct representation counterparts, possible because the

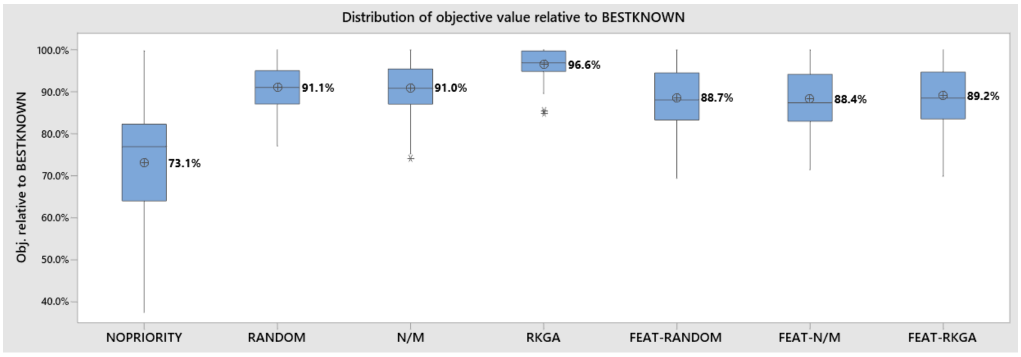


Fig. 1. Distribution of objective values relative to best known for each method (means labelled)

linear combination of raw features as $\phi = \beta^T[\zeta_1, \ldots, \zeta_n]$ weights features independently and does not encapsulate feature combinations. Identifying a more expressive feature set is an important task for future work.

In parallel, we are developing a MILP formulation of the same problem but operating over a much shorter horizon e.g. 1 week, with an approximate objective function. A key research question is to compare the MILP vs RKGA models on a common set of problem instances and evaluate over a long horizon. This will allow us to directly compare the value of long-horizon metaheuristic solutions v.s. a sequence of short-horizon exact solutions. A further task of interest is to evaluate the RKGA method over a standard, public set of RCMPSP instances for direct comparison with other methods in literature.

We conclude that the RCMPSP representation is highly applicable to the long-horizon training scheduling problem. The PRSA developed is extremely efficient for large-scale RCMPSP instances scaling up to millions of activities. This enables the use of genetic algorithms and other sampling intensive metaheuristics in the automated search for efficient priority rules, while optimising directly with respect to the long-horizon objective function.

Acknowledgements

We wish to thank Prof. Terry Caelli for his insightful comments and editorial feedback.

References

Blazewicz, J., Lenstra, J.K., Kan, A H.G. Rinnooy, 1983, “Scheduling subject to resource constraints : Classification and Complexity”, *Discrete Applied Mathematics*, Vol. 05, pp. 11-24

Gonçalves, J.F. and Resende, M.G., 2011, “Biased random-key genetic algorithms for combinatorial optimization”, *Journal of Heuristics*, Vol. 17(5), pp.487-525.

Kurtulus IB. and Davis, E.W., 1982, “Multi-project scheduling: Categorization of heuristic rules performance”,*Management Science*, Vol. 28(2), pp.161-172.

Umbarkar, A.J. and Sheth, P.D., 2015, “Crossover operators in genetic algorithms: A review”, *ICTACT journal on soft computing*, Vol. 06(1).

Vázquez, E.P., Calvo, M.P. and Ordóñez, P.M., 2015, “Learning process on priority rules to solve the RCMPSP”, *Journal of Intelligent Manufacturing*, Vol. 26(1), pp.123-138.

Villafáñez, F.A., Poza, D., López-Paredes, A. and Pajares, J., 2018, “A unified nomenclature for project scheduling problems (RCPSP and RCMPSP)”. *Dirección y Organización*, Vol. 64, pp. 56-60.

A mixed integer programming approach for scheduling aircraft arrivals at terminal airspace fixes and runway threshold

Sana Ikli[1], Catherine Mancel[1], Marcel Mongeau[1], Xavier Olive[2] and Emmanuel Racheslon[3]

[1] ENAC, Université de Toulouse, France
`sana.ikli, catherine.mancel, mongeau@recherche.enac.fr`
[2] ONERA DTIS, Université de Toulouse, France
`xavier.olive@onera.fr`
[3] ISAE-SUPAERO, Université de Toulouse, France
`emmanuel.rachelson@isae-supaero.fr`

Keywords: Aircraft landing scheduling, Delay minimization, Mixed integer linear programming.

1 Introduction and motivation

The continuous growth of air transportation demand exposes the Air Traffic Management (ATM) system to a serious risk of saturation, especially the terminal airspace – known as the Terminal Maneuvering Area (TMA) – of hub airports. A better utilization of current airport infrastructures can alleviate the saturation problem, for instance by reducing delays.

Several research works in the literature are interested in the optimization of runway sequences. However, most of such work focuses on the runway system. For instance, Beasley *et al.* (2000) propose a Mixed Integer Programming (MIP) approach for scheduling aircraft landings on multiple runways. Furini *et al.* (2015) consider the problem of scheduling simultaneously aircraft take-offs and landings on a single runway. Recently, Prakash *et al.* (2018) adapt the model of Beasley *et al.* (2000) to incorporate take-offs, and attempt at minimizing the schedule makespan. Readers may refer to the survey of Bennell *et al.* (2011) for a comprehensive review of the proposed solution approaches in the literature.

In this work, we propose a new Mixed Integer Linear Programming (MILP) approach for sequencing and scheduling aircraft arrivals at critical TMA points, called the Initial Approach Fixes (IAF), as well as on the runway threshold. What drives this work is that runway operations depend on other operations from the TMA, while the majority of works in the literature consider the runway as an independent resource. Another motivation comes from the observation of inbound traffic in the TMA, which reveals unbalanced traffic flows among the IAFs. Our objective is to re-distribute arriving aircraft among the existing TMA fixes and runways, so as to balance traffic volume. Preliminary results show that the average delay at runways can be reduced using our model.

2 Analogies and complexity

The problem of scheduling aircraft arrivals considering only runways is called the Aircraft Landing Problem (ALP) in the literature. It is similar to a number of classical combinatorial optimization problems, namely, the job-shop scheduling problem, the Traveling Salesman Problem (TSP), and the Vehicle Routing Problem (VRP).

The analogy between the ALP and a job-shop scheduling problem can be derived from Beasley *et al.* (2000) as follows. Runways correspond to machines, aircraft to be sequenced correspond to jobs, and the *safety separation* between two successive landings (see *Section* 3) correspond to the sum of the processing time of the previous job, and the sequence-dependent set up time. The typical objective function is to minimize the landing of the last aircraft in the sequence. It corresponds to minimizing the makespan of the schedule. In this case, if we consider a single runway, the problem is then equivalent to the TSP.

The analogy between the ALP and the VRP is pointed out in Briskorn and Stolletz (2014). In this context, runways represent vehicles to dispatch, aircraft represent customers to serve, and the *safety separation* between two successive landings correspond to the distance between two customers. The target landing times usually considered in the ALP, correspond to times at which customers prefer to be served. Upper and lower bounds on these target times correspond to the classical time-window restrictions. The objective function consists then in landing (serving) each aircraft (customer) within its time window, such that the cost associated to deviations from target times is minimized.

It can be deduced from these analogies that even the simplest versions of the ALP, involving only runways, are NP-hard problems. However, some versions of the ALP that impose particular restrictions can be solved in polynomial times. These include ALP with Constrained Position Shifting (CPS), as shown in Balakrishnan and Chandran (2006), and the ALP with aircraft classes, presented in Briskorn and Stolletz (2014).

3 Problem context and formulation

In this work, we consider the problem of sequencing and scheduling aircraft arrivals at two levels: first, over initial approach fixes, which are the points where the initial approach segment of an Instrument Flight Rule (IFR) approach begins (Figure 1), and then at the runway threshold.

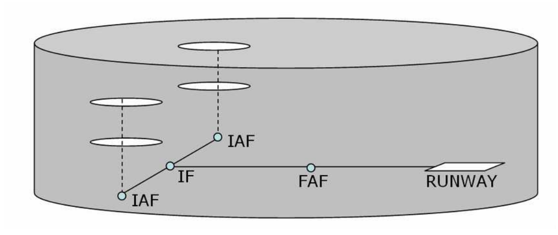


Fig. 1. A diagram of a typical IFR approach, with two IAFs and one runway (IF and FAF are intermediate way points on the approach trajectory). Source: Kelly and Painter (2006)

We propose a MILP formulation of this problem. The given data includes, for each aircraft: a target landing time, a latest acceptable landing time (based on fuel considerations), and the transition time between each IAF and the runways. Two types of safety constraints are considered:

- The pairwise separation of 3 Nautical Miles (NM) at each IAF.
- the Wake-Vortex (WV) separation at the runway threshold (Table 1), which corresponds to the minimal International Civil Aviation Organization (ICAO) separation requirements between two successive landings.

In our MILP formulation, two kinds of decision variables are proposed. First, binary variables are introduced for sequencing purposes as well as for runway and IAF assign-

Table 1. Minimum time separation between two successive landings in seconds, according to the three WV aircraft categories: Heavy (H), Medium (M), and Light (L). Source: Balakrishnan and Chandran (2006)

		Following aircraft		
		H	**M**	**L**
	H	96	157	196
Leading aircraft	**M**	60	69	131
	L	60	69	82

ments. Continuous optimization variables for assigning times at the IAF and at the runway threshold for each aircraft. The objective function is the total schedule delay to be minimized.

4 Preliminary results

The proposed model is implemented and solved via DoCplex, the Python API of CPLEX solver. The test problem instances are generated from real traffic in Paris-Orly airport. These instances feature three IAFs, and two runways (denoted R_1 and R_2 in Table 2).

We compare the three following solutions:

- FCFS: The basic technique usually used by air traffic controllers. It consists in assigning aircraft to the closest IAF according to its origin airport, then to the closest runway from the IAF. The sequence of aircraft is defined based on the First-Come First-Served (FCFS) order.
- RAS-MILP: The runway assignment and sequencing using our MILP model, without IAF assignment decisions.
- RIAS-MILP: The runway and IAF assignment and sequencing using our MILP model.

The results are reported in Table 2 in terms of average delay per aircraft (Avg Delay) of each schedule; it is computed as follows. The set $\mathcal{F}$ denotes the set of aircraft in the schedule, T_f and t_f are respectively the target and the scheduled time of aircraft $f \in \mathcal{F}$. The average delay of the schedule is then given by Equation (1).

$$\text{Avg Delay} = \frac{\sum\limits_{f \in \mathcal{F}} (t_f - T_f)}{|\mathcal{F}|} \tag{1}$$

The results of Table 2 show to what extent an optimization approach can be beneficial for realistic instances of the problem of sequencing and scheduling aircraft, even in the case RAS-MIP – in which IAF assignment is fixed according to the airport of origin – compared to the traditional solution used by controllers (FCFS). Moreover, our complete MILP (RIAS-MILP) yields average delays that are, as expected, further reduced (more degrees of freedom).

The gain remains moderate for these instances, but we are currently working on more congested problems for which our approach is likely to be more valuable.

5 Conclusion and perspectives

This work focuses on the problem of sequencing and scheduling aircraft arrivals at critical terminal airspace fixes and at the runway threshold. We propose a Mixed Integer

Table 2. Average delay comparison between three solution techniques (in seconds), for 10 problem instances from 12:00 am to 10:00 pm

		FCFS			RAS-MILP			RIAS-MILP		
Time	Nb Flights	R_1	R_2	Avg Delay	R_1	R_2	Avg Delay	R_1	R_2	Avg Delay
12:00-13:00	28	18	10	53	12	16	23	15	13	17
13:00-14:00	20	9	11	57	7	13	14	12	8	6
14:00-15:00	15	6	9	60	6	9	20	8	7	13
15:00-16:00	17	11	6	58	8	9	18	10	7	8
16:00-17:00	22	11	11	100	9	13	52	10	12	25
17:00-18:00	21	9	12	64	8	13	17	12	9	0
18:00-19:00	19	9	10	58	8	11	15	13	6	10
19:00-20:00	22	13	9	47	10	12	17	13	9	6
20:00-21:00	18	8	10	62	8	10	24	13	5	4
21:00-22:00	26	11	15	64	14	12	24	15	11	11
Min	15	6	6	53	6	9	14	8	5	0
Max	28	18	15	100	14	16	52	15	13	25

Linear Programming approach that takes into account safety requirements, and maximum acceptable delay based on fuel considerations. The preliminary results show that the average delay can be reduced using our model, compared to the traditional technique used by controllers. In future studies, we plan to validate our model on more congested data from other airports, such as the Paris Charles-de-Gaulle Airport. Then, we aim at discussing further with air traffic controllers so as to improve our model, to render it more realistic, in order to evaluate the viability of our approach in a real-time application.

References

Balakrishnan, H., Chandran, B., 2006, " Scheduling aircraft landings under constrained position shifting", *AIAA guidance, navigation, and control conference and exhibit*, pp. 6320.

Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M., Abramson, D., 2000, "Scheduling aircraft landings–The static case", *Transportation science*, Vol. 34, pp. 180-197.

Bennell, J. A., Mesgarpour, M., Potts, C. N., 2011, "Airport runway scheduling", *4OR*, Vol. 9,, pp. 115.

Briskorn, D., Stolletz, R, 2011, "Aircraft landing problems with aircraft classes", *Journal of Scheduling*, Vol. 17, pp. 31-45.

Furini, F., Kidd, M. P., Persiani, C. A., Toth, P., 2015, "Improved rolling horizon approaches to the aircraft sequencing problem", *Journal of Scheduling*, Vol. 18, pp. 435-447.

Kelly, W. E., Painter, J. H., 2006, "Flight segment identification as a basis for pilot advisory systems", *Journal of aircraft*, Vol. 43, pp. 1628-1635.

Prakash, R., Piplani, R., Desai, J., 2015, "An optimal data-splitting algorithm for aircraft scheduling on a single runway to maximize throughput", *Transportation Research Part C: Emerging Technologies*, Vol. 95, pp. 570-581.

Optimization of order for containers placement schedule in rail terminal operations

Nadiia Kalaida[1], **Rémy Dupas**[2] **and Igor Grebennik**[1]

[1]Dept. of System Engineering Kharkiv National University of Radio Electronics, Kharkiv, Ukraine
e-mail: nadiia.kalaida@nure.ua, igorgrebennik@gmail.com

[2] Laboratory IMS University of Bordeaux, Bordeaux, France
e-mail: remy.dupas@gmail.com

Keywords: rail terminal operations, container, mathematical model, combinatorial optimization.

1. Introduction

During the transportation of goods by rail, the processing of trains and performing rail terminal operations at the transshipment yard (TSY) in the railway station raise problems of optimization (Boysen et al.,2012; Boysen et al., 2011; Dotoli et al.,2013; Cicheński et al., 2017).

The work of the railway transshipment station includes the following processes:

1. Arrival of the train at the railway transshipment station.
2. Train service (unloading / loading of containers).
3. Departure of the train from the station.

Serving a train involves moving containers from one train to another, reloading the container from the train to storage area and from storage area to the train.

The components of a modern railway transshipment yard (Figure 1) are proposed in (Boysen et al., 2012):

1. A platform with parallel railway tracks on which trains are located;
2. Container storage area, which is located parallel to the tracks;
3. Gantry cranes are used to reload containers.

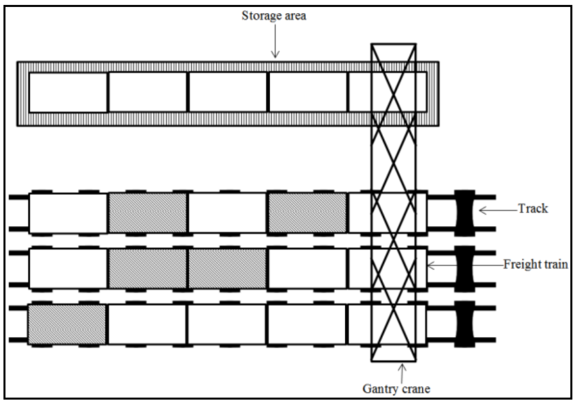


Figure 1 - Scheme of a railway transshipment yard

The operation of the transshipment yard is as follows: a train with containers (source train) arrives at the transshipment yard, containers from an incoming train are loaded onto the target train (if it is present at the station) or in the storage area. Containers from other trains and containers from the storage area can also be loaded onto the source train. At the end of processing, the train leaves the transshipment yard.

(Boysen et al., 2011 and 2012) describe the general problem of planning the work of a transshipment yard, in which the following levels are distinguished:

1. Service slot planning (transshipment yard scheduling problem, TYSP);
2. Assignment of trains to railway tracks in one slot;
3. Determining the position of the containers on trains;
4. Determining of zones of action of gantry cranes;
5. Determining the order of the moves of the containers done by the cranes.

(Boysen et al., 2012) solve the problem at level 1: the authors propose a mathematical model for creating a schedule for visiting trains at the transshipment yard. This approach aims at distributing the trains on service slots.

(Grebennik et al., 2017) extended the study of the level 1 problem and proceed to level 2. The authors propose to form time intervals for servicing trains (service slots) and to assign each train of such a service slot to a specific railway track of the transfer station. The proposed mathematical model is based on combinatorial configurations, see for example (Sachkov, 1996). (Grebennik et al., 2019) offer a solution of the level 3 problem which is based on the results of studies given in (Grebennik et al., 2017). The optimal position of the containers to be moved on the target train is determined. The combinatorial optimization model is proposed and analyzed.

In this paper, we consider the following hypotheses:

1. The cost of moving containers from the source train to the target train depends on the distance between the trains;

2. If the source train and the target train are in the same service slot, they must be located on the nearest tracks;

3. If the trains are assigned to different service slots, it is necessary to minimize the move of the gantry crane;

4. The cost of moving containers is estimated by the distance that the gantry crane travels.

This work is a continuation of the studies of (Boysen et al.,2012; Grebennik et al.,2017; Grebennik et al.,2019). Its purpose is to determine the optimal order of moves of containers between the trains and the storage area in the formed service slots.

2. Problem definition

The problem analyzed in this paper is based on two problems: the transshipment yard scheduling problem (TYSP) and the containers placement in rail terminal operations problem (CPRTOP). A set of trains with a certain number of empty freight platforms and containers and their known location on the train is considered. All the trains arriving at a transfer station have to be allocated to service slots. The number of trains that can be serviced simultaneously (in one slot) should not exceed the number of parallel tracks at the transshipment yard. The optimal service slot is constructed as the solution of the combinatorial optimization problem. Trains that are served in a single service slot need to reload containers from train to train, from train to storage area, or from storage area to train. Given the coordinates of the location of the containers for moving and the coordinates of the free platforms in the target trains and storage area, the problem of the optimal containers placement is solved as combinatorial optimization problem. Based on the results of solving the problem at levels 1-3 (Grebennik et. al., 2017; Grebennik et. al., 2019) we assume that the following data are available and known: positions of the trains selected in one slot, the assignment of each train on a railway track and the assignment of each container to a specified platform in this slot. Therefore the problem of the relocation order of containers at a transshipment yard (ROCTY) can be stated as follows: let a service slot with specified locations of trains on the tracks is defined, the coordinates of containers and free platforms on trains and in the storage area are determined; for each container has to be moved, the 2D coordinates of the platform on which it has to be placed are known. Let us determine the order of relocation of containers with a gantry crane, minimizing the cost of servicing the entire service slot.

3. Mathematical model of the problem of relocation order of containers at a transshipment yard

Let $V = \{v_1, v_2, \dots, v_n\}$, where $v_i = (v_x^i, v_y^i)$, be set of coordinates of the geometric centers of the relocating containers on source trains and in the storage area, free platforms on target trains and in the storage area, n is the number of the containers and the free platforms. We form set $D = \{d_1, d_2, \dots, d_m\}$, where $d_i = (d_i^f, d_i^t)$ is ordered pair of elements from set V, for which the container relocation is intended from a source point of coordinates d_i^f to the target point of coordinates d_i^t, m is number of containers have to be moved.

Set a mixed graph $G = (V, E)$, where $V = \{v_1, v_2, \dots, v_n\}$ is set of vertices, $E = D \cup C$ is union of the set of arcs D (which correspond to the relocation of containers from place to place) and undirected edges C (which correspond to move of the gantry crane without container). Set of edges

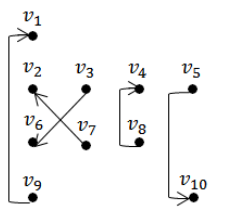
C is constructed in such way that graph $G_1 = (V, C)$ is a complete undirected graph. For graph G, we calculate the edge length matrix $\|c_{ij}\|$ based on the distance between the vertices of the graph which is calculated according to the Manhattan metric: $c_{ij} = |v_x^i - v_x^j| + |v_y^i - v_y^j|$, $i, j = \{1,2, \dots, n\}$.

We firstly choose the initial position of the crane at one of the vertices of the graph G. Then, using the description of the crane problem given in (I. I. Melamed et. al., 1989), we construct the traveling salesman route in graph G, with the following condition if arcs come out of a vertex and edges are connected with it, then you can move along the edge only if you cannot move along the arc. In accordance with the approach in (I. I. Melamed et. al., 1989), we transform this problem into the traveling salesman problem, see for example (Applegate, 2006). For this purpose, we put the lengths of all arcs $d_i \in D, i = 1, \dots m$, of the graph G equal to zero and then combine each pair of vertices connected by an arc into one vertex. In this case, the distance between the combined vertices is determined as follows. If vertex $\overline{v}_i$ is obtained by the union of vertices v_{i_1} and v_{i_2}, vertex $\overline{v}_j$ is obtained by the union of vertices v_{j_1} and v_{j_2}, then $\overline{v}_{ij} = \min(c_{i_1 j_1}, c_{i_1 j_2}, c_{i_2 j_1}, c_{i_2 j_2})$. Lastly, we solve the traveling salesman problem with the matrix $\overline{c}_{ij}$ using one of the known methods, for example, using one of the modern solvers Concorde TSP solver (available at http://www.math.uwaterloo.ca/tsp/concorde.html).

To obtain a solution to the crane problem, and then to solve the ROCTY problem, we return to the original graph G and add all arcs from the set D to the found traveling salesman route. The move route that is found as a result of solving the crane problem on graph G is the solution of the ROCTY problem.

4. An example of solving the ROCTY problem

Let $V = \{v_1, v_2, \dots, v_{10}\}$ be set of vertices, where $v_1 = (1,0)$, $v_2 = (1,1)$, $v_3 = (2,1)$, $v_4 = (3,1)$, $v_5 = (4,1)$, $v_6 = (1,2)$, $v_7 = (2,2)$, $v_8 = (3,2)$, $v_9 = (1,3)$, $v_{10} = (4,3)$; $D = \{d_1, d_2, d_3, d_4, d_5\}$ be set of arcs, where $d_1 = (v_9, v_1)$, $d_2 = (v_3, v_6)$, $d_3 = (v_7, v_2)$, $d_4 = (v_8, v_4)$, $d_5 = (v_5, v_{10})$; Figure 2.a describes the set of vertices V and set of arcs D, which connect pairs of vertices from set V; Figure 2.b shows the calculated distance matrix $\|c_{ij}\|$.



a)

b)

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
v_1	0	1	2	3	4	2	3	4	100	6
v_2	1	0	2	3	3	1	100	3	2	5
v_3	2	2	0	1	2	2	1	2	3	4
v_4	3	3	1	0	1	3	2	100	4	3
v_5	4	3	2	1	0	4	3	2	5	2
v_6	2	1	100	3	4	0	1	2	1	4
v_7	3	2	1	2	3	1	0	1	2	3
v_8	4	3	2	1	2	2	1	0	3	2
v_9	3	2	3	4	5	1	2	3	0	3
v_{10}	6	5	4	3	100	4	3	2	3	0

Figure 2 – a) Vertices and arcs of the graph, b) Distance matrix $\|c_{ij}\|$

Combine the vertices connected by arcs: $\overline{v}_1 = v_9 \leftrightarrow v_1$; $\overline{v}_2 = v_7 \leftrightarrow v_2$; $\overline{v}_3 = v_3 \leftrightarrow v_6$; $\overline{v}_4 = v_8 \leftrightarrow v_4$; $\overline{v}_5 = v_5 \leftrightarrow v_{10}$. Distance matrix $\|\overline{c}_{ij}\|$ for complete graph with vertices $\overline{V} = \{\overline{v}_1, \overline{v}_2, \overline{v}_3, \overline{v}_4, \overline{v}_5\}$ is represented on Figure 3. To solve the traveling salesman problem with a matrix $\|\overline{c}_{ij}\|$ we use a public web service, as a result of which a closed route of moves between the vertices is obtained $\overline{V} = \{\overline{v}_1, \dots, \overline{v}_5\}$: $\overline{v}_4 \to \overline{v}_5 \to \overline{v}_3 \to \overline{v}_1 \to \overline{v}_2 \to \overline{v}_4$.

	$\overline{v}_1$	$\overline{v}_2$	$\overline{v}_3$	$\overline{v}_4$	$\overline{v}_5$
$\overline{v}_1$	0	1	1	3	3
$\overline{v}_2$	1	0	1	1	3
$\overline{v}_3$	1	1	0	1	2
$\overline{v}_4$	3	1	1	0	1
$\overline{v}_5$	3	3	2	1	0

Figure 3 – Distance matrix $\|\overline{c}_{ij}\|$

To construct the gantry crane path along the vertices of the original graph G, we add all arcs of the set D to the found closed route. The result of such a transformation will be the route of the gantry crane over the set of vertices $V = \{v_1, v_2, \dots, v_{10}\}$,; the final solution for the input dataset is: $v_8 \to v_4 \to v_5 \to v_{10} \to v_3 \to v_6 \to v_9 \to v_1 \to v_7 \to v_2 \to v_8$.

5. Conclusions

The paper considers the problems of the functioning of the railway transshipment yard. The known problems of the transshipment yard scheduling problem (TYSP) and the containers placement in rail terminal operations problem (CPRTOP) are presented. Based on their solutions, problem of the relocation order of containers at a transshipment yard (ROCTY) is formulated. An optimization model of the ROCTY problem is constructed, which can be transformed into the traveling salesman problem. Based on the solution of the traveling salesman problem, the solution of the ROCTY problem is constructed. A computational experiment is carried out, its results are discussed.

References

Applegate D., Robert E. Bixby, Vasek Chvátal & William J. Cook, 2006, The Traveling Salesman Problem: A Computational Study, Princeton University Press.

Boysen, Jaehn, and Pesch, 2011, “Scheduling Freight Trains in Rail-Rail Transshipment Yards,” Transportation Science, Vol. 45(2), pp. 199–211.

Boysen N., F. Jaehn, and E. Pesch, 2012, "New bounds and algorithms for the transshipment yard scheduling problem", J. Sched., Vol.15, pp. 499–511.

Cicheński M., and F. Jaehn, and G. Pawlak, and E. Pesch, and G. Singh, and J. Blazewicz, 2017, "An integrated model for the transshipment yard scheduling problem", Journal of Scheduling, Vol. 20, pp.57-65.

Dotoli M., N. Epicoco, M. Falagariob, D. Palmaa, B. Turchianoa, 2013, “A Train Load Planning Optimization Model for Intermodal Freight Transport Terminals: A Case Study”, Proceedings - 2013 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013. pp. 3597-3602.

Grebennik I., R. Dupas, O. Lytvynenko and I. Urniaieva, 2017, “Scheduling Freight Trains in Rail–rail Transshipment Yards with Train Arrangements,” International Journal of Intelligent Systems and Applications (IJISA), Vol. 9(10), pp.12–19.

Grebennik I., R. Dupas, I. Urniaieva, N. Kalaida and V. Ivanov, 2019, "Mathematical Model of Containers Placement in Rail Terminal Operations Problem," 2019 9th International Conference on Advanced Computer Information Technologies (ACIT), pp. 129-132.

Melamed I. I., S. I. Sergeev, I. Kh. Sigal, 1989, “The traveling salesman problem. Issues in theory”, Autom. Remote Control, Vol. 50(9), pp. 1147–1173.

Sachkov V., 1996, Combinatorial Methods in Discrete Mathematics, Cambridge University Press, Cambridge.

A Generation Scheme for the Resource-Constrained Project Scheduling Problem with Partially Renewable Resources and Time Windows

Mareike Karnebogen and Jürgen Zimmermann

Clausthal University of Technology, Germany
`mareike.karnebogen@tu-clausthal.de, juergen.zimmermann@tu-clausthal.de`

Keywords: Project scheduling, Partially renewable resources, RCPSP/max,π

1 Introduction

Partially renewable resources are a generalized form of renewable and non-renewable resources. For each partially renewable resource a capacity is given, which only applies to predetermined time periods. This type of resources was first mentioned by Böttcher *et al.* (1996) in the context of projects restricted by precedence constraints (RCPSP/π). In this paper the resource-constrained project scheduling problem with partially renewable resources and time windows (RCPSP/max,π) is considered, which for example allows to take more flexible working arrangements in case of deployment planning into account. The RCPSP/max,π is a generalization of the RCPSP/π and thus hard to solve (NP-hard). Therefore, we present a generation scheme for the RCPSP/max,π to construct feasible solutions within a short period of time. Section 2 contains a general description of the problem and a mixed-integer linear formulation (MILP). In Section 3 the developed generation scheme is presented. Finally, Section 4 includes a preliminary performance analysis, in which the results of our generation scheme are compared to results obtained by *IBM CPLEX* applied on the MIP model of Watermeyer *et al.* (2018).

2 Problem description

The activities and temporal constraints of the resource-constrained project scheduling problem with time windows and partially renewable resources (RCPSP/max,π) can be represented graphically as an activity-on-node network. The nodes correspond to the activities of the project $V = \{0, 1, \ldots, \text{n+1}\}$ consisting of the fictitious project start 0, n real activities, and the fictitious project completion $n + 1$. Each activity $i \in V$ has a deterministic processing time $p_i \in \mathbb{Z}_{\geq 0}$ during which the activity can not be interrupted. General temporal constraints between two activities $i, j \in V$ are represented by arcs $\langle i, j\rangle$ between the corresponding nodes of the network. The arc weight $\delta_{ij} \in \mathbb{Z}$ of an arc $\langle i, j\rangle \in E$ corresponds to a minimal time lag between the start times of activity i and activity j which has to be satisfied. The maximal project duration given by $\bar{d}$ can be represented by an arc from $n + 1$ to 0 weighted with $-\bar{d}$. As it is common practice in literature, let d_{ij} be the longest distance from node $i \in V$ to node $j \in V$ in the project network which can be calculated by the Floyd-Warshall tripel algorithm. Then $ES_i = d_{0i}$ and $LS_i = -d_{i0}$ are the earliest and latest start time of activity $i \in V$, respectively, and $\mathcal{T}_i = \{ES_i, ES_i + 1, ..., LS_i\}$ implies all time-feasible integer start time points of activity i.

When executing the activities a set of partially renewable resources $\mathcal{R} = \{1, \ldots, m\}$ have to be observed. Each activity $i \in V$ has a resource demand $r_{ik} \in \mathbb{Z}_{\geq 0}$ for each period it is executed. In addition each partially renewable resource $k \in \mathcal{R}$ has a resource capacity R_k, which only applies to a subset of not necessarily consecutive time periods of the given planning horizon $\Pi_k \subseteq \{1, 2, \ldots, \bar{d}\}$. For all time periods not contained in Π_k it is assumed that the capacity is not restricted. Obviously, the relevant composite resource consumption of activity i of resource k depends on the number of time periods activity i is in execution during the capacitated time periods of resource k and can be calculated by $r^c_{ik}(S_i) = |\{S_i + 1, S_i + 2, \ldots, S_i + p_i\} \cap \Pi_k| \cdot r_{ik}$ (Watermeyer *et al.* 2018).

The objective of our problem is to find a time- and resource-feasible schedule $S = (S_0, S_1, \ldots, S_{n+1})$ which minimizes the project duration S_{n+1}. A schedule is called time-feasible, if $S_j - S_i \geq \delta_{ij}\ \forall \langle i, j\rangle \in E$, what means that all temporal constraints are observed. To obtain a resource feasible schedule the accumulated resource demand occurring across all capacitated periods must not exceed R_k for any partially renewable resource $k \in \mathcal{R}$. Formally, the RCPSP/max,π described above can be formulated as follows:

$$\begin{array}{lll}
\text{Minimize} & f(S) = S_{n+1} & \\
\text{subject to} & S_j - S_i \geq \delta_{ij} & (\langle i, j\rangle \in E) \\
 & S_0 = 0 & \\
 & \sum_{i \in V} r^c_{ik}(S_i) \leq R_k & (k \in \mathcal{R}) \\
 & S_i \in \mathbb{Z}_{\geq 0} & (i \in V)
\end{array}$$

3 Generation scheme

Algorithm 1 shows our generation scheme which is based on the generation scheme for the RCPSP/max developed by Franck *et al.* (2001). Let C be the set of scheduled activities. In the initialization process project start 0 is fixed at $t = 0$ and appended to C. Also counter u is set to zero. In the main step for each partially renewable resource $k \in \mathcal{R}$ and each activity not scheduled so far the minimal composite demand r^{min}_{ik} and the maximal composite demand r^{max}_{ik} is calculated. As well as the remaining capacity RC_k

Algorithm 1 Generation Scheme

```
1: $\mathcal{C} := \{0\}$, $S_0 := 0$, $u := 0$
2: $r_{ik}(t)$ for all $i \in V$ and $k \in \mathcal{R}$ and $t \in \mathcal{T}_i$
3: while $\mathcal{C} \neq V$ do
4:     $r^{min}_{ik} := \min_{t \in \mathcal{T}_i} r_{ik}(t)$ for all $i \in V \setminus \mathcal{C}$ and $k \in \mathcal{R}$
5:     $r^{max}_{ik} := \max_{t \in \mathcal{T}_i} r_{ik}(t)$ for all $i \in V \setminus \mathcal{C}$ and $k \in \mathcal{R}$
6:     $RC_k := R_k - \sum_{i \in V \setminus \mathcal{C}} r^{min}_{ik} - \sum_{i \in C} r_{ik}(S_i)$ for all $k \in \mathcal{R}$
7:     for all $k \in \mathcal{R}$ do
8:         if $RC_k > \sum_{i \in V \setminus \mathcal{C}} r^{max}_{ik}$ then $\mathcal{R} = \mathcal{R} \setminus \{k\}$
9:     $\mathcal{E} := \{i \in V \setminus \mathcal{C} \mid Pred^{\prec_D}(i) \subseteq \mathcal{C}\}$
10:    priority based choice of an activity $j^* \in \mathcal{E}$ to be scheduled next
11:    $Z_{j^*} := \{t \in \mathcal{T}_{j^*} \setminus Tabu_{j^*} | r_{j^*kt} - r^{min}_{j^*k} \leq RC_k$ for all $k \in \mathcal{R}$ and
           $\min_{k \in \mathcal{R}}\{r_{j^*kt} - r_{j^*k\tau}\} < 0$ for all $\tau \in \mathcal{T}_{j^*} | \tau < t\}$
12:    if $Z_{j^*} = \emptyset$ then $u := u + 1$ and Unschedule
13:    else
14:        priority based choice of a point in time $t^* \in Z_{j^*}$ as start time of $j^*$
15:        $S_{j^*} := t^*$, $\mathcal{C} := \mathcal{C} \cup \{j^*\}$
16:        for all $h \in V \setminus \mathcal{C}$ do (* update $ES_h$ and $LS_h$ *)
17:            $ES_h := \max(ES_h, S_{j^*} + d_{j^*h})$
18:            $LS_h := \min(LS_h, S_{j^*} - d_{hj^*})$
19: return $S$
```

which results from R_k minus the consumption of all scheduled activities $i \in C$ as well as the minimal necessary resource consumption r_{ik}^{min} of all not yet scheduled activities $i \in V \setminus C$. If RC_k outruns the maximal potential resource consumption r_{ik}^{max} of all activities $i \in V \setminus C$, the resource has no longer to be taken in consideration. Afterwards, the eligible set $\mathcal{E}$ containing all activities $i \in V \setminus C$ whose immediate predecessors regarding the distance order $\prec_D$ are scheduled is established (Neumann *et al.* 2003). An activity $j^* \in \mathcal{E}$ is selected based on a certain priority rule and the related set Z_{j^*} of resource- and time-feasible start times which are not dominated by an earlier feasible start time is determined. If $Z_{j^*} = \emptyset$ an unscheduling step is performed and counter u is increased by one. Otherwise a point in time $t^* \in Z_{j^*}$ is chosen based on a priority rule and assigned as S_{j^*} while j^* is added to C. Finally, for all unscheduled activities $i \in V \setminus C$ ES_i and LS_i are updated. This procedure is repeated until all activities $i \in V$ are scheduled time- and resource-feasible.

Algorithm 2 Unschedule

```
 1: if $u \geq \hat{u}$ then terminate
 2: if $ES_{j^*} \neq d_{0j^*}$ then $\mathcal{U} := \{i \in \mathcal{C} \mid ES_{j^*} = S_i + d_{ij^*}\}$
 3: if $LS_{j^*} \neq -d_{j^*0}$ then $\mathcal{U} := \mathcal{U} \cup \{i \in \mathcal{C} \mid LS_{j^*} = S_i - d_{j^*i}\}$
 4: if $\mathcal{U} := \emptyset$ then $\mathcal{U} := \{i \in \mathcal{C} \mid \min\{r_{ikS_i}, r_{j^*k}\} > 0$ for at least one $k \in \mathcal{R}\}$
 5: for all $i \in \mathcal{U}$ do
 6:     $\mathcal{C} := \mathcal{C} \setminus \{i\}$
 7:     $Tabu_i = Tabu_i \cup \{S_i\}$
 8: $Tabu_{j^*} := \emptyset$
 9: for all $i \in \mathcal{C}$ with $S_i > \min_{h \in \mathcal{U}} S_h$ do
10:     $\mathcal{C} := \mathcal{C} \setminus \{i\}$
11: for all $h \in V \setminus \mathcal{C}$ do
12:     $ES_h := d_{0h}$
13:     $LS_h := -d_{h0}$
14:     for all $i \in \mathcal{C}$ do
15:         $ES_h := \max(ES_h, S_i + d_{ih})$
16:         $LS_h := \min(LS_h, S_i - d_{hi})$
```

Algorithm 2 shows the unscheduling step which is performed if no time- and resource-feasible starting point of activity j^* exists. In case u is higher than a prescribed maximal number of unscheduling steps $\hat{u}$ the algorithm terminates. Otherwise a set U of activities $i \in C$ which have to be unscheduled and rescheduled in order to obtain a feasible schedule is determined. For this purpose, we first examine if one or more activities $i \in C$ restrict the scheduling timeframe of the chosen activity j^* i.e. increases ES_{j^*} or decreases LS_{j^*}. If $U = \emptyset$, we determine all those activities $i \in C$ using some resources $k \in \mathcal{R}$ activity j^* also requires for execution. Afterwards, all activities $i \in U$ are removed from C as well as all activities $i \in C$ with $S_i > \min_{h \in \mathcal{U}} S_h$ because some of them could possibly be executed earlier. Moreover, for all activities $i \in U$ the current start point S_i is forbidden by storing in the tabu-list $Tabu_i$ whereas $Tabu_{j^*}$ is cleared. Points in time $t \in Tabu_i$ can not be choosen as start time of activity i in the scheduling phase of the generation scheme (compare Algorithm 1 line 11). Finally, for all activities $i \in V \setminus C$ the values for ES_i and LS_i are recalculated.

4 Performance analysis

In order to evaluate the performance of our generation scheme we conduct a computational study performed on an Intel Core i7-7700K CPU with 4.2 GHz and 64 GB RAM under Windows 10. The generation scheme was coded in FICO® Xpress Optimization. The instance set we used was established by Watermeyer *et al.* (2018) including 729 instances with 10, 20, and 50 activities and is based on the well-known UBO instances of Schwindt (1998) extended by 30 partially renewable resources with varying specifications.

Within the computational study for the choice of the activity scheduled next the priority rules LSTd (smallest "Latest Start Time dynamic" first) and TFd (smallest "Total Float dynamic" first), whereas for the choice of the scheduling point in time the objectives T_{min} (earliest "Start Time"), RD (minimal total "Resource Demand") and RL ("Resource Leveling") were tested. Starting with $\bar{d}$ as the RCPSP/max upper bound $\sum_{\langle i,j\rangle \in E} |\delta_{ij}|$, we perform a preprocessing to specify $\bar{d}$ including two deterministic runs with the priority rules LSTd-T_{min} and TFd-T_{min}. In the main step for each of the six combinations of the priority rules 100 runs were conducted per instance, whereby the choice of j^* and t^* is taken randomly based on selection probabilities. If a feasible solution with $S_{n+1} < \bar{d}$ is found, $\bar{d}$ is set to $S_{n+1} - 1$.

Table 1 shows preliminary results for all combinations of the established priority rules. Displayed are the percentage of instances (%feas) our generation scheme was able to find a feasible solution, the average percentage gap (%Gap) with regard to the best solution of the MIP found in at most 3.600 seconds and the average computing time (∅CPU) in seconds required per run.

Table 1. Preliminary results of the computational study

		$UBO10^\pi$			$UBO20^\pi$			$UBO50^\pi$		
		T_{min}	RD	RL	T_{min}	RD	RL	T_{min}	RD	RL
	%feas	99.04	99.59	99.45	96.16	97.39	97.94	95.58	96.25	96.28
LSTd	%Gap	1.36	1.28	1.40	5.34	5.28	4.92	18.79	19.72	20.86
	∅CPU	0.63	0.61	0.61	2.42	2.09	2.05	35.71	32.59	33.35
	%feas	98.90	99.45	99.60	96.85	97.40	97.94	95.89	96.38	96.50
TFd	%Gap	1.40	1.47	1.36	5.89	5.70	5.73	18.81	19.78	21.07
	∅CPU	0.62	0.61	0.60	2.31	1.96	1.94	34.28	31.22	31.37

The results show that our generation scheme is able to generate feasible solutions for nearly all tested instances in particular by using the resource-based priority rules RD and RL. Note, that for some instances of $UBO50^\pi$ the generation scheme is able to find better solutions than the MIP in one hour. For the tested instances the quality of the solutions generated with the priority rules LSTd and TFd is very similar. For smaller instances the resource-based rules RD and RL mostly outperform the time-based rule T_{min}, whereas for larger instances it turns into its opposite. Besides higher gaps it can be observed that the computation time increases by a growing number of activities. In a next step the generation scheme should be coded in C++ and further priority rules should be examined.

References

Álvarez-Valdés R., E. Crespo, J.M. Tamarit and F. Villa, 2008, "GRASP and path relinking for project scheduling under partially renewable resources", *European Journal of Operational Research*, Vol. 189, pp. 1153-1170.

Böttcher J., A. Drexl, R. Kolisch, F. Salewski, 1996, "Project scheduling under partially renewable resource constraints", Technical Report, *Manuskripte aus den Instituten für Betriebswirt-schaftslehre 398*, University of Kiel.

Franck B., K. Neumann and C. Schwindt, 2015, "Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling". *OR Spektrum*, Vol. 23, pp. 297-324.

Neumann K., C. Schwindt, J. Zimmermann, 2003, "Project scheduling with Time Windows and Scarce Resources", ed.2, Springer, Berlin.

Schirmer A., 1999, "Project scheduling with scarce resources: models, methods and applications", Dr. Kovač, Hamburg.

Schwindt C., 1998, "Generation of resource-constrained project scheduling problems subject to temporal constraints", *Technical Report WIOR-543*, University of Karlsruhe.

Watermeyer K. and J. Zimmermann, 2018, "A Branch-and-Bound Procedure for the Resource-Constrained Project Scheduling Problem with Partially Renewable Resources and Time Windows", *Proceedings of the 16th International Conference on Project Management and Scheduling*, Rom, pp. 259-262.

On a Polynomial Solvability of the Routing Open Shop with a Variable Depot

Antonina P. Khramova[1], Ilya Chernykh[1,2,3]

[1] Sobolev Intitute of Mathematics, Russia
akhramova@math.nsc.ru idchern@math.nsc.ru
[2] Novosibirsk State University, Russia
[3] Novosibirsk State Technical University, Russia

Keywords: Open shop, routing open shop, unrelated travel times, variable depot, computational complexity, exact algorithm, approximation algorithm.

1 Preliminaries

The open shop problem to minimize finish time (Gonzalez and Sahni 1976) is one of the classical multistage scheduling problems and can be described as follows. Let $\mathcal{M} = \{M_1, \ldots, M_m\}$ be a set of m machines and $\mathcal{J} = \{J_1, \ldots, J_n\}$ be a set of n jobs. Each job J_j consists of m operations $(O_{j1}, \ldots, O_{jm})$. The operation O_{ji} takes p_{ji} time units and has to be processed on machine M_i, and no two operations of the same job can be processed at the same time, as well as no machine can process two jobs simultaneously. However, unlike flow shop model, the operations of a job can be processed in any order. We follow standard notation (Lawler *et. al.* 1993) to denote this problem as $Om||C_{\max}$.

It is known (Gonzalez and Sahni 1976) that if the number of the machines is at least three, $Om||C_{\max}$ is NP-hard. However, $O2||C_{\max}$ is polynomially solvable. Several algorithms are known to solve this problem in linear time. The first one was introduced by Gonzalez and Sahni (1976). Other algorithms for this problem were proposed by Pinedo and Schrage (1982), de Werra (1989), and Soper (2015).

The routing open shop problem is a certain generalization of the open shop problem and can be described as follows. Each job is assigned to a node of a transportation network given by an undirected edge-weighted graph G. The weight of an edge represents the time required by any machine to travel between the respective nodes. To process a job, a machine has to move to the node where the job is located. So, machines have to travel over the transportation network in order to process the jobs. It is assumed that any number of machines can travel over the same edge at the same time. All machines start at the same node, called the *depot*, and must return to the depot after completing all jobs. The goal is to minimize the makespan (*i.e.* the completion time of the last activity of a machine), denoted by $R_{\max}$. The notation $ROm||R_{\max}$ denotes the problem in case of m machines. We also use notation $ROm|G = X|R_{\max}$ in order to specify the structure X of the transportation network.

The routing open shop problem is a generalization of both the open shop problem (consider every edge of the graph to be of zero weight) and the metric traveling salesman problem (consider every operation to be of zero processing time), so it is obviously NP-hard in general case. The routing open shop problem was introduced and proved to be NP-hard even in the simplest case with two machines and $G = K_2$ in (Averbakh *et. al.* 2006). We further extend the problem statement with the following options introduced in (Chernykh 2016).

1. The depot in $ROm||R_{\max}$ may be either *fixed*, *i.e.* defined in the problem instance, or *variable*, *i.e.* it has to be chosen while composing a schedule. We write $ROm|variable-depot|R_{\max}$ to indicate the latter case.

2. The travel times between the nodes may differ for each machine. In particular, they can be *identical*, *uniform*, *i.e.* for any two machines M_{i_1} and M_{i_2} there is some $k > 0$ so that any edge for M_{i_1} is k times longer then it is for M_{i_2}, or *unrelated*. In the three-field notation, we write Qtt or Rtt in the last two cases respectively.

We write $easy - TSP$ in three-field notation if the structure of the transportation network G allows solving the underlying TSP in polynomial time. While the problem $RO2|variable - depot|R_{\max}$ is still NP-hard, being a generalization of the metric TSP, the algorithmic complexity of the problem $RO2|easy - TSP, variable - depot|R_{\max}$ was an open question. The special case $RO2|G = tree, Rtt, variable - depot|R_{\max}$ was proved to be polynomially solvable in (Chernykh 2016).

In this paper, we present a linear time algorithm for the $RO2|G = cycle, Rtt, variable - depot|R_{\max}$ problem, which also induces a new linear algorithm for classic open shop model. An important corollary of the result is the polynomial solvability of $RO2|Qtt, easy - TSP, variable - depot|R_{\max}$, which provides an answer to the open question mentioned above. As a by-product, we also provide an approximation result for the $RO2|Qtt, variable - depot|R_{\max}$ problem.

2 A linear time algorithm for $RO2|G = cycle, Rtt, variable - depot|R_{\max}$

Let G be a transportation network for an instance of general routing open shop problem. We use the lower bound $\overline{R}$ for the optimal makespan, defined by the formula $\overline{R} = \max_i\{\ell_i + T_i, d_{\max}\}$, where $d_{\max} = \max_j \sum_{i=1}^{m} p_{ji}$ is the *maximum job length*, $\ell_i = \sum_{j=1}^{n} p_{ji}$ is the *load* of the machine M_i, and T_i is the length of a minimal route over G for M_i.

For any list of jobs $\pi = (J_1, J_2, \dots J_n)$, define $S(\pi)$ to be an early schedule such that:
(a) the machine M_1 performs operations in order $O_{21} \to O_{31} \to \dots \to O_{n1} \to O_{11}$;
(b) the machine M_2 performs operations in order $O_{12} \to O_{22} \to O_{32} \dots \to O_{n2}$;
(c) for any job but J_1, the order of operations is $O_{j1} \to O_{j2}$.
The notation π^{+k} is used for a shifted list $(J_k, J_{k+1}, \dots, J_n, J_1, \dots, J_{k-1})$.

For $RO2|G = cycle, Rtt, variable - depot|R_{\max}$, consider the following
ALGORITHM $\mathcal{A}$:
Input: An instance of the $RO2|G = cycle, Rtt, variable - depot|R_{\max}$ problem.

1. Let $\pi = (J_1, J_2, \dots J_n)$ be a list of jobs such that in the list of respective nodes $(v_1, v_2, \dots, v_n)$ we have either $v_i = v_{i+1}$ or v_i and v_{i+1} are adjacent in G for all $i \in \{1, \dots, n-1\}$. Choose the node $v = v_1$ to be a depot.
2. If necessary, re-enumerate the machines so that $\ell_1 + T_1 \le \ell_2 + T_2$.
3. Compose a schedule $S(\pi)$.
4. If $R_{\max}(S(\pi)) = \overline{R}$, then **Output** $S(\pi)$.
   Else
   (a) Let J_k from a node u be the job that is processed after the last time the second machine idles in the schedule $S(\pi)$.
   (b) Taking u to be the depot, **Output** $S(\pi^{+k})$.

Theorem 1. ALGORITHM $\mathcal{A}$ *returns a schedule of length* $\overline{R}$ *in* $O(n)$ *time.*

Proof. Note that if $R_{\max}(S(\pi)) > \overline{R}$, then M_2 idles at some point. Indeed, if M_2 does not, then M_1 must. By definition of $S(\pi)$, the machine M_1 may only idle before starting O_{11}, which is only possible if O_{12} is processed in that idle interval. Then $R_{\max}(S_\pi) = \max\{d_1, l_2 + T_2\} \le \overline{R}$, a contradiction.

Let t be the completion moment of the last idle interval of M_2, which is also the starting time of O_{k2} for a certain $k \in \{1, \dots, n\}$. Consider the schedule $S'(\pi)$ that is obtained by shifting operations $O_{12}, \dots, O_{k-1,2}$ in $S(\pi)$ to the right so that M_2 only idles before processing of O_{21} starts as shown in Figure 1. The makespan of the schedule remains the same. Define the ***blocks*** (i.e. ordered sets of operations and travel times between the corresponding nodes) A_1, A_2, B_1, and B_2 as follows:

$A_1 = \boxed{\to O_{21} \to \dots \to O_{k1}}$; $A_2 = \boxed{\to O_{k+1,1} \to \dots \to O_{n1} \to O_{11}}$;
$B_1 = \boxed{O_{12} \to \dots \to O_{k-1,2} \to}$; $B_2 = \boxed{O_{k2} \to \dots \to O_{n2} \to}$.

The arrows denote the corresponding travel times.

Fig. 1. Example of schedule $S'(\pi)$

Let $\varDelta$ be the moment the processing of B_1 starts, so that $R_{\max}(S'(\pi)) = \varDelta + l_2 + T_2$. Note that the processing of A_2 ends at $l_1 + T_1$, and $l_1 + T_1 \le l_2 + T_2$ implies

$$\varDelta \le R_{\max}(S'(\pi)) - (l_1 + T_1). \tag{1}$$

Consider an schedule obtained by placing the block A_2 before A_1, and the block B_2 in front of B_2 as shown in Figure 2.

Fig. 2. Result of the block permutation

The schedule derived by the permutation is feasible due to the inequality (1), unless operations of J_k overlap, and in fact, it is exactly $S(\pi^{+k})$. In case O_{k2} does end later than O_{k1} starts, we obtain $S(\pi^{+k})$ by shifting O_{k1} to the right accordingly. By the construction of the schedule, the machine M_2 never idles, and M_1 may only idle before processing O_{k1}, so $R_{\max}(S(\pi^{+k}))$ is either the length of J_k, or $R_{\max}(S(\pi^{+k})) = \max\{l_1 + T_1, l_2 + T_2\} \le \overline{R}$. Hence $R_{\max}(S(\pi^{+k})) = \overline{R}$, as wanted.

It is evident that an early schedule can be obtained in linear time. Thus, Algorithm $\mathcal{A}$ runs in linear time, too. $\square$

3 Corollaries

Note that the problem $O2||C_{\max}$ is a special case of $RO2|variable - depot|R_{\max}$ when the travel time between any two nodes is zero. Thus, Algorithm $\mathcal{A}$ induces a linear algorithm for the classic two-machine open shop problem that differs qualitatively from the algorithms proposed before.

The main principle of ALGORITHM $\mathcal{A}$ is composing an early schedule such that the orders of operation processing for the two machines are identical up to cyclic permutation of jobs, with both machines following their optimal route at the same time. With that, we consider two subcases of $RO2|Rtt, variable - depot|R_{\max}$ that can be easily proved to be solvable with the use of ALGORITHM $\mathcal{A}$.

Corollary 1. *The problem $RO2|Qtt, easy - TSP, variable - depot|R_{\max}$ is solvable in time $O(n + t_{TSP})$, where t_{TSP} is the time required to solve TSP on G.*

Corollary 2. *The problem $RO2|Rtt, G = cactus, variable{-}depot|R_{\max}$ is solvable in $O(n)$.*

In case we have an approximate solution to TSP instead of an exact one, we can use ALGORITHM $\mathcal{A}$ to obtain the same approximation for $RO2|Qtt, variable - depot|R_{\max}$. In particular, by applying Christofides-Serdyukov algorithm (Christofides 1976, Serdyukov 1978), we derive the following

Corollary 3. *There exists a $\frac{3}{2}$-approximate algorithm for $RO2|Qtt, variable{-}depot|R_{max}$.*

Acknowledgements

This research was supported by the Russian Federation for Basic Research, project 20-01-00045.

References

Averbakh, I., Berman, O., Chernykh, I., 2006, “The routing open shop problem on a network: complexity and approximation”, *Eur. J. Oper. Res.*, Vol. 173(2), p. 531-539.

Chernykh, I., 2016, “Routing open shop with unrelated travel times”, In: DOOR 2016,Vladivostok, Russia, September 16-23, 2016, Proceedings, pp. 272-283.

Christofides, N., 1976, “Worst-case analysis of a new heuristic for the travelling salesman problem”, Report 388, Graduate School of Industrial Administration, CMU.

Gonzalez T., Sahni S., 1976, “Open shop scheduling to minimize finish time”, *J. ACM*, Vol. 23(4), pp. 665-21679.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A., Shmoys, D.B., 1993, Chapter 9: Sequencing and scheduling: Algorithms and complexity, In: Logistics of Production and Invertory, *Handbooks in Operations Research and Management Science*, Vol. 4, pp. 445-522. Elsevier.

Pinedo, M., Schrage, L., 1982, “Stochastic shop scheduling: a survey”, In: Deterministic and Stochastic Scheduling, Dempster, M., Lenstra, J.K., Rinnooy Kan, A. (eds.) *NATO Advanced Study Insitute Series*, Vol. 84, pp. 181-196. Springer, Dordrecht.

Serdyukov, A., 1978, “On some extremal routes routes in graphs” (in Russian), *Upravlyaemye Sistemy*, Vol. 17, pp. 76-79.

Soper, A., 2015, “A cyclical search for the two machine flow shop and open shop to minimize finish time”, *J. Sched*, Vol. 18, pp. 311-314.

de Werra, D., 1989, “Graph-theoretical models for preemptive scheduling”, In: Advances in Project Scheduling, Slowinski, R., Weglarz, J. (eds.) pp. 171-185. Elsevier, Amsterdam.

A Stochastic Programming Model to Schedule Projects under Cash Flow Uncertainty

Berfin Kutlağ[1], Nazlı Kalkan[2], Serhat Gul[1] and Öncü Hazır[3]

[1] Department of Industrial Engineering, TED University, Turkey
berfin.kutlag,serhat.gul@tedu.edu.tr
[2] Department of Industrial Engineering, Bilkent University, Turkey
nazli.kalkan@bilkent.edu.tr
[3] Rennes School of Business, 2 Rue Robert d'Arbrissel,35065, Rennes, France
oncu.hazir@rennes-sb.com

Keywords: project scheduling, stochastic programming, NPV maximization.

1 Introduction

The uncertainties inherent in project scheduling lead to a challenging problem for project managers. Many studies in the relevant literature ignore this factor even though the consideration of uncertainty is critical. Several of the stochastic project scheduling studies consider only the uncertainty in activity durations. However, there exists other uncertainty-inducing factors such as disruptions in resource usages/availabilities or delays in cash flows that are important while managing projects (see Hazir and Ulusoy 2019 for an extensive review of the subject). The delays in cash inflows are particularly common, because the financial positions of the clients depend on several uncontrollable factors. To the best of our knowledge, there does not exist any study in the project scheduling literature that considers delays in client payments.

In this article, we investigate the uncertainties associated with delays in client payments and model the effects of these in the net present value (NPV), which is a common criterion used to assess the financial feasibility of the projects. We formulated our project scheduling problem as a two-stage stochastic mixed integer program. The activity start times are the main decision variables in the model. The actual client payment times are represented as second-stage decision variables. The objective function maximizes the expected NPV.

Our research is related to two main streams of past research on project scheduling with financial objectives. The articles that proposed approaches different from stochastic programming models are grouped within the first category. Our research belongs to the second category of articles in which the stochastic programming models are formulated.

Russell (1970) is known as the leading work in the first category of articles. They maximized the NPV in the objective function and proposed a first-order Taylor series approximation based approach to linearize it. This work was followed by many others that consider NPV in the objective function (Elmaghraby and Herroelen 1990, Herroelen and Gallens 1993, Kazaz and Sepil 1996). Buss and Rosenblatt (1997) assumed uncertainty in activity durations and determined the optimal amount of delays beyond the earliest activity start times. Wiesemann et al. (2010) considered uncertainty in activity durations and cash flow amounts based on a discrete set of scenarios. They enforced nonanticipativity by imposing target process times for activities at each scenario. They solved the model using a branch and bound algorithm. Sobel et al. (2009) also considered randomness in activity durations and cash flow amounts. They investigated the optimal adaptive schedule by developing a continuous-time Markov decision chain model. Creemers et al. (2015) used a stochastic dynamic programming approach in their study where they considered technological uncertainty and stochastic activity durations. Their model incorporates the risk of activity failure which may result in project failure. Creemers (2018) maximized the ENPV in the objective function by studying with stochastic activity durations that are modeled using phase-type distributions. They used a new continuous-time Markov chain and a backward stochastic dynamic program to determine the optimal policy.

In the second category of articles, Klerides and Hadjiconstantinou (2010) developed two path-based two-stage stochastic integer programming models. The models include uncertainty in activity durations and costs. The main decision is about the execution mode of an activity. The objective functions in the models minimize the total cost and expected project duration, respectively. They proposed a decomposition-based algorithm to solve the model. Davari and Demeulemeester (2019) considered uncertainty in a resource-constrained project scheduling problem (RCPSP). To deal with the uncertainty, they studied the chance-constrained resource-constrained project scheduling problem (CC-RCPSP), which was introduced recently. They formulated the sample average approximation (SAA) counterpart of the CC-RCPSP (SAA-RCPSP) due to the large size of finite supporting set of realizations. They used a branch-and-bound algorithm (B&B) to solve the SAA-RCPSP. Lamas and Demeulemeester (2015) also modeled an RCPSP. Their model contains stochastic activity durations. They aimed to create a new procedure for generating a baseline schedule for the problem. They also studied the SAA of their original model. They - implemented a branch-and-cut algorithm to find a robust baseline schedule considering a new robustness measure that they introduced.

Our study is different from both category of articles, because we propose a two-stage stochastic programming model that consider uncertainty in client payment delays.

2 Problem Description

We formulated our problem as a two-stage stochastic mixed integer program (SMIP). The start time for each activity is set at the first stage under uncertainty related to the delay of client payment times. The actual times of cash inflows, which depend on activity completion times and delays in payments, are modeled as the second-stage variables. The cash outflows are observed at the beginning of each activity, however the inflows are observed at the completion of a given set of activities. A deadline is enforced on the project completion time, but the client payments are allowed to be received after this deadline. The objective is to maximize the expected net present value of the project. Following is a detailed description of our formulation.

Indices and Sets:

i, j: node (i.e. project activity) index
(i, j): index of the arc from node i to node j
t: time index
ω: scenario index
V: set of all nodes
V^I: set of cash-generating activities
E: set of all arcs (i.e. immediate precedence relationships)
T^i: set of time periods at which activity $i \in V$ can start (i.e. time periods between the earliest and latest start times for an activity)
P^i: set of time periods at which payment can be received for the completion of activity $i \in V^I$ (i.e. time periods after the earliest completion time for an activity)
Ω: set of all scenarios

Parameters:

p_i: duration of activity $i \in V$
$\overline{d}$: deadline for the completion of the project

n: number of activities of the project, excluding dummy nodes for project beginning (node 0) and project completion (node $n+1$)
c_i^{F+}: cash inflow due to the completion of activity $i \in V^I$ ($c_i^{F+} > 0$)
c_i^{F-}: cash outflow due to the initiation of activity $i \in V$ ($c_i^{F-} < 0$)
β: discount rate per time period
e_i^{ω}: delay in payment after the completion of activity $i \in V^I$ under scenario $\omega \in \Omega$

First-Stage Decision Variables:

$$x_{it} = \begin{cases} 1 & \text{if activity } i \in V \text{ starts at time } t \in T^i; \\ 0 & \text{otherwise,} \end{cases}$$

Second-Stage Decision Variables:

$$q_{it}^{\omega} = \begin{cases} 1 & \text{if payment for activity } i \in V^I \text{ is received at time } t \in P^i \text{ under scenario } \omega \in \Omega; \\ 0 & \text{otherwise,} \end{cases}$$

$$\max \quad \sum_{i \in V} \sum_{t \in T^i} \frac{c_i^{F-}}{(1+\beta)^t} x_{it} + \mathcal{Q}(\mathbf{x}) \tag{1}$$

s.t.

$$\sum_{t \in T^i} x_{it} = 1 \qquad \forall i \in V \tag{2}$$

$$\sum_{t \in T^j} t x_{jt} \geq \sum_{t \in T^i} t x_{it} + p_i \qquad \forall (i,j) \in E \tag{3}$$

$$\sum_{t \in T^{n+1}} t x_{(n+1)t} + p_{n+1} \leq \bar{d} \tag{4}$$

$$x_{it} \in \{0,1\} \qquad \forall i \in V, t \in T^i \tag{5}$$

where $\mathcal{Q}(\mathbf{x}) = E_{\xi}[\mathrm{Q}(\mathbf{x}, \xi(\omega))]$ is the expected recourse function, and

$$\mathrm{Q}(\mathbf{x}, \xi(\omega)) = \max \sum_{i \in V^I} \sum_{t \in P^i} \frac{c_i^{F+}}{(1+\beta)^t} q_{it}^{\omega}$$

s.t.

$$\sum_{t \in P^i} t q_{it}^{\omega} = \sum_{t \in T^i} t x_{it} + p_i + e_i^{\omega} \qquad \forall i \in V^I \tag{6}$$

$$\sum_{t \in P^i} q_{it}^{\omega} = 1 \qquad \forall i \in V^I \tag{7}$$

$$q_{it}^{\omega} \in \{0,1\} \qquad \forall i \in V^I, t \in P^i \tag{8}$$

The objective function (1) includes the net present value of the summation of the cash outflows that depend on the activity start times, and the expected second-stage function. The expected second-stage function maximizes the net present value of the cash inflows that occur after a possible delay following the activity completion time. First-stage constraints are represented by (2)-(5). We assume that the earliest/latest start times of each activity were calculated in advance using the forward-backward passes. Constraints (2) ensure that an activity starts in between its earliest and latest start time. Constraints (3) maintain that the start time of an activity is greater than or equal to the completion time of the activity that immediately precedes it. Constraints (4) require that the project is completed before the deadline. Constraints (5) enforce binary restrictions on the first-stage variables.

In the second stage, constraints (6) calculate the time of cash inflow by considering possible delay after the activity completion time.Constraints (7) ensure that the cash inflow for a completed activity is received as a lump-sum amount at a single period. Constraints (8) represent the binary restrictions on the second-stage variables.

3 Conclusion

We apply a sample average approximation (SAA) algorithm to solve the SMIP model (Kleywegt et al. 2002). The SAA algorithm approximates the true objective value by solving instances created by sampling N scenarios. The algorithm can be used to assess the optimality gap as well as for obtaining a solution. The SAA solves M instances, each having N scenarios to obtain an estimate of the lower bound. Then, an upper bound is calculated for each instance solution by evaluating its objective value over N' scenarios. Note that N' is generally set to a much larger value than N.

In our experiments, we intend to illustrate the impact of randomness in the delay of client payments into the activity start times. We also show the benefit of considering uncertainty in payment delays. We examine how the deadline constraint (i.e. constraint (4)) affects the net present value and optimal activity start times.

References

Buss A.H., M.J. Rosenblatt, 1997, “Activity Delay in Stochastic Project Networks", *Operations Research*, Vol. 45, pp. 129-139.

Creemers S., 2018, “Maximizing the expected net present value of a project with phase-type distributed activity durations: An efficient globally optimal solution procedure", *European Journal of Operational Research*, Vol. 267, pp. 16-22.

Creemers S., B. De Reyck and R. Leus, 2015, “Project planning with alternative technologies in uncertain environments", *European Journal of Operational Research*, Vol. 242, pp. 465-476.

Davari M., E. Demeulemeester, 2018, “A novel branch and bound algorithm for the chance-constrained resource-constrained project scheduling problem", *International Journal of Production Research*, Vol. 57, pp. 1265-1282.

Elmaghraby S.E., W.S. Herroelen, 1990, “The scheduling of activities to maximize the net present value of projects", *European Journal of Operational Research*, Vol. 49, pp. 35-49.

Hazir O., G. Ulusoy, 2019, “A classification and review of approaches and methods for modeling uncertainty in projects", *International Journal of Production Economics*, in press.

Herroelen W.S., E. Gallens, 1993, “Computational experience with an optimal procedure for the scheduling of activities to maximize the net present value of projects", *European Journal of Operational Research*, Vol. 65, pp. 274-277.

Kazaz B., C. Sepil, 1996, “Project Scheduling with Discounted Cash Flows and Progress Payments", *Journal of Operational Research Society*, Vol. 47, pp. 1261-1272.

Klerides E., E. Hadjiconstantinou, 2010, “A decomposition-based stochastic programming approach for the project scheduling problem under time/cost trade-off settings and uncertain durations", *Computers and Operations Research*, Vol. 37, pp. 2131-2140.

Kleywegt A.J., A. Shapiro and T. Homem-de-Mello, 2002, “The sample average approximation method for stochastic discrete optimization", *SIAM Journal on Optimization*, Vol. 12, pp. 479-502.

Lamas P., E. Demeulemeester, 2015, “A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations", *Journal of Scheduling*, Vol. 19, pp. 409-428.

Russell A.H., 1970, “Cash flows in networks", *Management Science*, Vol. 16, pp. 357-372.

Sobel M.J., J.G. Szmerekovsky and V. Tilson, 2009, “Scheduling projects with stochastic activity duration to maximize expected net present value", *European Journal of Operational Research*, Vol. 198, pp. 667-705.

Wiesemann W., D. Kuhn and B. Rustem, 2010, “Maximizing the net present value of a project under uncertainty", *European Journal of Operational Research*, Vol. 202, pp. 356-367.

Multi-Objective Robotic Assembly Line Balancing Problem: A NSGA-II Approach Using Multi-Objective Shortest Path Decoders

Lahrichi Y., Deroussi L., Grangeon N. and Norre S.

LIMOS-CNRS, Université Clermont Auvergne, Aubière, France
{youssef.lahrichi,laurent.deroussi,nathalie.grangeon,sylvie.norre}@uca.fr

Keywords: Robotic Assembly Line Balancing Problem, Multi-objective optimization, NSGA-II, Multi-objective decoder, Shorthest path.

1 Problem statement

The Robotic Assembly Line Balancing Problem (RALBP) is a combinatorial optimization problem that is concerned with simultaneously assigning a set of operations to a set of workstations placed among a serial assembly line and assigning to each workstation a type of robot. The processing time of an operation i depends on the type of robot r used and is denoted d_i^r. Each type of robot r is also characterised by its cost c_r. Besides, operations are linked by precedence relations. The workload of a workstation represents the sum of the processing times of the operations assigned to it. The cycle time stands for the maximum workload among the stations and is a key performance indicator of the assembly line.

The RALBP is of significant importance due to the growing robotization of assembly lines. In this study, we consider an additional parameter which is sequence-dependent setup times: in addition to processing times, setup times $t_{i,i'}^r$ should be considered if operation i is performed just before operation i' in some workstation equipped by a robot of type r. The workload of a workstation is the sum of processing times and sequence-dependent setup times induced by the operations assigned to it. Sequence-dependent setup times raises an additional decision which is the sequencing of operations in each workstation. Sequence-dependent setup times have been rarely considered in literature for the RALBP despite their industrial importance. We study the problem while minimizing simultaneously three objectives:

(Z_1): Minimizing cycle time,
(Z_2): Minimizing the total cost of robots used,
(Z_3): Minimizing the number of workstations used.

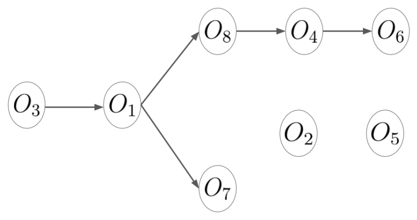
The problem has been introduced in Rubinovitz et al. (1993) where the basic assumptions are presented. Most authors consider the single objective of minimizing cycle time as Nilakantan et al. (2015) and Borba et al. (2018) and few perform multi-objective study (Yoosefelahi et al. (2012), Çil et al. (2016)). Sequence-dependent setup times have not been considered until very recently in Janardhanan et al. (2019). Table 1 positions our study in literature.

2 Example

We illustrate the problem with a small instance. We consider 8 operations and 3 types of robots. Precedence relations are illustrated in the precedence graph (Fig. 1). Processing times and sequence-dependent setup times are supposed given. A feasible solution is depicted in Fig. 2.

Table 1. Position of our study in the literature.

Article	Objectives Z_1	Z_2	Z_3	Sequence-dependent setup times
Rubinovitz et al. (1993)			✓	
Levitin et al. (2006)	✓			
Gao et al. (2009)	✓			
Yoosefelahi et al. (2012)	✓	✓		
Nilakantan et al. (2015)	✓			
Çil et al. (2016)	✓	✓	✓	
Borba et al. (2018)	✓			
Janardhanan et al. (2019)	✓			✓
Our study	✓	✓	✓	✓



$$\begin{tikzcd} & O_8 \arrow[r] & O_4 \arrow[r] & O_6 \\ O_3 \arrow[r] & O_1 \arrow[u] \arrow[d] & O_2 & O_5 \\ & O_7 & & \end{tikzcd}$$

Fig. 1. Precedence graph

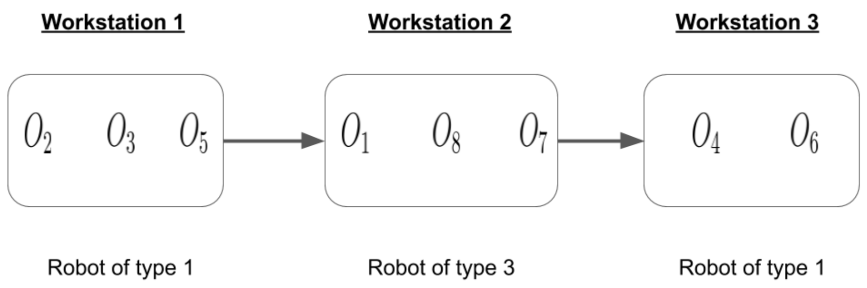


Fig. 2. Feasible solution

Let's compute the cost of this solution:

(Z_1) The total cost of robots used:

$$Z_1 = c_1 + c_3 + c_1$$

where c_1 is the cost of a robot of type 1 and c_3 is the cost of a robot of type 3.

(Z_2) Cycle time : Its is obtained by computing the maximum among the workloads of the workstations:

– On the first workstation, the workload is given by:

$$d_2^1 + t_{2,3}^1 + d_3^1 + t_{3,5}^1 + d_5^1 + t_{5,2}^1$$

– On the second workstation, the workload is given by:

$$d_1^3 + t_{1,8}^3 + d_8^3 + t_{8,7}^3 + d_7^3 + t_{7,1}^3$$

– On the third workstation, the workload is given by:

$$d_4^1 + t_{4,6}^1 + d_6^1 + t_{6,4}^1$$

(Z_3) The number of workstations used:

$$Z_3 = 3$$

3 Optimization method

We first derive a pseudo-polynomial time exact algorithm to compute all the Pareto optimal solutions provided the giant sequence of operations is given (in Figure 2, the giant sequence is $O_2, O_3, O_5, O_1, O_8, O_7, O_4, O_6$). We use then this algorithm as a decoder in a NSGA-II metaheuristic. For this purpose, we suggest a generalization of NSGA-II that supports multi-objective decoders.

3.1 Fixed giant sequence

We suppose that the giant sequence σ is fixed. Without lose of generality, we suppose $\sigma = (1, 2, \ldots, n)$. The problem is equivalent to finding a multi-criteria shortest path in an auxiliary graph $\mathcal{H}_\mathcal{I}(\sigma)$.

$\mathcal{H}_\mathcal{I}(\sigma) = (V, A)$ is a bi-valued oriented multi-graph. $V = \{0, 1, \ldots, n\}$ is the set of vertices. Vertex i $(i > 0)$ represents operation O_i while vertex 0 is fictitious. A is the multi-set (each element can have several duplicates) of arcs. It contains all arcs from i to j where $i < j$. An arc (i, j) represents a workstation to which the sequence of operations $O_{i+1}, O_{i+2}, \ldots, O_j$ is assigned. Each arc (i, j) is duplicated as many times as there are robots type r. A duplicate of (i, j) for robot type r is denoted $(i, j)_r$. The graph is bi-valued, each arc $(i, j)_r$ is weighted in $\mathbb{R}^2$ as follows:

$$w((i,j)_r) = [c_r, \sum_{k=i+1}^{j} d_k^r + \sum_{k=i+1}^{j-1} t_{k,k+1}^r + t_{j,i+1}^r]$$

The first weight represents the cost of the robot used in the workstation and the second weight represents the workload of the workstation, which means the duration to perform the sub-sequence $(i + 1, \ldots, j)$ with a robot of type r.

A path from vertex 0 to vertex n stands for a feasible solution of the balancing sub-problem. Solving optimally the multi-objective balancing problem given a giant sequence σ can be done by solving the multi-objective shortest path problem from vertex 0 to vertex n in $\mathcal{H}_\mathcal{I}(\sigma)$ minimizing simultaneously the sum of the first weights among the path, the sum of the second weights among the path and the number of arcs in the path.

All Pareto optimal solutions for the multi-objective shortest path problem can be computed thanks to a pseudo-polynomial algorithm (which we denote Split) within $O(c_{max}.N_r.n^3)$ where c_{max} is the maximum cost of a robot, N_r the number of robot types and n the number of operations.

3.2 General case

We derive an approximate method embedding the split to solve the problem in the general case where the giant sequence is not given. To encode a solution, we use a giant sequence. The split algorithm is used to decode a giant sequence. The general scheme of the metaheuritic is a NSGA-II (Non-dominated Sorting Genetic Algorithm). However, NSGA-II does not support multi-objective decoders, i.e decoders yielding several non-dominated solutions. For this reason we suggest a novel generalization of NSGA-II supporting multi-objective decoders.

4 Conclusion

In this study, a pseudo-polynomial algorithm is presented for solving the RALBP with sequence-dependent setup times given a fixed giant sequence in a multi-objective context. Then we derive an approximate method for solving the problem in the general case using a novel generalization of NSGA-II. Experiments are actually being held and the first results are encouraging.

Acknowledgements

The authors acknowledge the support received from the *Agence Nationale de la Recherche* of the French government through the program "Investissements d'Avenir"(16-IDEX-0001 CAP 20-25).

References

Borba, Leonardo, Marcus Ritt, and Cristóbal Miralles. 2018. “Exact and heuristic methods for solving the robotic assembly line balancing problem.” *European Journal of Operational Research* 270 (1): 146–156.

Çil, Zeynel Abidin, Süleyman Mete, and Kürşad Ağpak. 2016. “A goal programming approach for robotic assembly line balancing problem.” *IFAC-PapersOnLine* 49 (12): 938–942.

Gao, Jie, Linyan Sun, Lihua Wang, and Mitsuo Gen. 2009. “An efficient approach for type II robotic assembly line balancing problems.” *Computers & Industrial Engineering* 56 (3): 1065–1080.

Janardhanan, Mukund Nilakantan, Zixiang Li, Grzegorz Bocewicz, Zbigniew Banaszak, and Peter Nielsen. 2019. “Metaheuristic algorithms for balancing robotic assembly lines with sequence-dependent robot setup times.” *Applied Mathematical Modelling* 65:256–270.

Levitin, Gregory, Jacob Rubinovitz, and Boris Shnits. 2006. “A genetic algorithm for robotic assembly line balancing.” *European Journal of Operational Research* 168 (3): 811–825.

Nilakantan, J Mukund, Sivalinga Govinda Ponnambalam, N Jawahar, and Ganesan Kanagaraj. 2015. “Bio-inspired search algorithms to solve robotic assembly line balancing problems.” *Neural Computing and Applications* 26 (6): 1379–1393.

Rubinovitz, Jacob, Joseph Bukchin, and Ehud Lenz. 1993. “RALB–A heuristic algorithm for design and balancing of robotic assembly lines.” *CIRP annals* 42 (1): 497–500.

Yoosefelahi, A, M Aminnayeri, H Mosadegh, and H Davari Ardakani. 2012. “Type II robotic assembly line balancing problem: An evolution strategies algorithm for a multi-objective model.” *Journal of Manufacturing Systems* 31 (2): 139–151.

The Group Shop Scheduling Problem with power requirements

Damien Lamy[1] and Simon Thevenin[2]

[1] Mines Saint-Etienne, Institut Henri Fayol, F - 42023 Saint-Etienne, France
`Damien.lamy@emse.fr`
[2] IMT Atlantique, LS2N - UMR CNRS 6004, F-44307 Nantes, France
`simon.thevenin @imt-atlantique.fr`

Keywords: Scheduling, Energy-Efficiency, Group-shop, Linear Programming.

1 Introduction

The last few years have seen a growing interest in reducing the energy consumption of production systems since they are responsible for more than 50% of the global delivered energy worldwide (U.S. Energy Information Administration 2016). Besides technological advances, the consideration of energy consumption during operations management is an efficient approach to reduce energy wastes. Three main energy efficiency measures exist in scheduling : (1) total energy consumption; (2) time-of-use pricing; (3) peak power limit. This work focuses on power peak constraint, which prevents to exceed the power threshold contracted between the factory and its energy supplier. In other words, this constraint prevents the simultaneous processing of multiple operations with high energy consumption. Most of the literature on scheduling with energy constraints concerns the total energy consumption and time-of-use pricing, and very few works consider peak power limitations (Giret *et. al.* (2015)). Recently, Kemmoe *et. al.* (2017) proposed a method for the job shop scheduling problem with power thresholds. In the present work, we investigate the extent to which a more flexible shop-floor organization (namely, the group shop) improves the productivity under power limitations.

The Group-Shop Scheduling Problem (GSP) generalises the Job-Shop Scheduling Problem (JSP) and the Open-Shop Scheduling Problem (OSP). On the one hand, in the JSP, jobs are composed of operations to schedule according to pre-given routes. On the other hand, in an OSP the routing is a decision of the scheduling problem. The GSP stands at the frontier of these two problems since it gives the operations' routing partially. As the JSP and the OSP are NP-hard (Garey and Johnson 1979), the GSP is NP-hard too. Therefore, multiple metaheuristics have been proposed to solve the GSP, such as the ant colony optimization (Blum and Sampels 2004), tabu search and simulated annealing (Liu *et. al.* 2005), genetic algorithms (Ahmadizar and Shahmaleki 2014). In addition, some extension of the classical GSP have been considered, such as stochastic processing time and release dates (Ahmadizar*et. al.* 2010), or the GSP with sequence-dependent setup and transportation times (Ahmadizar and Shahmaleki 2014). However, to the best of our knowledge, the present work is the first to consider the GSP with power limitation constraints. The closest work is Liu *et. al.* (2019), where the authors consider an ultra-flexible Job-shop, but the objective is to minimize total energy consumption rather than to schedule operations with a power limitation.

The rest of the paper is organized as follows. Section 2 gives a formal description of the considered problem and a mixed-integer linear programming formulation, and Section 3 reports experimental results that assesses the impact of the flexibility offered by the group-shop to efficiently schedule operations subjected to a power limitation. Finally, a conclusion ends the paper.

2 Problem description

This section formally states the Group-Shop scheduling Problem with Power Requirements (GSPPR), before to give its mathematical formulation.

The GSPPR is to schedule a set of n jobs, where each job j consists of a set $\mathcal{O}_j = \{O_{j1} \dots O_{jm}\}$ of operations to perform on machine $M_1 \dots M_m$, respectively. Each operation k of the entire set of operation $\mathcal{O}$ is associated with a duration P_k and a power requirement W_k. The objective of the considered problem is to minimize the makespan c_{max}, that is, the completion time of the last performed job. However, the schedule must respect some precedence constraints between the operations. More precisely, the set of operations of a job j is partitioned into groups, and G_{jk} denotes the k^{th} group of job j. The precedence constraints require to complete all the operations of the group G_{jk} before the start of any operation of the group G_{jl} if $k \leq l$. However, the operations of a group can be scheduled in any order. In addition, the schedule must respect the energy threshold, that is the total energy consumption of the operations performed simultaneously must be lower than the threshold W_{max}. Finally, the operations are non-preemptive and available at time 0.

In short, the GSPPR requires to schedule all operations efficiently without exceeding the power threshold. Note that the GSP generalizes the JSP and the OSP. Indeed, an instance of the GSP with a single operation per group is an instance of the JSP, and an instance of the GSP with a single group per job is an instance of the OSP.

The disjunctive formulation with the flow representation of energy is classically used for the JSP (Kemmoe *et. al.* 2017), and the model (1) - (8) is the adaptation of this formulation for the GSPPR. Model (1) - (8) is based on the following variables:

- x_{ij} is equal to 1 if operation i is processed before operation j, and 0 otherwise
- ϕ_{ij} represents the flow transferred from operation i to j
- s_i is the starting time of operation i

$$\min c_{max} \tag{1}$$

s.t.

$$c_{max} \geq s_i + P_i \qquad \forall \quad i \in \mathcal{O} \tag{2}$$

$$s_j \geq s_i + P_i - M(1 - x_{ij}) \qquad \forall \quad i, j \in \mathcal{O} \tag{3}$$

$$\sum_{j \in \mathcal{O}} \phi_{0j} \leq W_{max} \tag{4}$$

$$\phi_{ij} \leq x_{ij} W_i \qquad \forall \quad i, j \in \mathcal{O} \tag{5}$$

$$\sum_{j \in \mathcal{O} - \{i\}} \phi_{ij} \leq W_j \qquad \forall \quad i \in \mathcal{O} \tag{6}$$

$$\phi_{0j} + \sum_{i \in \mathcal{O} - \{j\}} \phi_{ij} = W_i \qquad \forall \quad j \in \mathcal{O} \tag{7}$$

$$x_{ij} + x_{ji} = 1 \qquad \forall \quad i, j \in \mathcal{O} \tag{8}$$

To respect the precedence constraint, $x_{oo'}$ is set to 1 if operations o and o' belong to the same job j, $o \in G_{jk}$ and $o' \in G_{jl}$ with $k \leq l$. Equations (3) compute the start time of each operation based on its predecessors, and equations (2) set the makespan to the completion time of the last operation. The energy consumption is modeled with a flow. Constraint (4) ensures that the source transmits at most W_{max} units of energy in total, and Constraints (5) states that each operation can transmit the energy flow to one of its successors only. Equations (6) state that each operation j must receive W_j units of energy, whereas equations (7) forbid an operation to transmit more energy than it received. Finally, the redundant constraints (8) are introduced to strengthen the formulation.

3 Computational experiments

As this paper is the first to consider the GSPPR, no instances exist in the literature. Therefore, we generated the instances randomly, with a number of jobs and machines selected in the interval $[3, 10]$, and operations assigned randomly to groups. The duration of each operation i, P_j, is generated randomly in the interval $[1, 100]$, while its power requirement is generated randomly in the interval $[1, 30]$. Finally, three different values for the power limit W_{max} are considered: MaxThreshold (i.e. enough power to process all operations), MaxThreshold/2 and MaxThreshold/3. For each couple (m, W_{max}), 10 instances are generated.

The integer linear program (1) - (8) is implemented with CPLEX 12.8, and the experiments were run on a Xeon E3-1505M processor with a time limit of 600 seconds.

Table 1. Results on small instances with different power thresholds

m	MaxThreshold			MaxThreshold/2			MaxThreshold/3		
	AVG_CPU(s)	AVG_GAP(%)	NB_OPT	AVG_CPU(s)	AVG_GAP(%)	NB_OPT	AVG_CPU(s)	AVG_GAP(%)	NB_OPT
3	92.75	3.94	7	456.125	27.29	2	526	50.73	1
4	222.75	5.71	6	383	28.81	3	600	66.17	0
5	224.5	6.49	6	450.25	31.05	2	600	68.12	0
6	237.5	6.24	5	388.625	23.46	3	539	53.87	1
7	280.75	10.29	5	450.5	33.41	2	526	57.96	1
8	328.625	12.36	4	455.375	32.47	2	525	55.07	1
9	300.5	11.38	5	402.125	27.00	3	600	45.31	1
10	451.25	23.90	2	457.625	40.36	2	526	50.40	1

Table 1 reports the performance of the CPLEX solver for different power thresholds. Each row of the table corresponds to a set of instances with the same number of machines (m). When the power threshold is high, the optimal solutions (see NB_OPT) is easy to reach. On the contrary, CPLEX has some difficulties to find optimal solutions for low power thresholds (closed to the minimal value under which it is not possible to schedule operations). Actually, CPLEX was not able to find an upper bound for a fifth of the instances in this scenario.

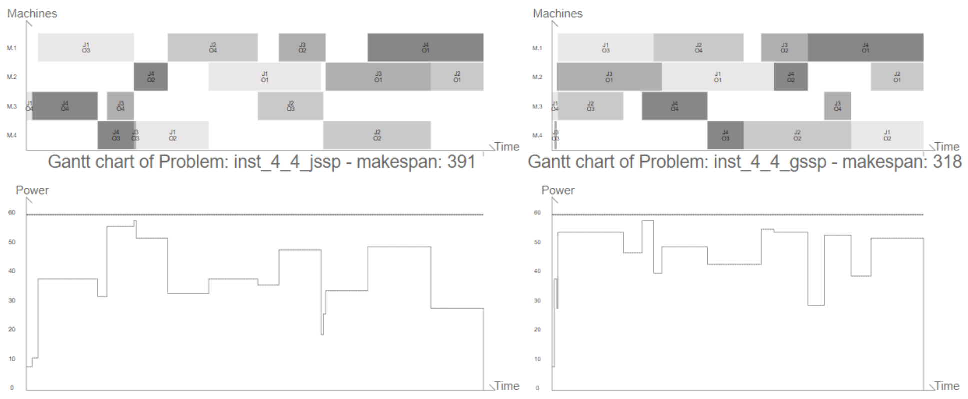


Fig. 1. Gantt chart of the optimal solution of the GSP (on the right), and the solution of the Job-shop instance (on the left) created by adding random precedences between the operations of a group.

Figure 1 compares the makespan in GSP and JSP. The left side gives the Gantt chart of the optimal solution of a GSPPR instance, whereas the left side shows the solution of JSSPR instance obtained by adding random precedences between the operations of each group. The GSPPR instance has a makespan of 318 versus 391 for JSPPR. Moreover, the power thresholds of the GSPPR instance can be reduced up to 30%, and the makespan remains lower than the one of the JSPPR with the initial power limit.

In production management, the process plans (the operation to perform and their order) are classically decided before to schedule the operation. This study shows that integrating these decision yields some flexibility on the shop floor, and this allows better performance, or to operate with lower energy thresholds.

4 Conclusion

This paper investigates the problem of minimizing the makespan in a Group-shop Scheduling Problem with power requirements and a power limitation (GSPPR). As the Group-shop allows some flexibility in the processing order of the operations of a job, preliminary results show that the Group-shop leads to a significant reduction of the makespan in the context of power-constrained schedules when compared to the classical Job-shop. For instance, an operation with a low power requirement can be scheduled at the right moment, when the available power is not used by other operations. As CPLEX solves small size instances only, future works include the development of metaheuristics and constrained programming approaches for the GSPPR. In addition, extensions of the GSPPR are of practical interest. For instance, the present model contains only operations with constant power requirements, which is close to cumulative problems (as stressed in Baptiste *et. al.* (2001)), and it could be extended to more real power profiles. Also, in the presence of human operators, there exist some uncertainties on the processing time of the operation. These random processing times lead to random power limit excesses, and the design of schedule robust to these uncertainties is crucial to avoid exceeding contracts based on power thresholds.

References

Ahmadizar, F., Ghazanfari, M. and Fatemi Ghomi, S.M.T. 2010, "Group shops scheduling with makespan criterion subject to random release dates and processing times", *Computers Operations Research*, Vol. 37 No. 1, pp. 152-162.

Ahmadizar, F. and Shahmaleki, P., 2014, " Group-shop scheduling with sequence-dependent set-up and transportation times", *Applied Mathematical Modelling*, Vol. 38 No. 22, pp. 5080-5091.

Baptiste P., Le Pape C., Nuijten W., 2001, " Cumulative Scheduling Problems", *In: Constraint-Based Scheduling. International Series in Operations Research Management Science*, Vol. 39., pp. 149-158.

Blum, C. and Sampels, M., 2004, "An ant colony optimization algorithm for shop scheduling problems", *Journal of Mathematical Modelling and Algorithms*, Vol. 3 No. 3, pp. 285-308.

Garey, M.R. and Johnson, D.S., 1979. "Computers and intractability", vol. 29, *New York: wh freeman.*

Giret, A., Trentesaux, D. and Prabhu, V., 2015, "Sustainability in manufacturing operations scheduling: A state of the art review", *Journal of Manufacturing Systems*, Vol. 37, pp. 126-140.

Kemmoe, S., Lamy, D. and Tchernev, N., 2017, "Job-shop like manufacturing system with variable power threshold and operations with power requirements", *International Journal of Production Research*, Vol. 55 No. 20, pp. 6011-6032.

Liu, N., Zhang, Y.F. and Lu, W.F., 2019, "Improving Energy Efficiency in Discrete Parts Manufacturing System Using an Ultra-Flexible Job Shop Scheduling Algorithm", *International Journal of Precision Engineering and Manufacturing-Green Technology.*

Liu, S.Q., Ong, H.L. and Ng, K.M., 2005, "A fast tabu search algorithm for the group shop scheduling problem", *Advances in Engineering Software*, Vol. 36 No. 8, pp. 533-539.

A two-stage robust approach for minimizing the weighted number of tardy jobs with profit uncertainty

Henri Lefebvre[1], François Clautiaux[2] and Boris Detienne[2]

[1] DEI, University of Bologna, Italy
henri.lefebvre@unibo.it
[2] IMB, University of Bordeaux, Inria Bordeaux Sud-Ouest, France
francois.clautiaux,boris.detienne@math.u-bordeaux.fr

Keywords: Single machine scheduling, Robust optimization, Exact algorithm.

1 Introduction

We investigate a stochastic variant of the well-know $1|r_j|\sum w_j U_j$ problem, in which the jobs are subject to unexpected failure which leads to additional costs. The decision maker is then allowed to take recourse actions such as outsourcing or spending more time on the jobs to fix them. We are interested in worst-case optimization, with polyhedral uncertainty set affecting the objective function.

In our problem, called *Two-Stage Robust Weighted Number of Tardy jobs* (2SRWNT) in the sequel, an instance consists of a set of jobs $\mathcal{J}$, each of which is characterized by a release date r_j, a due date d_j, and a nominal processing time p_j. A weight w_j can be interpreted as the cost for executing the job tardy, or the opposite of the profit of processing the job on time. At the first stage, *here-and-now* decisions are to select a subset of jobs $\mathcal{J}^* \subseteq \mathcal{J}$ to process. After that, a subset of the jobs can be affected by unexpected failures, those being governed by the uncertainty set $\Xi = \left\{\xi \in \mathbb{R}_+^{|J|} \;\middle|\; \xi_j \leq 1, \forall J_j \in \mathcal{J} \text{ and } \sum_{j|J_j\in\mathcal{J}} \xi_j \leq \Gamma\right\}$. The realization of alea $\xi \in \Xi$ determines a profit degradation for each job $J_j \in \mathcal{J}$ defined as $\delta_j(\xi) = \bar{\delta}_j \xi_j$, where $\bar{\delta}_j$ is the maximum additional cost linked to the job's failure. Input parameter Γ is the largest number of jobs that can incur their maximum degradation. At the second stage *recourse* actions have to be taken. For each $j \in \mathcal{J}^*$, one can choose *(i)* to keep the revealed profit ; *(ii)* to repair the job, adding τ_j time units to its processing time to recover its initial profit ; or *(iii)* to reject the job, and pay a fixed outsourcing cost f_j. Finally, jobs in $\mathcal{J}^*$ that are not rejected must be scheduled so that they meet their time windows. The objective is to select a subset of jobs as well as the recourse actions that minimize the worst-case overall cost (equivalently, maximizes the overall worst-case profit).

(van den Akker, Hoogeveen and Stoef 2018) also study a variant of $1||\sum U_j$ where the processing times are uncertain. Given a discrete scenario-based uncertainty set, one has to determine an initial, feasible for nominal processing times, sequence of jobs. At second stage, once the scenario of actual processing times is revealed, the sequence must be made feasible for those actual processing times by rejecting some jobs. The objective is to minimize the expected cost of the repaired solution. Exact methods are proposed for this problem. Our study differs by the basic problem (we consider unequal release dates and weights), the nature of the uncertainty set (polyhedral vs. discrete, scenario-based), the uncertain data (objective vs. constraints) and the possible recourse actions.

Robustness is known to be a hard issue in scheduling. (Aloulou and Della Croce 2008) and (Yang and Yu 2002) show that even simple scheduling problems become $\mathcal{NP}$-hard as soon as the uncertainty set contains more than one scenario. A possible way to address our problem is to use the so-called finite adaptability model of (Bertsimas and Caramanis 2010). This heuristic approach consists in restricting the problem by determining at first stage a set of K recourse solutions, while the second stage is reduced to choosing the best of those for the revealed alea. On the one hand, when K is small enough, this approach has the advantage to produce tractable problems. On the other hand, it may produce suboptimal solutions, since it restricts the number of resource actions that can be performed.

The contribution of this abstract is to propose the first exact method for this problem. It is based on an MILP formulation based on a recent result of (Arslan and Detienne 2018). We solve the model using a branch-and-price algorithm.

2 Mixed Integer Linear Programming model

We first recall the idea of the ILP model proposed in (Detienne 2014) for $1|r_j|\sum w_j U_j$, which we extend to the robust case. Their approach is based on the fact that minimizing the weighted number of tardy jobs can be decomposed into two distinct decisions: (1) decide which jobs are to be executed tardy and (2) in what order the on-time jobs are executed. We know that if the jobs have agreeable time windows (i.e., that the tasks can be ordered in such a way that for each J_i before J_j we have $r_i \leq r_j$ and $d_i \leq d_j$), then a feasible sequence of on-time jobs exists iff the earliest due-date first rule (EDD) yields a feasible solution. The main idea of (Detienne 2014) is to reformulate the general $1|r_j|\sum w_j U_j$ problem into a problem of selecting jobs with agreeable time windows. To do so, a set of so-called *job occurrences* is created from the original set of jobs in such a way that EDD may still be applied. Formally, consider a job $J_i \in \mathcal{J}$. For any job $J_j \in \mathcal{J}$ whose time window is not agreeable with that of J_i (i.e., $r_i < r_j$, $d_i > d_j$, and $r_i + p_i + p_j \leq d_j$), we create a job occurrence $J_k \in \widetilde{\mathcal{J}}$ such that $r_k = r_i, p_k = p_i, w_k = 0, f_k = f_i, \bar{\delta}_k = \bar{\delta}_j, \tau_k = \tau_j$ and a hard deadline $\bar{d}_k = d_j$ and which represents the scheduling of J_i before J_j. The original job J_i is also added to the set of job occurrences $\widetilde{\mathcal{J}}$, with a null weight as well. We define, for every job $J_j \in \mathcal{J}$, $\mathcal{G}_j$ as the set gathering all the job occurrences related to J_j. The following proposition, established in (Detienne 2014), naturally extends to the robust case.

Proposition 1. *There is at least one optimal solution of 2SRWNT such that selected job occurrences are scheduled according to a non-decreasing order of their deadlines with ties being broken in a non-decreasing order of their release dates.*

In the remainder, we assume that job occurrences are sorted according to a non-decreasing order of their deadlines and denote by $\bullet_k$ the data $\bullet$ of the kth occurrence in that order (e.g., p_k now denotes the processing time of the kth job occurrence in that order). Similarly to (Detienne 2014), we assign reversed time windows to each job occurrence given by $[\hat{r}_j, \hat{d}_j] = [\max_i d_i - d_j, \max_i d_i - r_j]$, which helps writing an ILP model with a stronger linear relaxation.

For every job $J_j \in \mathcal{J}$, we introduce decision variable U_j which is equal to 1 if J_j is tardy, 0 otherwise. For every job occurrence $J_k \in \mathcal{G}_j$, we denote by y_k the selection variable of the kth job occurrence, and z_k the decision variable indicating whether the job occurrence is repaired or not. More precisely, if $U_j = 0$, then J_j is decided to be executed on-time in the first stage. Once the uncertainty is revealed, the sequencing of the jobs and the recourse actions have to be decided. The following cases may arise: *(i)* $\exists J_k \in \mathcal{G}_j, y_k = z_k = 1$, *i.e.* the job is executed and repaired ; *(ii)* $\exists J_k \in \mathcal{G}_j, y_k = 1$ and $z_k = 0$, *i.e.* the job is executed and the deteriorated profit is undertaken ; *(iii)* $\forall J_k \in \mathcal{G}_j, y_k = z_k = 0$, the job is outsourced. Let us introduce the set $\mathcal{Y} \subset \{0,1\}^{|\widetilde{\mathcal{J}}|} \times \{0,1\}^{|\widetilde{\mathcal{J}}|} \times \mathcal{R}_+^{|\widetilde{\mathcal{J}}|}$ of all feasible second-stage solutions:

$$\mathcal{Y} = \begin{cases} \rho_k = p_k y_k + \tau_k z_k \quad \forall k | J_k \in \widetilde{\mathcal{J}} & (1) \\ z_k \leq y_k \quad \forall k | J_k \in \widetilde{\mathcal{J}} & (2) \\ \sum_{k | J_k \in \mathcal{G}_j} y_k \leq 1 \quad \forall J_j \in \mathcal{J} & (3) \\ \hat{t}_k + \rho_k - M_k(1 - y_k) \leq \hat{d}_k \quad \forall k | J_k \in \widetilde{\mathcal{J}} & (4) \\ \hat{t}_{k-1} - \hat{t}_k - \rho_k \geq 0 \quad \forall k \neq 1 | J_k \in \widetilde{\mathcal{J}} & (5) \\ \hat{t}_k \geq \hat{r}_k, \rho_k \geq 0 \quad \forall k | J_k \in \widetilde{\mathcal{J}} & (6) \\ y_k, z_k \in \{0, 1\} \quad \forall k | J_k \in \widetilde{\mathcal{J}} & (7) \end{cases}$$

Here, $\hat{t}_k$ is the variable equal to the (reverse) starting time of occurrence k, while ρ_k is equal to the processing time of occurrence k, and M_k is a large constant. Constraints (1) define the processing time of a job with respect to the recourse action. Constraints (2) enforce that a job may be repaired only if it is scheduled. Constraints (3) limits the number of selected occurrences to one per job. Constraints (6) and (4) respectively enforce that no job starts before its release date or finish after its deadline, while constraint (5) makes sure that no two jobs overlap. By denoting $\mathcal{Y}(U) = \{(y, z, \hat{t}, \rho) \in \mathcal{Y} \mid \sum_{k|J_k \in \mathcal{G}_j} y_k \leq 1 - U_j \quad \forall J_j \in \mathcal{J}\}$ the set of feasible second-stage solutions that are consistent with first-stage solution U, the objective function is given by:

$$\min_{U \in \{0,1\}^{|\mathcal{J}|}} \sum_{j | J_j \in \mathcal{J}} w_j U_j + f_j(1 - U_j) + \max_{\xi \in \varXi} \min_{(y,z,\hat{t},\rho) \in \mathcal{Y}(U)} R(\xi, y, z)$$

where $R(\xi, y, z)$ denotes the cost of recourse action (y, z) corresponding to scenario ξ given by: $R(\xi, y, z) = \sum_{j|J_j \in \mathcal{J}} \sum_{k|J_k \in \mathcal{G}_j} \left[(\bar{\delta}_k \xi_j - f_k) y_k - \bar{\delta}_k \xi_j z_k\right]$. Note that the outsourcing cost has been moved to the first-stage: it is assumed that outsourcing is always paid for on-time jobs unless the job is scheduled in the second stage. Also, remark that for a given $U \in \{0, 1\}^{|\mathcal{J}|}$ the recourse function $Q(U, \xi) = \min_{(y,z) \in \mathcal{Y}(U)} R(\xi, y, z)$ is not a convex function of ξ. That implies that the worst-case is in general not achieved at an extreme point of $\varXi$, so that more than $\varGamma$ jobs might see their profit degraded by a small amount.

This formulation of 2SRWNT possesses interesting features. First, the uncertainty is polyhedral and only enters the objective function. Second, the constraints linking the first and second stages $\sum_{k|J_k \in \mathcal{G}_j} y_k \leq 1 - U_j \quad \forall J_j \in \mathcal{J}$ can be expressed as $\gamma \leq \beta$, with γ and β vectors of binary decision variables associated respectively with the second and first stage. This allows us to use the methodology introduced in (Arslan and Detienne 2018) to reformulate 2SRWNT, which is based on the following successive steps: (i) replacing $\mathcal{Y}(U)$ with its convex hull expressed in terms of its extreme points (using Minkowski-Weyl theorem) ; (ii) permuting the inner max and min (using von Neumann theorem) ; (iii) linearizing the inner max using LP duality (Bertsimas and Sim 2004). Denoting by $(\mathbf{y}^e, \mathbf{z}^e), e \in E$ the extreme points of conv $\mathcal{Y}$, we obtain the following MILP model:

$$(ColGen) : \min \sum_{J_j \in \mathcal{J}} [w_j U_j + f_j(1 - U_j) + v_j] + \varGamma u - \sum_{k|J_k \in \widetilde{\mathcal{J}}} \left[f_k \sum_{e \in E} \mathbf{y}_k^e \alpha_e \right]$$

$$\text{s.t.} \sum_{e \in E} \alpha_e = 1 \tag{8}$$

$$\sum_{e \in E} \mathbf{y}_k^e \alpha_e \leq 1 - U_j \quad \forall k | J_k \in \mathcal{G}_j, \forall J_j \in \mathcal{J} \tag{9}$$

$$u + v_j \geq \sum_{k|J_k \in \mathcal{G}_j} \left[\bar{\delta}_k \sum_{e \in E} (\mathbf{y}_k^e - \mathbf{z}_k^e) \, \alpha_e \right] \quad \forall j | J_j \in \mathcal{J} \tag{10}$$

$$U_j \in \{0,1\} \quad \forall J_j \in \mathcal{J}, \alpha_e \geq 0 \quad \forall e \in E, u \geq 0, v_j \geq 0 \quad \forall J_j \in \mathcal{J}$$

Here, decision vector α represents the convex combination multipliers from the reformulation of $\text{conv}(\mathcal{Y})$ while u and v are the dual variables associated to the constraint $\xi \in \Xi$. Constraint (9) links the recourse action with the first-stage decision. Constraint (8) enforces that the recourse actions are convex combinations of the extreme points of $\text{conv}(\mathcal{Y})$. Finally, constraint (10) embeds the dualized cost associated to the worst-case scenario.

Problem 2SRWNT is trivially $\mathcal{NP}$-hard. This formulation proves as a corollary, that quite surprisingly for a min-max-min problem with integer recourse, it lies inside class $\mathcal{NP}$ and is thus $\mathcal{NP}$-complete (Arslan and Detienne 2018).

3 Numerical experiments

We develop a branch-and-price algorithm to solve model (*ColGen*), based on the C++ library BapCod (Vanderbeck 2005). The pricing problem consists in finding a solution in $\mathcal{Y}$ minimizing the reduced-cost. This variant of $1|r_j|\sum w_j U_j$ with two possible modes per job (normal or repaired) is solved with a MILP solver. Our approach is compared against the finite adaptability method of (Hanasusanto, Kuhn and Wiesemann 2015), which is the method that is the closest to our ours, although it results in a heuristic formulation. We solve this model directly using a general purpose commercial solver.

We compare both approaches on a set of 3200 randomly generated instances. Our branch-and-price algorithm solves to optimality all 20 job-instances of our test bed within one hour, and 85% of the 25 job-instances. Our method provides as by-product, for each solved instance, the number K^* of recourse solutions required to achieve optimality. When using K^* as the parameter of the finite adaptability model, it fails at solving some 10 job-instances. It solves less than 17% of the instances for which $K^* \geq 2$ and $|I| = 25$.

4 Conclusion

We have proposed a numerically effective algorithm to solve a hard robust scheduling problem exactly. It compares favorably to the finite adaptability approach, in terms of computing time and quality of solutions.

References

Aloulou, M. A. and Della Croce, F.: 2008, Complexity of single machine scheduling problems under scenario-based uncertainty, *Oper. Res. Lett.* **36**(3), 338–342.

Arslan, A. N. and Detienne, B.: 2018, Reformulation approaches for a two-stage robust knapsack problem, *International workshop on robust optimization, 2018*, Avignon.

Bertsimas, D. and Caramanis, C.: 2010, Finite Adaptability in Multistage Linear Optimization, *IEEE Transactions on Automatic Control* **55**(12), 2751–2766.

Bertsimas, D. and Sim, M.: 2004, The Price of Robustness, *Operations Research* **52**(1), 35–53.

Detienne, B.: 2014, A mixed integer linear programming approach to minimize the number of late jobs with and without machine availability constraints, *European Journal of Operational Research* **235**, 540–552.

Hanasusanto, G. A., Kuhn, D. and Wiesemann, W.: 2015, K-adaptability in two-stage robust binary programming, *Operations Research* **63**(4), 877–891.

van den Akker, M., Hoogeveen, H. and Stoef, J.: 2018, Combining two-stage stochastic programming and recoverable robustness to minimize the number of late jobs in the case of uncertain processing times, *Journal of Scheduling* **21**(6), 607–617.

Vanderbeck, F.: 2005, Bapcod - a generic branch-and-price code.
URL: *https://realopt.bordeaux.inria.fr/?page_id=2*

Yang, J. and Yu, G.: 2002, On the robust single machine scheduling problem, *Journal of Combinatorial Optimization* **6**(1), 17–33.

Scheduling of battery charging tasks with limited common power source

Lemański T.[1], Różycki R.[1], Waligóra G.[1], and Węglarz J.[1]

[1]Poznan University of Technology, Poland
e-mail: Rafal.Rozycki@cs.put.poznan.pl

Keywords: continuous resource, e-mobility, scheduling, makespan.

1. Introduction

In this work we consider a problem of scheduling battery charging tasks, assuming that the available amount of shared power is limited, and insufficient to charge all batteries in parallel. The battery charging process is very complex, and depends mainly on the type of batteries. A very popular type of battery is a Li-ion battery, used in both portable electronic equipment and electric cars. There are four main charging periods (Manwell, McGowan(1993)), of which the longest is the saturation phase. In this phase, along with the passing of time, approximately a linear decrease in power usage is observed. Therefore, it is justified to model this process using a linear function.

In this work we consider the problem of charging a set of batteries of the same type with different capacities and degrees of discharge. For simplicity, we will assume that charging each battery is limited to the saturation phase only.

Moreover, we assume that the number of charging points (a discrete resource) is unlimited, and the only limited resource is the available power, which by its nature can be allocated to charging tasks in any amounts from a certain range, i.e. it is a continuous renewable resource (Błażewicz et al. (2007)). Once started, the charging task cannot be interrupted, as it would impair the properties of the battery being charged. As one can see, such a non-classical scheduling problem is non-trivial, because one should specify the order of charging tasks that would lead to feasible schedules with the best value of the adopted criterion. In our case, this criterion is the length of the schedule.

In the next part of the work we will present the formulation of the problem, selected properties of the problem solution, and the results of a computational experiment.

2. Problem formulation

We consider a problem of scheduling n independent, non-preemptable jobs (charging tasks). Each job requires for its processing some amount of power, and consumes some amount of energy during its execution. The number of machines (a machine represents a single charging point) is unlimited and discrete resources have no influence on the final solution. Each job i, $i = 1,2,\ldots,n$, is characterized by the amount e_i of consumed energy, which represents the size of the job, the initial power usage P_{0i}, and the power usage function $p_i(t)$. This function can be, in general, arbitrary, however, in this research we assume decreasing linear power usage functions of jobs, as discussed in the Introduction. Moreover, at the completion of a job its power usage is equal to 0. Consequently, a job is sufficiently described by only two parameters, namely: e_i, P_{0i}, in our simplified situation. The model of a job is showed on Fig. 1, where s_i, c_i represent the start and completion times of job i, respectively.

Thus, the assumed job model can be given as follows:

$$p_i(t) = \begin{cases} 0 & for\ t < s_i \\ P_{0i} - \frac{P_{0i}}{d_i}(t - s_i) & for\ s_i \le t \le c_i \\ 0 & for\ t > c_i \end{cases} \qquad (1)$$

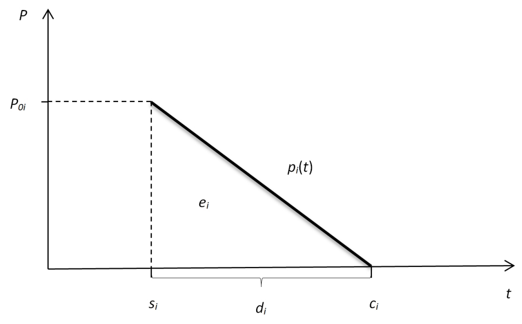


Figure 1. Graphical presentation of the job model

Notice that having defined the size e_i of a job, its initial power usage P_{0i}, and the power usage function $p_i(t)$, the processing time d_i of job i can be calculated using the following equation (2):

$$d_i = 2e_i/P_{0i} \qquad (2)$$

Thus, we have a set of jobs, from among which each is graphically represented, in the system of coordinates p and t, by a rectangular triangle of height P_{0i} and length d_i.

The objective of the problem is to minimize the schedule length. However, the total amount of power available at a time is limited. We denote by P the total amount of power available at time t. Obviously, it must hold that $P \geq \max_{i=1,\dots,n} \{P_{0i}\}$, otherwise no feasible schedule exists. Let $p(t)$ be the total power used by all jobs processed at time t, i.e.:

$$p(t) = \sum_{i \in A_t} p_i(t)$$

where A_t is the set of jobs processed at time t. Taking into account equation (1) we can write:

$$p(t) = \sum_{i \in A_t} \left(P_{0i} - \frac{P_{0i}}{d_i}(t - s_i) \right) \qquad (3)$$

and consequently, the considered problem can be mathematically formulated as:

Problem T

minimize $$C_{max} = \max_{i=1,\dots,n} \{c_i\} \qquad (4)$$

subject to $$c_i = s_i + d_i, \;\; i = 1,2,\dots,n \qquad (5)$$

$$\sum_{i \in A_t} \left(P_{0i} - \frac{P_{0i}}{d_i}(t - s_i) \right) \leq P \quad \text{for any } t \qquad (6)$$

Thus, the problem is to find a vector $\mathbf{s} = [s_1, s_2, \dots, s_n]$ of starting times of jobs that minimizes the schedule length C_{max} subject to the above constraints..

3. Properties of solutions

Let us now discuss some properties of the defined problem that can be useful for the developed solution approach.

Since due to insufficient power, in general it is not possible to start all tasks in parallel, the question arises - how to construct a schedule of the minimal length. Suppose we know a certain order of job execution, i.e. there exists a list *JL* where jobs are ordered according to their non-decreasing starting times. For each job in position q, $q = 2, 3,\ldots, n$, on *JL* the following condition holds:

$$s_{JL[q]} \geq s_{JL[q-1]}\,, q = 2, \ldots, n$$

which means that job $JL[q]$ in position q on *JL* must not start before any of its predecessors on *JL*. In this situation, the following property is useful.

Property 1. For a defined job list JL, an optimal schedule is obtained by scheduling each successive job i from the list at the earliest possible time when the required amount P_{0i} of power becomes available.

Note that the consequence of Property 1 is that in the optimal schedule, at time 0 one should run as many initial jobs from the *JL* list as possible. The moment of starting the next job from the list with the initial power consumption equal P_{0j} can be obtained by transforming (3) to the following formula:

$$s_j = \frac{P_{0j} - P + \sum_{i \in A_t} P_{0i} + \sum_{i \in A_t} \frac{P_{0i}}{d_i} s_i}{\sum_{i \in A_t} \frac{P_{0i}}{d_i}} \qquad (7)$$

In this formula, the key role is played by the information how many tasks are actually performed at the moment of starting task j, because some of the tasks may have already been finished. On the basis of Property 1, the following algorithm can be proposed, which for the known job order (represented by a particular *JL* list) determines the optimal moments of starting consecutive jobs for the considered scheduling criterion.

Algorithm A

```
Step 1. At time 0 run the maximum number of initial jobs from the JL list enabled by the
        available amount of power
Step 2. If any job remains on the JL list, repeat:
Step 2a. Find the moment to start the next job from the JL list from (7) based on information
        about jobs being currently performed.
Step 2b. If the calculated starting time is later than the fastest-ending job being completed,
        Then: remove the fastest-ending job from the set of jobs being currently performed;
        Otherwise: remove the consecutive job from the JL list, put it in the schedule at the
                calculated start time, and add to the set of jobs being currently performed.
        Return to the beginning of Step 2
Step 3. Take the finish time of the last job as the schedule length.
```

Algorithm A has the complexity of $O(n^2)$ which results from Step 2. The importance of Property 1 follows from the fact that optimal schedule can be found by using Algorithm A for each possible job permutation on the *JL* list. Of course, a full enumeration technique has an exponential complexity and is computationally inefficient. However, the solutions found in that way can be a useful reference point for assessing solutions obtained using heuristic algorithms.

Another immediate consequence of the above property is the following natural observation for the identical jobs case (i.e. $P_{0i} = P_0$ and $e_i = e$ for $i = 1,2,\ldots,n$).

Property 2. For identical jobs, Algorithm A finds an optimal schedule.

It is obvious that for identical jobs the choice of the next job to perform is of no importance – each job is represented by the same profile of power usage. As a result, the jobs can be scheduled in an arbitrary order, e.g. according to their increasing indices.

Let us denote by n_1 the number of jobs started at the moment 0 and by $s_{JL[n_1+1]}$ the moment when the next job from JL will be launched (calculated from (7)). The following property may also be relevant for the situation where, additionally, the number of charging connections is limited.

Property 3. The maximum number of jobs performed at a given moment does not exceed the number $n_1 + 1 + x$, where x is the maximum integer for which the inequality is met:

$$s_{JL[n_1+1]} + \sum_{i=1}^{x} \frac{1}{n_1 + i} < 1$$

4. Computational experiment

Simple priority rules can be used to set a suboptimal order of jobs in the *JL* list. The parameters that can be taken into account are the following: P_{0i}, d_i and the ratio P_{0i}/d_i. Of course, one can sort the jobs on the list according to non-decreasing or non-increasing values of these parameters.

In order to examine the suitability of individual priority rules, preliminary computational experiments were carried out. The assumptions of the experiments were as follows:

- number of jobs, $n = 12$; available power amount $P = 12$;
- values of P_{0i}, $i = 1, 2,\ldots, n$, were chosen randomly according to discrete uniform distribution from the set $\{1,.., P_0^{\max}\}$, and $P_0^{\max}$ took the following two values in particular groups of experiments: 3 (a large number of jobs run in parallel in the resulting schedule), 8 (a small number of jobs executed in parallel in the resulting schedule)
- values d_i, $i = 1, 2,\ldots, n$, were chosen randomly according to discrete uniform distribution from the set $\{1,.., d^{\max}\}$, and $d^{\max}$ took the following two values in particular groups of experiments: 12 (short jobs) and 50 (long jobs).

Ten test instances were generated for each case. Both: non-decreasing and non-increasing values of the chosen parameters were tested. For each sequence of jobs in *JL*, the final schedule was generated by using Algorithm A. The results obtained in this way were compared to optimal solutions obtained by the full enumeration technique (all possible permutations of *n* jobs on a *JL* list) and with random sequence of jobs on *JL*. The obtained results of the experiment show that under the adopted assumptions for the considered scheduling problem, the best rule for ordering jobs on *JL* is according to non-increasing order of d_i. The representative results for the experiment with $P_0^{\max} = 8$ and $d^{\max} = 12$ are shown in Table 1, where average (Δ_{ave}) and maximum ($\Delta_{\max}$) deviations from the optimal solutions are presented for each tested rule.

Table 1. Exemplary results of the computational experiment

Rule:	random	$\uparrow P_{0i}$	$\downarrow P_{0i}$	$\uparrow d_i$	$\downarrow d_i$	$\uparrow P_{0i}/d_i$	$\downarrow P_{0i}/d_i$
Δ_{ave}	0.29	0.23	0.34	0.32	**0.06**	0.11	0.3
$\Delta_{\max}$	0.58	0.43	0.54	0.5	**0.14**	0.22	0.5

5. Conclusions

In this work we have considered a problem of scheduling non-preemptable and independent jobs with power demands linearly decreasing with time in order to minimize the schedule length. We have shown that in an optimal schedule each job should be started as soon as the required power amount becomes available. As a result, in order to find a globally optimal schedule, all sequences of jobs have to be examined, in general. Thus, some priority rules can be applied to look for an optimal job permutation. We have performed computational tests to examine a few simple priority rules. They have shown that ordering the jobs according to their non-increasing processing times leads to the best suboptimal solutions.

References

Manwell J. F., McGowan J. G.,1993, “Lead acid battery storage model for hybrid energy systems”, Solar Energy, vol. 50, pp 399 -405, 1993.

Błażewicz J., Ecker K., Pesch E., Schmidt G., Sterna M., Węglarz J., “Handbook on Scheduling: from Theory to Applications”, Springer, Heidelberg, 2019.

Computational Experiments for the Heuristic Solutions of the Two-Stage Chain Reentrant Hybrid Flow Shop and Model Extensions

Lowell Lorenzo[1]

[1]Department of Industrial Engineering and Operations Research, University of the Philippines
Diliman 1101 Quezon City, Philippines
e-mail: Lowell.Lorenzo@up.edu.ph

Keywords: scheduling, reentrant hybrid flow shop, flexible job shop, heuristics.

1. Introduction

This is the second part of a research paper for the two-stage chain reentrant hybrid flow shop. In the first part of the research paper which was presented in Lorenzo (2017), this problem was shown to be strongly NP-hard. Lower bounds for the solution were derived and were used to develop heuristic solutions for the problem. For this paper, we now explore the performance of these heuristic solutions against the best derived lower bounds via computational experiments. Then, we develop model extensions namely the reverse two-stage chain reentrant hybrid flow shop and the two-stage flexible job shop with the corresponding heuristic solutions and computational experiments.

The outline of this paper is as follows. A short discussion of the definition of the problem and the derived lower bounds are in Sections 2 and 3. Section 4 presents the base heuristic algorithms and then the discussion on the results of the computational experiments is in Section 5. Finally, model extensions of the problem, modified heuristic algorithms, solutions and computational experiments are then presented in Sections 6-9.

2. Background and Problem Definition

Consider a simple flow shop with *m* stages. At every stage *i*,*i*=1,…,*m*, there is a single machine M_i available to process an operation of a job. Let ϕ_k be the stage visited to perform the *k*th operation of a job where $\phi_k \in \{1,2,\dots,m\}$. Then $\phi = (\phi_1, \phi_2, \dots, \phi_m) = (1,2,\dots,m)$ is the stage flow sequence for all jobs and consists of m elements or operations. In a simple flow shop, the number of operations a job undergoes is equal to the number of stages. In an *m*-stage chain reentrant flow shop, its stage flow sequence $\phi = (1,2,\dots,m,1)$ has now *(m+1)* operations due to an occurrence of a single reentrant operation. The single reentrant characteristic occurs in the *(m+1)*th operation which is performed at stage 1 and is referred to as the finishing operation.

When there are m_i identical parallel machines available in stage *i*, the resulting system is referred to as a hybrid flow shop. Let this group of m_i machines in stage *i* be referred to as work center WC_i in stage *i*.

In the two-stage chain reentrant flow shop, each job $J_j, j = 1,\dots,n$ has a stage flow sequence $\phi = (1,2,1)$. The processing time of the first operation of job J_j is a_j, its processing time in the second operation is b_j and the reentrant processing time for the finishing operation is c_j. Let the processing time vector for each job be (a_j, b_j, c_j) or simply referred to now as the processing times of J_j in the two-stage chain reentrant flow shop. Since each job is processed in every operation in the chain reentrant flow shop, then $A = (a_1,\dots,a_n), B = (b_1,\dots,b_n), C = (c_1,\dots,c_n)$ are the vectors of processing times for each operation in ϕ respectively.

In the two-stage chain reentrant hybrid flow shop, there are two work centers WC_1 and WC_2 with m_1 and m_2 identical machines in parallel at stages 1 and 2 respectively. There are *n* jobs that have to be processed and the completion time of J_j occurs when the third or finishing operation at any of the m_1 machines in WC_1 is completed. Let CRF_{m_1,m_2} be a two-stage chain reentrant hybrid flow shop where our objective is to find a schedule that minimizes the maximum completion time. Using the three-tuple convention of defining scheduling problems proposed by Graham *et al.* (1979), minimizing makespan in CRF_{m_1,m_2} can be identified by $F(m_1, m_2)|chain\ reentrant|C_{max}$ for which the optimal objective function value is $C^*_{CRF_{m_1,m_2}}$.

The CRF_{m_1,m_2} system is a general case of the two-stage chain reentrant flow shop studied by Wang *et al.* (1997). In their paper, they study the makespan minimization of $CRF_{1,1}$ and derive a Johnson based heuristic solution with complexity *O(nlogn)* and worst-case error bound of $3/2$ is derived. In Drobouchevitch and Strusevich (1999), another heuristic solution is presented for the same problem with complexity *O(nlogn)* and an improved worst-case error bound of $4/3$.

For the two-stage chain reentrant hybrid flow shop CRF_{m_1,m_2}, we construct two auxiliary two-stage flow shops. These two auxiliary two-stage flow shops are $AF1_{1,1}$ and $AF2_{1,1}$ with their respective processing times $\left(\frac{1}{m_1}a_j, \frac{1}{m_2}b_j\right)$ and $\left(\frac{1}{m_2}b_j, \frac{1}{m_1}c_j\right)$ and their corresponding makespans $C_{AF1_{1,1}}$and $C_{AF2_{1,1}}$. The AFs just introduced help in the development of lower bounds and this is the focus of the next section.

3. Lower Bounds for $C^*_{CRF_{m_1,m_2}}$

Lower bounds for $C^*_{CRF_{m_1,m_2}}$ can be developed from the constructed auxiliary two-stage flow shops described in the previous section. Since the proofs of these lower bounds were already presented in Lorenzo (2017), these will not be shown here anymore.

Lemma 1. Let $LB_1 = max\left(C_{AF1_{1,1}}, C_{AF2_{1,1}}, \frac{1}{m_1}\sum_{j=1}^{n}(a_j + c_j)\right)$. Then $LB_1 \leq C^*_{CRF_{m_1,m_2}}$.

Lemma 2. There is a permutation 1,2,…,n associated with an arbitrary schedule *S* such that for every $1 \leq i \leq n$, there is a $1 \leq k_i \leq i$ such that $\frac{1}{2}\sum_{j=1}^{i}\left(\frac{a_j}{m_1} + \frac{b_j}{m_2}\right) \leq \frac{1}{m_1}\sum_{j=1}^{k_i} a_j + \frac{1}{m_2}\sum_{j=k_i}^{i} b_j$.

Lemma 3. There is a permutation 1,2,…,n associated with an arbitrary schedule *S* such that for every $1 \leq i \leq n$, there is a $1 \leq k_i \leq i$, such that $\frac{1}{m_1+m_2}\sum_{j=1}^{i}(a_j + b_j) \leq \frac{1}{m_1}\sum_{j=1}^{k_i} a_j + \frac{1}{m_2}\sum_{j=k_i}^{i} b_j$.

Lemma 4. There is a permutation 1,2,…,n associated with an arbitrary schedule *S* such that for every $1 \leq i \leq n$, there is a $1 \leq k_i \leq i$ such that $\frac{1}{m_1+1}\sum_{j=1}^{i}\left(a_j + \frac{b_j}{m_2}\right) \leq \frac{1}{m_1}\sum_{j=1}^{k_i} a_j + \frac{1}{m_2}\sum_{j=k_i}^{i} b_j$.

Lemma 5. There is a permutation 1,2,…,n associated with an arbitrary schedule *S* such that for every $1 \leq i \leq n$, there is a $1 \leq k_i \leq i$ such that $\frac{1}{m_2+1}\sum_{j=1}^{i}\left(\frac{a_j}{m_1} + b_j\right) \leq \frac{1}{m_1}\sum_{j=1}^{k_i} a_j + \frac{1}{m_2}\sum_{j=k_i}^{i} b_j$.

Lemma 6. There is an optimal schedule S^* for CRF_{m_1,m_2} such that, for every $1 \leq k_1 \leq k_2 \leq n$, $\frac{1}{m_1}\sum_{j=1}^{k_1} a_j + \frac{1}{m_2}\sum_{j=k_1}^{k_2} b_j + \frac{1}{m_1}\sum_{j=k_2}^{n} c_j \leq C^*_{CRF_{m_1,m_2}}$.

Lemma 7. Let C_{JA_1} be the makespan derived by Johnson's Algorithm (JA) for the auxiliary two-stage flow shop problem with processing times $(a'_j + b'_j, c'_j)$. For any of the following set of values of a'_j, b'_j and c'_j,

$$a'_j = \frac{1}{2m_1}a_j,\ b'_j = \frac{1}{2m_2}b_j,\ c'_j = \frac{1}{m_1}c_j \text{ or} \tag{1}$$

$$a'_j = \frac{1}{m_1+m_2}a_j, b'_j = \frac{1}{m_1+m_2}b_j,\ c'_j = \frac{1}{m_1}c_j \text{ or} \tag{2}$$

$$a'_j = \frac{1}{m_1+1}a_j, b'_j = \frac{1}{m_2(m_1+1)}b_j,\ c'_j = \frac{1}{m_1}c_j \text{ or} \tag{3}$$

$$a'_j = \frac{1}{m_1(m_2+1)}a_j, b'_j = \frac{1}{m_2+1}b_j,\ c'_j = \frac{1}{m_1}c_j, \tag{4}$$

$C_{JA_1} \leq C^*_{CRF_{m_1,m_2}}$.

Let C_{JA_2} be the makespan derived by JA for the auxiliary two-stage flow shop problem with processing times $(b''_j + c''_j, a''_j)$, For any of the following set of values of a''_j, b''_j, and c''_j,

$$a''_j = \frac{1}{m_1}a_j,\ b''_j = \frac{1}{2m_2}b_j,\ c''_j = \frac{1}{2m_1}c_j \text{ or} \tag{5}$$

$$a''_j = \frac{1}{m_1}a_j,\ b''_j = \frac{1}{m_1+m_2}b_j,\ c''_j = \frac{1}{m_1+m_2}c_j \text{ or} \tag{6}$$

$$a''_j = \frac{1}{m_1}a_j, b''_j = \frac{1}{m_2(m_1+1)}b_j,\ c''_j = \frac{1}{m_1+1}c_j \text{ or} \tag{7}$$

$$a''_j = \frac{1}{m_1}a_j, b''_j = \frac{1}{m_2+1}b_j,\ c''_j = \frac{1}{m_1(m_2+1)}c_j, \tag{8}$$

$$C_{JA_2} \leq C^*_{CRF_{m_1,m_2}}.$$

4. Heuristic Algorithms for $F(m_1, m_2)|chain\ reentrant|C_{max}$

Since makespan minimization in CRF_{m_1,m_2} is a general case of makespan minimization in $CRF_{1,1}$, then it is NP-hard as well. This motivates the development of heuristics that will enable in the formulation of a solution to the problem. Towards this end, we also make the following observations. The makespan of CRF_{m_1,m_2} is attained in WC_1, where the first and third operations are processed. The following lemma establishes an optimal property for the scheduling of these operations in WC_1. Proofs of these lemmas and theorems have been presented in the paper of Lorenzo (2017) and will therefore not be discussed.

Lemma 8. To minimize the makespan of CRF_{m_1,m_2}, it is sufficient to consider a schedule wherein all the first operations of all jobs always precede the third operations of all jobs in any of the m_1 machines in WC_1.

In the development of a chain reentrant flow shop heuristic, you first need a sequence to specify the schedule of the jobs. Aside from the sequence you also need to assign each operation on the available machines in the corresponding stage. To do the assignment of operations to machines, we will utilize the first available machine (FAM) and last busy machine (LBM) rules. As the name implies, the FAM rule assigns a job from a sequence based on the first machine that becomes available. The LBM rule on the other hand is a mirror image of the FAM rule. Specifically, the assignment of jobs to machines for example in the second operation using the LBM rule is as follows.

Given a constant $T > 0$ and a sequence S',

Step 1. Set $t_m = T$ for $m = 1, \ldots, m_2$.

Step 2. Let J_j be the last unscheduled job in S' and $m = max_{1 \le m \le m_2}\{t_m\}$. Schedule J_j on machine m such that it finishes at time t_m.

Step 3. Set $t_m = t_m - b_j$. $S' = S' - \{j\}$. If $S' = \{\emptyset\}$, stop else go to step 2.

Three heuristics will now be presented for CRF_{m_1,m_2}.

Heuristic H1

Let S_1 be the JA schedule for the AF with processing times $\left(\frac{1}{m_1}a_j, \frac{1}{m_2}b_j\right)$ and S_2 be the JA schedule for the AF with processing times $\left(\frac{1}{m_2}b_j, \frac{1}{m_1}c_j\right)$.

Step 1. Using the sequence S_1,

a. Apply FAM on A on the stage 1 machines.

b. Apply LBM on B on the stage 2 machines and schedule these tasks as early as possible. Let T' be the largest completion time until the second operation.

c. Apply FAM on C on the stage 1 machines from T' and schedule these tasks as early as possible.

d. Calculate the makespan, C_{S_1}.

Step 2. Emulate Step 1 by using the sequence S_2 instead of S_1. Replace A with C in step 1a, replace C with A in step 1c and calculate the makespan, C_{S_2}.

Step 3. The makespan of the heuristic is $C_{H1} = min(C_{S_1}, C_{S_2})$.

Theorem 1. Let $C^* = C^*_{CRF_{m_1,m_2}}$, then $\frac{C_{H1}}{C^*} \le \frac{3}{2}\left(2 - \frac{1}{m}\right)$, where $m = max(m_1, m_2)$.

Heuristic H2

In Heuristic H2, Step 1c of Heuristic H1 is replaced by an LBM procedure namely:

Step 1c: Apply LBM on C on the stage 1 machines from $T' = \sum_{j=1}^{n}(a_j + b_j + c_j)$ and schedule these tasks as early as possible.

Theorem 2. Let $C^* = C^*_{CRF_{m_1,m_2}}$, then $\frac{C_{H2}}{C^*} \le \frac{3}{2}\left(2 - \frac{1}{m}\right)$, where $m = max(m_1, m_2)$.

Heuristic H3

Heuristics H1 and H2 use two symmetric JA sequences derived from two of the three processing times of the problem. With the lower bounds that have been derived in Lemma 7 based on the three processing times of the problem, we can modify the heuristic's input to now use two symmetric JA sequences based on all three processing times. In Lemma 7, the set of values (1), (2), (3) and (4) are symmetric to (5), (6), (7) and (8) respectively.

Consider the two AF problems with processing times $(a'_j + b'_j, c'_j)$ and its symmetric pair $(b''_j + c''_j, a''_j)$. Apply JA to these AF problems to obtain their corresponding schedule σ_k k=1,2. Replace S_1 and S_2 with σ_1 and σ_2 in steps 1 and 2 respectively in H1.

We can use any of the following four pairs of three processing time JA schedules in H3. We distinguish them by the following:

1. H3.1 when the pair of JA schedules is based on (1) and (5).
2. H3.2 when the pair of JA schedules is based on (2) and (6).
3. H3.3 when the pair of JA schedules is based on (3) and (7).
4. H3.4 when the pair of JA schedules is based on (4) and (8).

5. Computational Experiments for $F(m_1, m_2)|chain\ reentrant|C_{max}$

A computational experiment using various parameter values was conducted to assess the performance of the heuristic algorithms H1 and H3. H1 uses a two processing time JA schedule input while H3 uses a three processing time JA schedule. For each combination of values below, we generated 10 problem instances using: number of jobs, *n*, number of machines in stage *i*, m_i and the processing times, a_j, b_j and c_j which were randomly generated from U(l,u), which is a discrete uniform distribution in [l,u]. The values considered for these parameters are shown in Table 1.

Table 1 shows the average deviation of the heuristic solution from the lower bound, LB_1 which was established in Lemma 1. Since LB_1 dominates all the lower bounds established in Lemma 7, it is used in the calculation of the average deviation. The average deviation *AD* is given by the formula, $AD = (C((H_i) - LB_1) * 100\%/LB_1$ where $C(H_i)$ is the makespan obtained in heuristic H_i.

The following can be observed from the computational experiment.

1. The average deviation increases as m_1 increases. This can be explained by the behavior of the lower bound LB_1. Recall that $LB_1 = max\left(C_{AF1_{1,1}}, C_{AF2_{1,1}}, \frac{1}{m_1}\sum_{j=1}^{n}(a_j + c_j)\right)$ where $C_{AF1_{1,1}}$ is the makespan from a JA schedule with processing times $\left(\frac{1}{m_1}a_j, \frac{1}{m_2}b_j\right)$ and $C_{AF2_{1,1}}$ is the makespan from a JA schedule with processing times $\left(\frac{1}{m_2}b_j, \frac{1}{m_1}c_j\right)$. When m_1 is small, the lower bound is dominated by $\frac{1}{m_1}\sum_{j=1}^{n}(a_j + c_j)$. As the value of m_1 increases, the lower bound now gets dominated by either $C_{AF1_{1,1}}$ or $C_{AF2_{1,1}}$.

2. The H3 heuristic generated a better solution than H1 based on the observed lower average deviation. In all the problem cases, each variant of H3 yielded a lower average deviation versus H1. Among the variants of the H3 heuristic, H3.3 generated the smallest overall average deviation of 0.81%. However, H3.3 did not generate the smallest average deviation per problem scenario.

3. From Table 1, we can see that as the number of jobs increases, the average deviation decreases. When *n* = 40, the average deviation is 1.82% and when *n* = 80, it decreases to 0.89%. This can be explained by the higher utilization of the machines in the work centers when there are more jobs.

4. As the variability of processing time increases, the average deviation in H3 also increases. When the processing times are uniformly distributed in the interval [1,20], the average deviation is 0.78% and when the processing times are uniformly distributed in the interval [10,50] it increases to 1.24%. This cannot be extended for H1 as seen in Table 1.

Table 1: % Average Deviation from Lower Bound

n = 40	$a_j, b_j, c_j \sim$ U(10,50)					$a_j, b_j, c_j \sim$ U(1,20)				
	$H1$	$H3.1$	$H3.2$	$H3.3$	$H3.4$	$H1$	$H3.1$	$H3.2$	$H3.3$	$H3.4$
m_1=2, m_2=6	0.49	0.09	0.15	0.16	0.18	0.28	0.08	0.08	0.08	0.08
m_1=4, m_2=4	1.59	0.69	0.69	0.48	0.70	1.46	0.25	0.25	0.20	0.20
m_1=6, m_2=2	7.94	4.71	4.10	3.98	4.84	8.66	3.38	2.24	1.96	4.49
Average	3.34	1.83	1.64	1.54	1.91	3.47	1.24	0.86	0.75	1.59
n = 80										
m_1=2, m_2=6	0.18	0.06	0.07	0.06	0.07	0.27	0.02	0.02	0.02	0.02
m_1=4, m_2=4	1.01	0.19	0.19	0.22	0.18	1.03	0.10	0.10	0.10	0.10
m_1=6, m_2=2	4.04	1.99	1.85	1.89	2.32	5.72	1.36	0.73	0.61	2.29
Average	1.75	0.75	0.70	0.72	0.86	2.34	0.49	0.28	0.24	0.80

6. Model Extensions

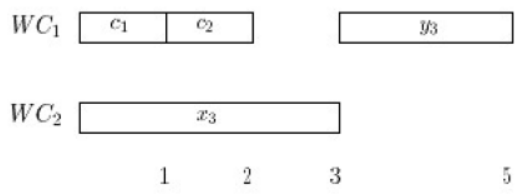
Consider the two-stage flexible job shop problem $FJ(m_1, m_2)|o = 3|C_{max}$ where $o_1, \dots, o_n = 3$ is the number of operations of job J_j. In this problem, there are two work centers WC_1 and WC_2 with m_1 and m_2 identical machines in parallel respectively. In the two-stage flexible job shop, each job J_j can either start in WC_1 or WC_2 and must complete three operations wherein no two consecutive operations are done on the same work center. There are thus two types of jobs, type 1 jobs which start in WC_1 and type 2 jobs which start in WC_2. There are n_1 type 1 jobs and n_2 type 2 jobs. Type 1 jobs are therefore processed in a CRF_{m_1,m_2} system with processing times (a_j, b_j, c_j) for operations one, two and three respectively. Type 2 jobs have processing times (x_j, y_j, z_j) for operations one, two and three respectively wherein the *x* and *z* tasks are performed in WC_2 and the y task is processed in WC_1. Let $X = (x_1, \dots, x_n), Y = (y_1, \dots, y_n), Z = (z_1, \dots, z_n)$ be the vectors of processing times for each operation of the type 2 jobs. Let us refer to this two-stage flexible job shop as FJ_{m_1,m_2} where our objective is to minimize makespan which is attained at $C^*_{FJ_{m_1,m_2}}$. Whereas the makespan of CRF_{m_1,m_2} is always attained in WC_1, the makespan of FJ_{m_1,m_2} can either occur in WC_1 or WC_2. Lemma 8 established an optimal property of the arrangement of the first operations and third operations of the type 1 jobs in WC_1 for CRF_{m_1,m_2}. In FJ_{m_1,m_2}, the work centers WC_1 (WC_2) aside from processing the first operations and third operations of the type 1 (2) jobs also process the second operations of the type 2 (1) jobs. The following lemma establishes an order between the three operations of the jobs in an optimal solution.

Lemma 9. To minimize the makespan of FJ_{m_1,m_2}, it is sufficient to consider an optimal schedule wherein all the *A (X)* tasks precede the *B (Y)* and the *C (Z)* tasks.

Proof: The proof is similar to Lemma 8.■

This lemma however does not establish the optimality of the precedence of a *Y (B)* task before a *C (Z)* task. Without loss of generality, consider WC_1 and WC_2 where each work center consists of one machine. Consider also two identical type 1 jobs with processing times (0,0,1) and one type 2 job with processing times (3,2,0). The optimal solution is shown in Figure 1. The optimal solution shows the *C* tasks precede the *Y* task. Making the *Y* task precede the *C* tasks will result to an increase of the optimal makespan. Thus, we cannot establish the precedence of a second operation before a third operation in an optimal solution.

Figure 1: Optimal Solution



Although the previous lemma does not establish the optimality of the precedence of a second operation before a third operation, we can utilize this property in the development of a heuristic algorithm for this problem. But first, we develop some lower bounds for $C^*_{FJ_{m_1,m_2}}$.

7. Lower Bounds For $C^*_{FJ_{m_1,m_2}}$

Type 1 jobs are symmetric to type 2 jobs because their stage flow sequence for each operation and the number of machines at each stage are interchanged. Since type 1 jobs are processed in a CRF_{m_1,m_2} system, a symmetric system must likewise be defined for the type 2 jobs. Let RRF_{m_2,m_1} be the reverse two-stage chain reentrant hybrid flow shop which is applicable for the type 2 jobs. Since the type 2 jobs are processed first in WC_2, the RRF_{m_2,m_1} system therefore has a stage flow sequence $\phi = (2,1,2)$.

Consider RRF_{m_2,m_1} where the processing time vector for a job J_j is (x_j, y_j, z_j). Two auxiliary two-stage flow shops $AF3_{1,1}$ and $AF4_{1,1}$ can be constructed with their respective processing times $\left(\frac{1}{m_2}x_j, \frac{1}{m_1}y_j\right)$ and $\left(\frac{1}{m_1}y_j, \frac{1}{m_2}z_j\right)$. By applying JA to these auxiliary flow shops, their corresponding makespans are $C_{AF3_{1,1}}$ and $C_{AF4_{1,1}}$. By appropriately substituting $C_{AF3_{1,1}}$ and $C_{AF4_{1,1}}$ in Lemma 1, we have $max(C_{AF3_{1,1}}, C_{AF4_{1,1}}) \leq C^*_{RRF_{m_2,m_1}}$.(32)

Lemma 10. Let $LB_2 = max\left(\frac{1}{m_1}\sum_{j=1}^{n}(a_j + y_j + c_j),\ \frac{1}{m_2}\sum_{j=1}^{n}(x_j + b_j + z_j)\right)$. Then $LB_2 \leq C^*_{FJ_{m_1,m_2}}$.

Proof: If the makespan is attained in WC_1, then $\frac{1}{m_1}\sum_{j=1}^{n}(a_j + y_j + c_j) \leq C^*_{FJ_{m_1,m_2}}$ and if the makespan is attained in WC_2, then $\frac{1}{m_2}\sum_{j=1}^{n}(x_j + b_j + z_j) \leq C^*_{FJ_{m_1,m_2}}$.■

Lemma 11. $C^*_{RRF_{m_2,m_1}} \leq C^*_{FJ_{m_1,m_2}}$ and $C^*_{CRF_{m_1,m_2}} \leq C^*_{FJ_{m_1,m_2}}$.

Proof: The proof is obvious.■

Consider again RRF_{m_2,m_1} and the auxiliary two-stage flow shop now with processing times $(x'_j + y'_j, z'_j)$. Let C_{JA_3} be the makespan derived by JA for this AF. By appropriately substituting the following set of values of x'_j, y'_j and z'_j in Lemma 7, we have $C_{JA_3} \leq C^*_{RRF_{m_2,m_1}} \leq C^*_{FJ_{m_1,m_2}}$.

$$x'_j = \frac{1}{2m_2}x_j,\ y'_j = \frac{1}{2m_1}y_j,\ z'_j = \frac{1}{m_2}z_j \text{ or} \quad (9)$$

$$x'_j = \frac{1}{m_1+m_2}x_j, y'_j = \frac{1}{m_1+m_2}y_j,\ z'_j = \frac{1}{m_2}z_j \text{ or} \quad (10)$$

$$x'_j = \frac{1}{m_2+1}x_j, y'_j = \frac{1}{m_1(m_2+1)}y_j,\ z'_j = \frac{1}{m_2}z_j \text{ or} \quad (11)$$

$$x'_j = \frac{1}{m_2(m_1+1)}x_j, y'_j = \frac{1}{m_1+1}y_j,\ z'_j = \frac{1}{m_2}z_j, \quad (12)$$

By symmetry, let C_{JA_4} be the makespan derived by JA for the AF with processing times $(y''_j + z''_j, x''_j)$.By appropriately substituting the following set of values of x''_j, y''_j and z''_j in Lemma 7, we have, $C_{JA_4} \leq C^*_{RRF_{m_2,m_1}} \leq C^*_{FJ_{m_1,m_2}}$.

$$x''_j = \frac{1}{m_2}x_j,\ y''_j = \frac{1}{2m_1}y_j,\ z''_j = \frac{1}{2m_2}z_j \text{ or} \quad (13)$$

$$x''_j = \frac{1}{m_2}x_j,\ y''_j = \frac{1}{m_1+m_2}y_j,\ z''_j = \frac{1}{m_1+m_2}z_j \text{ or} \quad (14)$$

$$x''_j = \frac{1}{m_2}x_j, y''_j = \frac{1}{m_1(m_2+1)}y_j,\ z''_j = \frac{1}{m_2+1}z_j \text{ or} \quad (15)$$

$$x''_j = \frac{1}{m_2}x_j, y''_j = \frac{1}{m_1+1}y_j,\ z''_j = \frac{1}{m_2(m_1+1)}z_j, \quad (16)$$

8. Heuristic Algorithms for $FJ(m_1, m_2)|o = 3|C_{max}$

Since CRF_{m_1,m_2} and RRF_{m_2,m_1} are special cases of the more general problem FJ_{m_1,m_2}, the heuristics H1 and H3 can be modified accordingly to solve it. Since it was observed that as the number of jobs increased in CRF_{m_1,m_2}, the performance of the heuristics H1 and H3 also improved. With this insight, a modification of these heuristics can be constructed to solve FJ_{m_1,m_2}.

Lemma 9 does not establish the precedence of the second operation before the third operation in an optimal solution. However, in order to establish some structure in the heuristic, we will make the second operation always precede the third operation. The following heuristic illustrates the use of this property together with the optimal property stated in Lemma 9. The modified versions of H1 and H3 are now presented as H1a and H3a respectively.

Heuristic H1a

Let $S_{1.1}$ be the JA schedule for the AF with processing times $\left(\frac{1}{m_1}a_j, \frac{1}{m_2}b_j\right)$ and $S_{1.2}$ be the JA schedule for the AF with processing times $\left(\frac{1}{m_2}x_j, \frac{1}{m_1}y_j\right)$. Let $S_{2.1}$ be the JA schedule for the AF with processing times $\left(\frac{1}{m_2}b_j, \frac{1}{m_1}c_j\right)$ and $S_{2.2}$ be the JA schedule for the AF with processing times $\left(\frac{1}{m_1}y_j, \frac{1}{m_2}z_j\right)$.

Step 1. Using the sequence $S_{1.1}$ for A, B and C and $S_{1.2}$ for X, Y and Z:

a. Apply FAM on A and X in WC_1 and WC_2 respectively.

b. Apply LBM on B and Y in WC_2 and WC_1 respectively and schedule them as early as possible.

c. Apply FAM on C and Z in WC_1 and WC_2 respectively.

d. Calculate the makespan CS_f.

Step 2. Emulate Step 1 by using the sequences $S_{2.1}$ and $S_{2.2}$ instead of $S_{1.1}$ and $S_{1.2}$ respectively. Replace A with C and replace X with Z in step 1a. Replace C with A and Z with A in step 1c and calculate the makespan, CS_b.

Step 3. The makespan of the heuristic $C_{H1a} = \min(CS_f, CS_b)$.

Heuristic H3a

Similar to H3, consider the two AF problems with processing times $(x_j' + y_j', z_j')$ and its symmetric pair $(y_j'' + z_j'', x_j'')$. Apply JA to these AF problems to obtain their corresponding schedule *k*, *k* = 1,2. Recall that in H3, σ_1 and σ_2 replaced S_1 and S_2 respectively in H1. Similarly, replace $S_{1.1}$ and $S_{1.2}$ with σ_1 and $\sigma_2$respectively and $S_{2.1}$ and $S_{2.2}$ with τ_1 and τ_2 respectively in H1a.

We can use any of the following four pairs of three processing time JA schedules in H3a. We distinguish them by the following:

1. H3.1a when the two pairs of JA schedules are based on (1) and (5) for A, B and C and (9) and (13) for X, Y and Z.
2. H3.2a when the two pairs of JA schedules are based on (2) and (6) for A, B and C and (10) and (14) for X, Y and Z.
3. H3.3a when the two pairs of JA schedules are based on (3) and (7) for A, B and C and (11) and (15) for X, Y and Z.
4. H3.4a when the two pairs of JA schedules are based on (4) and (8) for A, B and C and (12) and (16) for X, Y and Z.

9. Computational Experiments for $FJ(m_1, m_2)|o = 3|C_{max}$

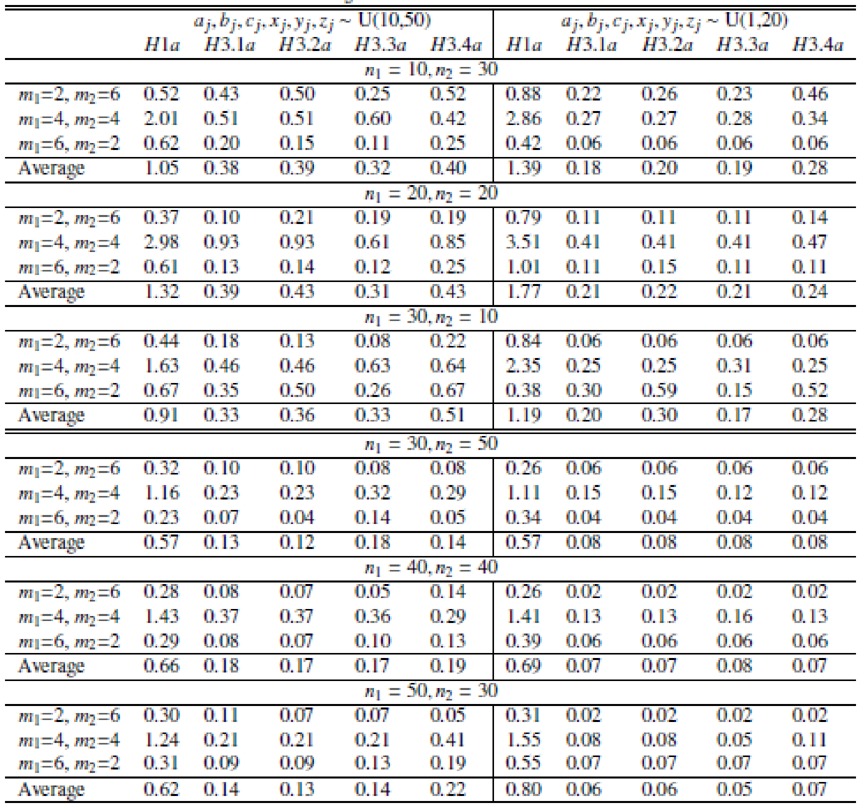
Similarly as in Section 5, a computational experiment using various parameter values was conducted to assess the performance of the heuristic algorithms H1a and H3a. H1a uses a two processing time JA schedule input while H3a uses a three processing time JA schedule. For each combination of values below, we generated 10 problem instances using: number of jobs, n_1 and n_2, number of machines in stage i,m_i and the processing times, $a_j, b_j, c_j,\ x_j, y_j$ and z_j which were randomly generated from U(l,u), which is a discrete uniform distribution in [l,u]. The values of these parameters are shown in Table 2.

Table 2 shows the average deviation of the heuristic solution from the lower bound, LB_2 which was established in Lemma 10. From the computational experiments, we find that LB_2 dominates all the lower bounds established for FJ_{m_1,m_2} and is thus used in the calculation of the average deviation AD.

Since the utilization of the machines in the two work centers increases when processing both types 1 and 2 jobs, it is expected that the average deviation of the heuristic solutions will improve over the values seen in Table 1. This is validated by the results shown in Table 2. The following additional observations can be made from the computational experiment.

1. The H3a heuristic generates a better solution than H1a based on the observed lower average deviation. The average deviation for H3a was 0.21% while it was 0.96% for H1a. In all the problem cases, each variant of H3a yielded a lower average deviation versus H1a. Among the variants of the H3a heuristic, H3.3a generated the smallest overall average deviation of 0.18%. However, H3.3a did not generate the smallest average deviation per problem scenario.
2. As the number of jobs increases, the average deviation decreases. When *n* = 40, the average deviation of H1a is 1.27% and it decreases to 0.65% when *n* = 80. For H3a, the average deviation is 0.30% when *n* = 40 and it decreases to 0.11% when *n* = 80. This can be explained by the higher utilization of the machines in the work centers when there are more jobs.
3. As the variability of processing times increases, the average deviations in H3a also increase. This is observed when the average deviation of 0.15% for the interval [1,20] increases to 0.27% for the interval [10,50].

Table 2: % Average Deviation from Lower Bound



	$a_j, b_j, c_j, x_j, y_j, z_j$ ~ U(10,50)					$a_j, b_j, c_j, x_j, y_j, z_j$ ~ U(1,20)				
	$H1a$	$H3.1a$	$H3.2a$	$H3.3a$	$H3.4a$	$H1a$	$H3.1a$	$H3.2a$	$H3.3a$	$H3.4a$
$n_1 = 10, n_2 = 30$										
m_1=2, m_2=6	0.52	0.43	0.50	0.25	0.52	0.88	0.22	0.26	0.23	0.46
m_1=4, m_2=4	2.01	0.51	0.51	0.60	0.42	2.86	0.27	0.27	0.28	0.34
m_1=6, m_2=2	0.62	0.20	0.15	0.11	0.25	0.42	0.06	0.06	0.06	0.06
Average	1.05	0.38	0.39	0.32	0.40	1.39	0.18	0.20	0.19	0.28
$n_1 = 20, n_2 = 20$										
m_1=2, m_2=6	0.37	0.10	0.21	0.19	0.19	0.79	0.11	0.11	0.11	0.14
m_1=4, m_2=4	2.98	0.93	0.93	0.61	0.85	3.51	0.41	0.41	0.41	0.47
m_1=6, m_2=2	0.61	0.13	0.14	0.12	0.25	1.01	0.11	0.15	0.11	0.11
Average	1.32	0.39	0.43	0.31	0.43	1.77	0.21	0.22	0.21	0.24
$n_1 = 30, n_2 = 10$										
m_1=2, m_2=6	0.44	0.18	0.13	0.08	0.22	0.84	0.06	0.06	0.06	0.06
m_1=4, m_2=4	1.63	0.46	0.46	0.63	0.64	2.35	0.25	0.25	0.31	0.25
m_1=6, m_2=2	0.67	0.35	0.50	0.26	0.67	0.38	0.30	0.59	0.15	0.52
Average	0.91	0.33	0.36	0.33	0.51	1.19	0.20	0.30	0.17	0.28
$n_1 = 30, n_2 = 50$										
m_1=2, m_2=6	0.32	0.10	0.10	0.08	0.08	0.26	0.06	0.06	0.06	0.06
m_1=4, m_2=4	1.16	0.23	0.23	0.32	0.29	1.11	0.15	0.15	0.12	0.12
m_1=6, m_2=2	0.23	0.07	0.04	0.14	0.05	0.34	0.04	0.04	0.04	0.04
Average	0.57	0.13	0.12	0.18	0.14	0.57	0.08	0.08	0.08	0.08
$n_1 = 40, n_2 = 40$										
m_1=2, m_2=6	0.28	0.08	0.07	0.05	0.14	0.26	0.02	0.02	0.02	0.02
m_1=4, m_2=4	1.43	0.37	0.37	0.36	0.29	1.41	0.13	0.13	0.16	0.13
m_1=6, m_2=2	0.29	0.08	0.07	0.10	0.13	0.39	0.06	0.06	0.06	0.06
Average	0.66	0.18	0.17	0.17	0.19	0.69	0.07	0.07	0.08	0.07
$n_1 = 50, n_2 = 30$										
m_1=2, m_2=6	0.30	0.11	0.07	0.07	0.05	0.31	0.02	0.02	0.02	0.02
m_1=4, m_2=4	1.24	0.21	0.21	0.21	0.41	1.55	0.08	0.08	0.05	0.11
m_1=6, m_2=2	0.31	0.09	0.09	0.13	0.19	0.55	0.07	0.07	0.07	0.07
Average	0.62	0.14	0.13	0.14	0.22	0.80	0.06	0.06	0.05	0.07

10. Conclusion

This paper evaluated the performance of the heuristic solutions developed by Lorenzo (2017) for the $F(m_1, m_2)|chain\ reentrant|C_{max}$ problem against the best established lower bound through computational experiments. The results showed that the heuristic solutions gave very good approximations to the optimal solution. The CRF_{m_1,m_2}model is then extended to include features of the RRF_{m_2,m_1} model which leads to the formulation of the FJ_{m_1,m_2} model. The previous heuristic solutions and lower bounds developed for the CRF_{m_1,m_2} model are then modified for the $FJ(m_1, m_2)|o = 3|C_{max}$ problem and the computational experiments again yield very good approximations to the optimal solution. Future research shall be directed towards the improvement of the heuristic solutions for CRF_{m_1,m_2} which may yield better worst-case error bounds.

Acknowledgements

The author would like to thank the Paul Dumol PCA from the UP Diliman College of Engineering and the UP Diliman ERDT/FRDG program for providing funding for this research paper and attendance to the 17th International Conference on Project Management and Scheduling respectively.

References

Aldakhilallah K A., Ramesh R., 2001, "Cyclic scheduling heuristics for a re-entrant job shop manufacturing environment", International Journal of Production Research, Vol.12, pp. 2635-2657.

Bispo C., Tayur S., 2001, "Managing simple reentrant flow lines: theoretical foundation and experimental results", Vol. 33, IIE Transactions, pp. 609-623.

Buten R E., Shen V Y., 1973, "A scheduling model for computer systems with two classes of processors", Proc. 1973 Sagamore Computer Conference on Parallel Processing, pp. 130-138.

Drobouchevitch I G., Strusevich VA., 1999, "A heuristic algorithm for two-machine re-entrant shop scheduling", Annals of Operations Research, Vol. 86, pp. 417-439.

Graham R L., Lawler, E L. and Lenstra J K., 1979, "Optimization and approximation in deterministic sequencing and scheduling: a survey", Ann. Discrete Math, Vol. 5, pp. 287-326.

Guinet A G., Solomon M M.. 1996, "Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time", Vol. 34, pp. 1643-1654.

Gupta J N D., Tunc E A., 1994, "Scheduling a two-stage hybrid flowshop with separable setup and removal times", Vol. 77, pp. 415-428.

Hall N G., Lee T E., Posner M E., 2002, "The complexity of cyclic shop scheduling problems", Journal of Scheduling, Vol. 5, pp. 307-327.

Hoogeveen J A., Lenstra J K., Veltman B., 1996, "Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard", European Journal of Operational Research, Vol. 89, pp. 172-175.

Hwang H, Sun J U., 1997, "Production sequencing problem with reentrant work flows and sequence dependent setup times" Vol. 33, pp. 773-776.

Johnson S M., 1954, "Optimal two and three-stage production schedules with setup times included", Naval Research Logistics Quarterly, Vol. 1, pp. 61-68.

Koulamas C., Kyparisis G J., 2000, "Asymptotically optimal linear time algorithms for two-stage and three-stage flexible flow shops", Naval Research Logistics, Vol. 47, pp. 259-268.

Kubiak W., Lou S X., Wang Y., 1996, "Mean flow time minimization in reentrant job shops with a hub", Operations Research, Vol. 44, pp. 764-776.

Lee C Y., Vairaktarakis G L., 1994, "Minimizing makespan in hybrid flowshops", Operations Research Letters, Vol. 16, pp. 149-158.

Lev V., Adiri I., 1984, "V-Shop scheduling", European Journal of Operational Research, Vol. 18, pp. 51-56.

Lorenzo, L., 2017, "Minimizing Makespan in a Class of Two-Stage Chain Reentrant Hybrid Flow Shops", Proc. of the World Congress on Engineering (WCE) 2017, Vol. I, pp. 50-56.

Lu S, Kumar P. R., 1991, "Distributed scheduling based on due dates and buffer priorities", IEEE Trans. on Automatic Control, Vol. 12, pp. 1406-1416.

Wang M Y., Sethi S P. and van de Velde S L., 1997, "Minimizing makespan in a class of reentrant shops", Operations Research Vol. 45, pp. 702-712.

Minimizing Flow Time on a Single Machine with Job Families and Setup Times.

Malapert Arnaud[1] and Nattaf Margaux[2]

[1] Université Côte d'Azur, CNRS, I3S, France
arnaud.malapert@univ-cotedazur.fr
[2] Univ. Grenoble Alpes, CNRS, Grenoble INP**, G-SCOP, 38000 Grenoble, France
margaux.nattaf@grenoble-inp.fr

Keywords: single machine scheduling ; flow time ; job families ; setup times

Context and problem description In (Mason and Anderson 1991), the authors consider a one machine scheduling problem with the goal of minimizing the average weighted flow time. In this problem, jobs are grouped into families with the property that a setup time is required only when processing switches from jobs of one family to jobs of another family. Furthermore, setup times are additive: each setup for a new family of job consists of the setdown from the previous family followed by a setup of the new family. These additive setups are considered as sequence-independent. Note that a setup is required at time 0 before the very first job is processed. For this problem, they define several properties of an optimal solution. These properties are then used in a branch-and-bound procedure.

If jobs belonging to the same family have equal processing times and unit weights, the application of their results directly leads to a polynomial time algorithm. However, if no setup time is required at time 0, i.e. before the very first job, their results cannot be applied directly. The first goal of this paper is to prove this affirmation. The second objective is to show how their results can be adapted in this special case. These results are then used to define a polynomial time algorithm to compute the optimal flow time for this special case. More generally, the objective is to use this algorithm for solving a parallel machine scheduling problem with time constraints and machine qualifications described in (Malapert and Nattaf 2019).

The problem considered in this paper is now formally described. Consider the problem of scheduling a set $\mathcal{N}$ of n jobs on one machine. Each job belongs to a family $f \in \mathcal{F}$ and the family associated with a job j is denoted by f_j. Each family is then associated with a number of jobs to process n_f, a processing time p_f, and a sequence independent setup time s_f. Thus, switching the production from family f to $f' \neq f$ will require a setup time $s_{f'}$. Note that no setup time is needed between two jobs of the same family. The important difference to the original problem is that the setup times are not additive anymore because no setup time is required at time 0 before the very first job is processed. The goal is to minimize the flow time which is the sum of the finishing time of all jobs.

Optimal solution properties The goal of this section is to describe several properties and characteristics of an optimal solution. The properties will then be used to design our polynomial-time algorithm.

To represent a solution, let S be a sequence representing an ordered set of n jobs. Then, S can be seen as a series of blocks, where a block is a maximal consecutive subsequence of jobs in S from the same family. Let B_i be the i-th block of the sequence $S = \{B_1, B_2, \dots, B_r\}$.

** Institute of Engineering Univ. Grenoble Alpes

Successive blocks contain jobs from different families. Therefore, there will be a setup time before each block (except the first one). The idea is to adapt SPT Smith's rule (Smith 1956) for blocks instead of individual jobs. To this end, blocks are considered as individual jobs with processing time P_i and weight W_i with: $P_i = s_{f_i} + |B_i| \cdot p_{f_i}$ and $W_i = |B_i|$ where f_i denotes the family of jobs in B_i (which is the same for all jobs in B_i).

The following theorem states that there exists an optimal solution S containing exactly $|\mathcal{F}|$ blocks and that each block B_i contains all jobs of the family f_i.

Theorem 1. *Let $\mathcal{I}$ be an instance of the problem. There exists an optimal solution $S^* = \{B_1, \ldots, B_{|\mathcal{F}|}\}$ such that $|B_i| = n_{f_i}$ where f_i is the family of jobs in B_i.*

Proof. Consider an optimal solution $S = \{B_1, \ldots, B_u, \ldots, B_v, \ldots, B_r\}$ with two blocks B_u and B_v ($u < v$), containing jobs of the same family $f_u = f_v = f$. We show that moving the first job of B_v at the end of block B_u can only improve the solution.

Let us define P and W as: $P = \sum_{i=u+1}^{v-1} P_i + s_f$ and $W = \sum_{i=u+1}^{v-1} |B_i|$. Note that P (resp. W) is the total time of all jobs, including setups, (resp. the number of jobs) performed between the last job of B_u and the first job of B_v. Let us call π this partial sequence.

Let S' be the sequence formed by moving the first job of B_v, say job j_v, at the end of block B_u , that is swapping the position of π and j_v (see Fig. 1).

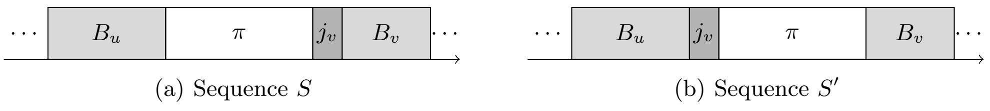


Fig. 1. Construction of sequence S' from sequence S.

First, the difference on the flow time is computed in the cases where $|B_v| > 1$ and where $|B_v| = 1$.
If $|B_v| > 1$ (that is there are still jobs left in B_v after the removal of job j_v), then the flow times of jobs sequenced before B_u or after B_v do not change. All the jobs in π have their flow times increased by p_f and j_v has its flow time decreased by P, giving a difference in the total flow time of:

$$FT_{S'} - FT_S = W \cdot p_f - P$$

If $|B_v| = 1$, then B_v is left with no jobs after moving job j_v, and so the setup time associated with B_v is deleted from the sequence. This reduces the flow times of all jobs sequenced after π by the amount of s_f, giving an additional reduction in the total net flow time:

$$FT_{S'} - FT_S = W \cdot p_f - P - \sum_{i=v+1}^{r} |B_i| \cdot s_f$$

Hence, if one can prove that $P/W \geq p_f$, $FT_S - FT_{S'} < 0$ and the flow time can only be improved in S'.

Lemma 1. $\frac{P}{W} \geq p_f$

Proof. Consider the sequence S and suppose that $p_f > P/W$. Let S'' be the sequence formed by moving the last job of B_u at the beginning of block B_v. If $|B_u| > 1$, then the difference of flow time is: $FT_{S''} - FT_S = P - W \cdot p_f$. Since, by definition, $p_f > P/W$, we have that $FT_{S''} - FT_S < 0$ which contradict the fact that S is optimal.

If $|B_u| = 1$, $FT_{S''} - FT_S = \begin{cases} W \cdot p_f - P - \sum_{i=v+1}^{r} |B_i| \cdot s_f & \text{if } u \neq 1 \\ W \cdot p_f - P - \sum_{i=v+1}^{r} |B_i| \cdot s_f - \sum_{i=u+1}^{r} |B_i| \cdot s_{f_{u+1}} & \text{if } u = 1 \end{cases}$
And then, S'' is a better solution than S which is a contradiction. Hence, we have $P/W \geq p_f$. $\square$

Therefore, by Lemma 1, $FT_{S'} - FT_S \leq 0$. Hence, swapping the position of job j_v with π leads to a solution S' at least as good as S. Repeated applications of this operation yield the result.

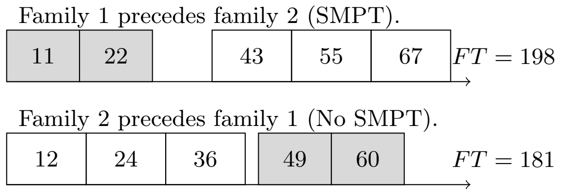
The mean processing time of a block B_i can be defined as $MPT(B_i) = P_i/W_i$. Theorem 2 generalized the SPT rule for blocks.

Theorem 2. *In an optimal sequence of the problem, the blocks 2 to $|\mathcal{F}|$ are ordered by SMPT (Shortest Mean Processing Time). That is, if $1 < i < j$ then $MPT(B_i) \leq MPT(B_j)$.*

This theorem is not exactly the same as the one in (Mason and Anderson 1991). Indeed, their theorem allows the ordering of all blocks according the SMPT rule while Theorem 2 orders only blocks 2 to $|\mathcal{F}|$. Figure 2 gives a counterexample showing that the SMPT rule is not optimal when there is no setup at time 0.

f	$(\, n_f \times$	$p_f +$	$s_f\,) \div\, n_f =$	MPT_f
1	2	11	2	12
2	3	12	9	15

(a) Problem instance with 2 families and 5 jobs.


Family 1 precedes family 2 (SMPT).
11
22
43
55
67
$FT = 198$
Family 2 precedes family 1 (No SMPT).
12
24
36
49
60
$FT = 181$

(b) Scheduling jobs of a family before those of the other (Numbers are completion times of the jobs).

Fig. 2. The SMPT rule may lead to suboptimal solutions when no setup time is required at time 0.

Proof (Th. 2). Let $S = \{B_1, B_2, \ldots, B_u, B_{u+1}, \ldots, B_{|\mathcal{F}|}\}$ be a sequence in which blocks B_u and B_{u+1}, $1 < u \leq u+1 \leq |\mathcal{F}|$, are not in $SMPT$ order; that is, $MPT(B_u) > MPT(B_{u+1})$. Consider the change in total flow time if the processing order of B_u and B_{u+1} is reversed. Clearly the flow times of any jobs originally scheduled before run B_u will not be changed. Also, the total time to complete blocks B_u and B_{u+1} will not be changed, so the flow times of any jobs scheduled after B_{u+1} will not be altered. Hence, only the flow times of the jobs in blocks B_u and B_{u+1} needs to be considered. Each job in B_{u+1} has its completion time reduced by P_u and each job in B_u has its completion time increased by P_{u+1}. Thus, the change in weighted flow time is given by:

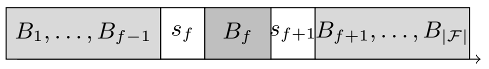
$$\Delta F_w = W_u \cdot P_{u+1} - W_{u+1} \cdot P_u$$

But, since $MPT(B_u) > MPT(B_{u+1})$, $P_u \cdot W_{u+1} > P_{u+1} \cdot W_u$, and so $\Delta F_w < 0$.

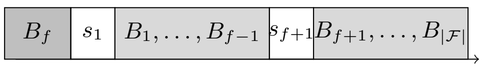
The following section explains how these results are used to define a polynomial time algorithm for solving the problem.

Polynomial-time algorithm Theorem 1 states that there exists an optimal solution S containing exactly $|\mathcal{F}|$ blocks and that each block B_i contains all jobs of family f_i. Theorem 2 states that the blocks B_2 to $B_{|\mathcal{F}|}$ are ordered by $SMPT$. Finally, one only needs to determine which family is processed in the very first block.

Algorithm 1 take as input the jobs grouped in blocks and in SMPT orders. The algorithm starts by computing the flow time of this schedule. Each block is then successively moved to the first position (see Figure 3) and the new flow time is computed. The solution returned by the algorithm is therefore the one achieving the best flow time.



(a) SMPT Sequence



(b) B_f is moved in the first position.

Fig. 3. SMPT Sequence and Move Operation.

For sake of readability, let $F(f)$ denote the internal flow time of the block f, *i.e* the flow time of its jobs when starting at time 0: $\sum_1^{n_f} i \times p_f$

```
Algorithm 1: SMPT Scheduling without setup at time 0.
  Data: $n_f, p_f, s_f$ for $f \in \mathcal{F}$ in SMPT order ($MPT_f \leq MPT_{f+1}$) such that $n_f > 0$.
  Result: The optimal flow time $FT$.
  // Compute the flow time $FT_1$ of the SMPT sequence $\{B_1, \dots, B_{|\mathcal{F}|}\}$
  $FT_1 \leftarrow F(1)$;    $P \leftarrow n_1 \times p_1$;
  for $f \leftarrow 2$ to $|\mathcal{F}|$ do
      // shift jobs of $B_f$ and add their internal flow times
      $FT_1 \leftarrow FT_1 + n_f \times (P + s_f) + F(f)$
      $P \leftarrow P + s_f + (n_f \times p_f)$                       // Ending time of $B_f$
  // Note that $P$ is the makespan of the SMPT sequence
  // Compute the flow time when the block $B_f$ is in the first position
  $FT \leftarrow FT_1$;    $W \leftarrow n$;
  for $f \leftarrow |\mathcal{F}|$ to 2 do
      $W \leftarrow W - n_f$                        // Number of jobs in $\{B_1, \dots, B_{f-1}\}$
      $P \leftarrow P - s_f - (n_f \times p_f)$   // Ending time of $B_{f-1}$ in the SMPT sequence
      // Compute the variations of the flow time
      $\Delta_f \leftarrow -(P + s_f) \times n_f$                                   // For $B_f$
      $\Delta^- \leftarrow (n_f \times p_f + s_1) \times W$                  // For $\{B_1, \dots, B_{f-1}\}$
      $\Delta^+ \leftarrow (s_1 - s_f) \times (n - n_f - W)$              // For $\{B_{f+1}, \dots, B_{|\mathcal{F}|}\}$
      $FT_f \leftarrow FT_1 + \Delta^- + \Delta_f + \Delta^+$               // For the entire sequence
      if $FT_f < FT$ then $FT \leftarrow FT_f$              // Update the best flow time
```

The complexity for ordering the families in SMPT order is $O(|\mathcal{F}| \log |\mathcal{F}|)$. The complexity of Algorithm 1 is $O(|\mathcal{F}|)$. So, the complexity for finding the optimal flow time is $O(|\mathcal{F}| \log |\mathcal{F}|)$.

References

Malapert A., Nattaf M., 2019, "A New CP-Approach for a Parallel Machine Scheduling Problem with Time Constraints on Machine Qualifications" *CPAIOR 2019*, pp. 426-442.

Mason, A. J. and Anderson, E. J., 1991, "Minimizing flow time on a single machine with job classes and setup times" *Naval Research Logistics (NRL)*, Vol. 38(3), pp. 333-350.

Smith, W. E., 1956, "Various optimizers for single-stage production" *Naval Research Logistics (NRL)*, Vol. 3(1-2), pp. 59-66.

Using exponential smoothing to integrate the impact of corrective actions on project time forecasting

Annelies Martens[1] and Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
annelies.martens@ugent.be, mario.vanhoucke@ugent.be
[2] Technology and Operations Management Area, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, UK

Keywords: project management, project time forecasting, exponential smoothing.

1 Introduction

During project execution, uncertainty and risks often result in deviations from the plan. Therefore, project time forecasting is an important aspect of project management. The aim of this study is to improve the accuracy of project time forecasting by using exponential smoothing to account for the impact of corrective actions during project execution. The well-known project monitoring methodologies Earned Value Management (EVM, Fleming and Koppelman (2010)) and Earned Duration Management (EDM, Khamooshi and Golafshani (2014)) serve as a basis to forecast the final project duration. The forecasting accuracy of the approach presented in this study is evaluated for eight real-life projects that have been conducted in recent years and have been followed-up in real-time.

2 Project time forecasting

Project time forecasting is used to predict the final project duration during execution, given the current project performance. In recent literature, this topic has been explored in several studies. Barraza *et. al.* (2004) applied the concept of stochastical S-curves to improve the forecasting accuracy. De Marco *et. al.* (2009) reviewed the practicality and predictability of traditional forecasting methods by means of an empirical study. Monte Carlo simulations were used by Elshaer (2013) to incorporate the activity sensitivity measures in the project time forecasting process. Further, Kim and Kim (2014) examined the sensitivity of EVM forecasting methods to the characteristics of the planned value and earned value S-curves. Wauters and Vanhoucke (2015) used historical data and conducted a simulation experiment to study the stability and accuracy of EVM forecasts. Furthermore, Wauters and Vanhoucke (2016) and Wauters and Vanhoucke (2017) applied artificial intelligence methods for project time forecasting. Finally, exponential smoothing for project time forecasting has been applied by Khamooshi and Abdi (2016) and Batselier and Vanhoucke (2017) to give greater weights to the project performance of recent periods.

Exponential smoothing is a technique for time series forecasting that assigns exponentially decreasing weights from recent to older observations. Both Khamooshi and Abdi (2016) and Batselier and Vanhoucke (2017) use simple exponential smoothing, which can be expressed as follows:

$$s_0 = x_0 \tag{1}$$

$$s_t = \alpha x_t + (1-\alpha)s_{t-1} \tag{2}$$

with x_t the raw data, s_t the forecasted values and α the smoothing parameter. The smoothing parameter α indicates the level of forecasting responsiveness to the most recent periods and can vary from 0 to 1. The closer α is set to 1, the more weight is assigned to recent observations.

Since determining an appropriate value for α is important to achieve accurate project time forecasts, Batselier and Vanhoucke (2017) introduced two viewpoints to select a value for α, namely a static viewpoint and a dynamic viewpoint. The *static viewpoint* entails that a constant value for α is selected for the entire project execution. A *dynamic viewpoint* implies that the value for α may vary for different periods during project execution. In this viewpoint, the smoothing parameter may be adjusted based on human insight of the project manager (*qualitative dynamic viewpoint*) or by using a quantitative approach (*quantitative dynamic viewpoint*). While Khamooshi and Abdi (2016) applied the static viewpoint, Batselier and Vanhoucke (2017) applied both the static and the quantitative dynamic viewpoint.

In this study, a qualitative dynamic viewpoint to set the smoothing parameter will be applied in order to account for the impact of corrective actions. More precisely, corrective actions are interventions taken by the project manager to get the project back on track. Since these interventions affect the project progress, the smoothing parameter α will be adapted when a corrective action has been taken during the most recent period. The methodology section describes the applied procedure to select adequate smoothing parameter values and discusses the case study data used to evaluate this procedure.

3 Methodology

3.1 Selection of smoothing parameter values

Since both EVM and EDM are established methodologies for project time forecasting, our approach will be applied to both EVM and EDM project time forecasting. In order to account for the impact of corrective actions on the project outcome, two distinct smoothing parameters will be used (α_1 and α_2). The α_1 smoothing parameter will be used when no corrective actions have been taken in the previous period, otherwise smoothing parameter α_2 is used to forecast the expected project duration with exponential smoothing.

In the empirical experiment, the forecasting accuracy for α_1 and α_2 will be evaluated for values ranging from 0.1 until 1 (in steps of 0.1) in order to determine which values for the smoothing parameters are appropriate. A total of 10×10 combinations of α_1 and α_2 will thus be analysed. The forecasting accuracy of the best performing combination of α_1 and α_2 will be compared to the traditional EVM and EDM forecasting methods and to EVM/EDM project time forecasting with exponential smoothing.

3.2 Case study

Since the presented approach requires the timing of corrective actions throughout the project execution, real-life projects had to be followed up in real time to document the required information on the project progress and timing of corrective actions taken during this progress. Therefore, eight recent projects have been followed up in real-time. The main characteristics of these projects are summarised in Table 1. The columns #acts and #TPs represent the number of activities and the number of tracking periods at which a forecast has been made, respectively.

Table 1. Main characteristics of table

ID	Project description	Baseline start	Baseline end	Industry	BAC (EUR)	# acts	#TPs
P1	Apartment complex	30/07/15	14/08/17	Residential building	1.192.979	86	10
P2	Social Housing	20/01/17	28/05/18	Residential building	734.602	18	10
P3	Emergency Department	15/07/16	13/02/18	Civil construction	967.878	17	22
P4	Nuclear Healthcare	06/01/16	09/06/17	Civil construction	4.318.950	33	24
P5	Fuel Tank Filter	09/05/16	20/05/18	Production	1.456.000	15	10
P6	Production line change	31/10/16	01/09/18	Production	1.512.000	23	11
P7	Gluing machine	11/09/17	06/04/18	Production	107.500	8	10
P8	Labeling machine	04/09/17	09/02/18	Production	114.700	7	9

4 Results

In the empirical experiment, the forecasting accuracy is measured using the Mean Absolute Percentage Error (MAPE):

$$\text{MAPE} = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{A-F_t}{A}\right| \tag{3}$$

with T the number of periods, A the project duration at completion and F_t the forecasted project duration at period t. First, the presented approach is evaluated for EVM and EDM. Further, the most adequate combination of values for α_1 and α_2 is determined. Finally, the forecasting accuracy of the presented approach is compared to existing EVM and EDM forecasting methods. This comparison is made over the entire project lifecycle and for different stages of the project (early stage = $<$ 30% completion, middle stage = between 30% and 70 % completion and late stage = $>$ 70% completion).

The first part of the experiment showed that the presented approach applied to EDM resulted in a higher forecasting accuracy than EVM, for each combination of values for α_1 and α_2. More precisely, the MAPE for EDM is on average 14.49% lower than the EVM MAPE. This result is in line with previous studies on project time forecasting (Khamooshi and Abdi 2016, de Andrade *et al.* 2019). Further, the most accurate combination of smoothing parameters consists of a low value for α_1 (0.1) and a high value for α_2 (0.7). The low value for α_1, i.e., the smoothing parameter when no corrective actions have been taken in the previous period, is in line with the results of Khamooshi and Abdi (2016) and Batselier and Vanhoucke (2017). The higher smoothing parameter α_2 allows to emphasise that the project progress has been improved during the previous period due to a corrective action. Finally, compared to the traditional methods and exponential smoothing with a single smoothing parameter, the forecasting accuracy of the presented approach is slightly more accurate. However, when the forecasting accuracy is evaluated over the different project phases, the following observations are made. In the early phase, the accuracy of all forecasting methods is relatively low (average MAPE of 20%). This can be explained by the limited information that is available in this phase. In the middle and late phase, the accuracy of all forecasting methods improves. In these phases, the presented approach clearly outperforms the other forecasting methods. Compared to the standard EDM forecasting methods, the MAPE of the presented approach is more than 50% lower in the middle phase and 10% lower in the late phase. Compared to exponential smoothing with 1 smoothing parameter, the MAPE of the presented approach is 30% lower in the middle phase and 10% lower in the late phase.

To conclude, to obtain the most accurate project time forecasts during project execution, standard EDM forecasting can be used in the early project phase, while it is recommended to use the presented approach with $\alpha_1 = 0.1$ and $\alpha_2 = 0.7$ from the middle phase until project completion.

Future research could focus on collecting on additional projects to improve the currently available data on corrective actions during project execution. With this additional data, the most adequate smoothing parameters for different types of corrective actions and for different types of project can be determined.

References

Barraza, G.A., Back, W.E. and F. Mata, 2004, "Probabilistic forecasting of project performance using stochastic S curves", *Journal of Construction Engineering and Management*, Vol. 130, pp. 25-32.

Batselier, J. and M. Vanhoucke, 2017, "Improving project forecast accuracy by integrating earned value management with exponential smoothing and reference class forecasting", *International Journal of Project Management*, Vol. 35, pp. 28-43.

de Andrade, P., Martens A. and M. Vanhoucke, 2019, "Using real project schedule data to compare earned schedule and earned duration management project time forecasting capabilities", *Automation in Construction*, Vol. 99, pp. 68-78.

De Marco, A., Briccarello, D. and C. Rafele, 2009, "Cost and schedule monitoring of industrial building projects: Case study", *Journal of Construction Engineering and Management*, Vol. 135, pp 853-862.

Elshaer, R., 2013, "Impact of sensitivity information on the prediction of project's duration using earned schedule method", *International Journal of Project Management*, Vol. 31, pp.579-588.

Fleming, Q. and J. Koppelman, 2010, *"Earned value project management"*, Project Management Institute, Newton Square, Pennsylvania, 3rd edition.

Khamooshi, H. and H. Golafshani, 2014, "EDM: Earned Duration Management, a new approach to schedule performance management and measurement", *International Journal of Project Management*, Vol. 32, pp. 1019-1041.

Khamooshi, H. and A. Abdi, 2016, "Project duration forecasting using earned duration management with exponential smoothing techniques", *Journal of Management in Engineering*, Vol. 33.

Kim, B.-C. and H.-J. Kim, 2014, "Sensitivity of earned value schedule forecasting to S-curve patterns", *Journal of Construction Engineering and Management*, Vol. 140.

Wauters, M. and M. Vanhoucke, 2015, "Study of the stability of earned value management forecasting", *Journal of Construction Engineering and Management*, Vol. 141, pp. 1-10.

Wauters, M. and M. Vanhoucke, 2016, "A comparative study of Artificial Intelligence methods for project duration forecasting", *Expert systems with applications*, Vol. 46, pp.249-261.

Wauters, M. and M. Vanhoucke, 2017, "A Nearest Neighbour extension to project duration forecasting with Artificial Intelligence", *European Journal of Operational Research*, Vol. 259, pp. 1097-1111.

An Experimental Investigation on the Performance of Priority Rules for the Dynamic Stochastic Resource Constrained Multi-Project Scheduling Problem

Philipp Melchiors[1], Rainer Kolisch[1] and John J. Kanet[2]

[1] Technical University of Munich, Germany
rainer.kolisch@tum.de
[2] University of Dayton, USA
jkanet1@udayton.edu

Keywords: Multi-project, Project Arrival, Stochastic Activity Duration, Priority Policies.

1 Introduction

Project-based organizations such as R&D, are often matrix organizations with numerous functional departments. Projects arrive in a stochastic manner and their activities having typically stochastic durations are processed by the resources available in the functional departments. As a consequence projects suffer from delay and missed due dates. A common method for approaching these real-life complicated conditions is via implementing a dynamic scheduling policy using a priority rules for specifying the order of processing of activities by a resource. Despite the practical relevance of this dynamic stochastic resource constrained multi-project scheduling problem (DSRCMPSP) only limited research has been undertaken so far to address it. A major part of the literature related to the multi-project scheduling problem is dedicated to the static and deterministic case with the set of projects given at the outset and all parameters such as activity durations known in advance. Thus, the objective of our research is to provide a comprehensive comparison of well-known priority rules, which have been proposed for simpler settings for the weighted tardiness objective and to adapt them as needed for application to the more practically relevant DSRCMPSP. From the literature, we identify a number of priority policies which have shown good performance in prior computational studies. Whereas all these rules have shown promising results for the weighted tardiness objective they have never been compared together in the same study nor in the dynamic stochastic resource constrained setting. In order to introduce the problem formally, we follow the framework of (Adler *et. al.* 1995) and depict an organization as a set $\mathcal{R}$ of resources. Resource $r \in \mathcal{R}$ comprises c_r identical units, each capable of processing one activity at a time. Projects arrive dynamically according to a stochastic arrival process. We consider a stream of projects $j = 1, \ldots, J$ where each project j arriving at time a_j is of type $p_j \in \mathcal{P}$ and is assigned a due date D_j. The idea of project types reflects the fact that often projects have structure in common such as new development projects or reformulation projects. Project type $p \in \mathcal{P}$ has a weight w_p, an interarrival rate λ_p and is comprised of a set of activities $\mathcal{V}_p$ and a set of precedence relations $\mathcal{A}_p$ of the type finish-to-start with minimum time lag of 0. Each precedence relation between activity i and i' is written as a tuple $(i, i') \in \mathcal{A}_p$. In order to be processed, activity $i \in \mathcal{V}_p$ seizes one unit of resource $r_{ip} \in \mathcal{R}$ for a stochastic duration d_i with mean $\overline{d}_{ip}$. $\mathcal{E}_r(t)$ refers to the set of activities being processed by resource r at time t where each activity i of project j is referred to by tuple (i, j). Define the random variable C_{ij} as the realized completion time for activity i of project j and C_j the realized completion date of the project with $C_j = \max_{i \in \mathcal{V}_j} C_{ij}$. The objective is to minimize the long term expected weighted project tardiness with J being the total number of projects arriving at the system:

$$\text{Min}Z = \lim_{J\to\infty} E\left[\frac{1}{J}\sum_{j=1}^{J} w_j(0, C_j - D_j)^+\right] \tag{1}$$

subject to the following constraints

$$a_j + d_{ij} \le C_{ij} \quad \forall j = 1, \ldots, J; i \in \mathcal{V}_j \tag{2}$$
$$C_{ij} + d_{i',j} \le C_{i',j} \quad \forall j = 1, \ldots, J; (i, i') \in \mathcal{A}_{p_j} \tag{3}$$
$$|\mathcal{E}_r(t)| \le c_r \quad \forall r \in \mathcal{R}, t \ge 0 \tag{4}$$
$$C_j \ge C_{ij} \quad \forall j = 1, \ldots, J, i \in \mathcal{V}_j \tag{5}$$

The objective function (1) minimizes the long term expected weighted project tardiness Z where J denotes the number of arrived projects and $(x)^+$ is $\max(0, x)$. Constraints (2) forces each activity i of a project j not to start before the project has arrived at a_j. Constraints (3) specify the precedence constraints between the activities of each project j. Constraints (4) depict the resource constraints: At any point in time t the number of activities that are in process by resources r, $|\mathcal{E}_r(t)|$, must not be greater than the number of available units, c_r of that resources. Constraint (5) defines the project completion time for project j.

2 Priority rules and experimental design

Rules addressing the weighted tardiness objective typically combine information about the input parameters: due date, activity processing time, and weight. There two main approaches according to which the pieces of information are combined. The first approach is to consider some ratio involving due dates and processing times. A second approach is to consider a time-sensitive binary switch in emphasis from due date to processing time. Thus, we distinguish between ratio rules and binary switch rules. Probably the first evidence of the "ratio" approach is found in the work of (Carroll 1965) and his so-called "c over t" dispatching rule which was inspirational for a series of work by (Lawrence and Morton 1993), and (Morton and Pentico 1993). For binary switch rules the seminal work is provided by (Baker and Bertrand 1982) with their "modified due date" priority rule and later further developed by (Baker and Kanet 1983), (Dumond and Mabert 1988), (Anderson and Nyirenda 1990), and (Kanet and Li 2004).

The following ratio rules have been considered:

– BD with Myopic Activity Costing (BD-MC)
– BD with Global Activity Costing and Uniform Resource Pricing (BD-GC-U)
– BD with Global Activity Costing and Dynamic Resource Pricing (BD-GC-D)

The major rules falling into the category of binary switching rules include the various forms of the "modified due date" approach first provided by Baker and Bertrand (Baker and Bertrand 1982). We study several variations of the modified due date approach applying it to the DSRCMPSP.

– Weighted Modified Due Date (WMDD)
– Weighted Modified Operation Due Date (WMOD)
– Weighted Critical Ratio and Shortest Processing Time (W(CR+SPT))
– Weighted Critical Ratio and Global Shortest Processing Time (W(CR+GSPT))
– Weighted Due Date Modified Shortest Activity from Shortest Project (WSASP-DD)

Finally, for benchmarking purposes we have added some simple but well known rules. They have been selected as they are frequently used in related studies of multi-project and job shop scheduling problems:

– First-Come, First-Serve (FCFS)
– Weighted Minimum Slack (WMINSLK)
– Weighted Shortest Processing Time (WSPT)
– Random (RAN)

For our study we created a number of problem instances with various parameters being controlled. In brackets the values used are indicated.

– α_p (20%,80%): Percentage of tardy projects of type p when using the RAN scheduling policy. This implicitly controls due date tightness.
– $\mathcal{R}$ (1, 3, 5, 10, 15, 20): Number of resources.
– ρ (0.7, 0.9): Resource utilization.
– $\text{CV}^{\overline{d}}$ ($[0, 0.4]$, $[0.8, 1]$): Coefficient of variation of the expected durations belonging to the activities processed by each resource.
– OS (0, 0.2, 0.4, 0.6, 0.8, 1.0): Order strength of the project networks.

In each problem instance we have two project types ($p \in \{1, 2\}$) having two different networks respectively but with the same order strength and number of activities (20). Finally, we obtained $1,152$ instances. For each problem instance we simulate each of the 11 priority policies in a simulation run which results in $12,672$ runs.

3 Summary of the results

In the following we provide the results of the simulation study. The performance of a rule is measured as the normalized weighted tardiness $Z^n(\pi)$ given by $Z^n(\pi) = Z(\pi)/Z(\pi_{\text{RAN}})$. Duncan tests were undertaken to detect groups of rules where rules in different groups show a significant difference in performance while rules in the same group do not. Table 1 provides the performance of all priority rules as well as the group a rules belongs to, according to the Duncan test. W(CR+SPT) is the best rule being significantly superior to all other rules. This extends the findings of (Kutanoglu and Sabuncuoglu 1999) for the dynamic job shop problem.

Rule	$\overline{Z}^n(\pi)$	Group
W(CR+SPT)	0.31	g
BD-GC-D	0.44	f
BC-MC	0.46	f
WMOD	0.51	f
WMINSLK	0.62	e
FCFS	0.63	e
WSASP-DD	0.81	d
BD-GC-U	0.95	c
WSPT	0.98	bc
W(CR+GSPT)	1.07	ab
WMDD	1.11	a

Table 1. Overall performance of the priority rules and group according to the Duncan test

References

Adler A.S., V. Nguyen and E. Schwerer, 1995, “From Project to Process Management: An Empirically-based Framework for Analyzing Product Development Time.", *Management Science*, Vol. 41, pp. 458-484.

Anderson E. J., Nyirenda J.C., 1990, “Two new rules to minimize tardiness in a job shop.", *International Journal of Production Research*, Vol. 28, pp. 2277-2292.

Baker K. R., Bertrand J. W. M., 1982, “A dynamic priority rule for scheduling against duedates.", *Journal of Operations Management*, Vol. 1, pp. 37-42.

Baker K. R., Kanet J. J._B, 1983, “Job shop scheduling with modified due dates.", *Journal of Operations Management*, Vol. 4, pp. 11-22.

Carroll D. C., 1965, “Heuristic Sequencing of Single and Multiple Component Jobs", *PhD thesis*, Sloan School of Management, MIT.

Dumond J., Mabert V. A., 1988, “Evaluating Project Scheduling and Due date Assignment Procedures: An Experimental Analysis.", *Journal Title*, Vol. 34, pp. 101-118.

Kanet J. J., Li X., 2004, “A weighted modified due date rule for sequencing to minimize weighted tardiness.", *Journal of Scheduling*, Vol. 7, pp. 261-276.

Kutanoglu E., Sabuncuoglu I., 1999, “An analysis of heuristics in a dynamic job shop with weighted tardiness objectives.", *International Journal of Production Research*, Vol. 37, pp. 165-187.

Lawrence S. R., Morton T. E., 1993, “Resource-constrained multi-project scheduling with tardy costs: Comparing myopic, bottleneck, and resource pricing heuristics.", *European Journal of Operational Research*, Vol. 64, pp. 168-187.

Morton T. E.,Pentico D. W., 1993, “Heuristic Scheduling Systems", *Wiley Series in Engineering & Technology Management*, John Wiley and Sons,Inc..

Conditional Value-at-Risk of the Completion Time in Fuzzy Activity Networks

Carlo Meloni[1], Marco Pranzo[2] and Marcella Samà[3]

[1] Politecnico di Bari, Italy
carlo.meloni@poliba.it
[2] Università di Siena, Italy
marco.pranzo@unisi.it
[3] Università Roma Tre, Italy
sama@ing.uniroma3.it

Abstract. The research deals with the evaluation of the Conditional Value-at-Risk (*CVaR*) for the completion time in scheduling problems represented as temporal activity networks where we assume that only a fuzzy representation for the activity integer valued durations is known to the scheduler. More precisely, we address the evaluation of the *CVaR* associated to a feasible schedule, and we extend the approach recently proposed for the case of interval valued durations. We develop and analyze a suitable computational method to obtain the fuzzy evaluation of the *CVaR* of the completion time of a given schedule. The proposed method enables to use the *CVaR* as quality criterion for wide classes of scheduling approaches considering risk-aversion in different practical contexts when only a fuzzy representation of activity durations is known.

Keywords: *CVaR*, Project Scheduling, Makespan, Fuzzy intervals, Activity Networks

1 Introduction

This paper addresses the evaluation of the Conditional Value-at-Risk (*CVaR*) of the completion time (hereinafter indicated as makespan) when the scheduling problem is represented by a temporal network (Elmaghraby 1977). At this aim, we consider scheduling models with a fixed and given set of precedence relations, but (independent) fuzzy processing times. More specifically, this research work assumes that non-deterministic durations are considered as fuzzy sets (Słowiński and Hapke 2000). The objective is to evaluate the *CVaR* of the makespan which is in general an NP-complete problem (Hagstrom 1988). We extend the approach recently proposed in (Meloni and Pranzo 2020), by developing and analyzing a suitable computational method to obtain the fuzzy evaluation of the *CVaR* of the makespan of a given schedule. The proposed algorithms are pseudo-polynomial in the general case, but they have been tested on a wide set of realistic instances for the experimental validation of their efficiency. The evaluation of the *CVaR* of the uncertain makespan $\mathbf{C}_{max}$ is an issue arising in several practical applications such as: project or task bidding, risk evaluation and mitigation, due date setting or acceptance, order proposal and acceptance, and robust scheduling (Elmaghraby 2005, Lawrence and Sewell 1997). The goals of this research is to propose and test a method for computing the *CVaR* of the makespan in the case of fuzzy durations, and to stimulate future research works devoted to the development of algorithms and/or other useful performance measures.

2 The *CVaR* of the makespan

In many applications the scheduler may be risk-averse and may prefer solutions that do not just perform well “on average", but that also perform satisfactorily “in most cases" (Elmaghraby 2005). Nevertheless there is no universally accepted single risk measure. In fact, each measure has its own advantages and disadvantages (Bertsimas *et. al.* 2004) and the choice also reflects a subjective preference of the decision makers. Several scalar performance indicators have been used to characterize the makespan $\mathbf{C}_{max}$ of a stochastic activity network. They include the *CVaR* at a probability level γ:
$CVaR_\gamma(\mathbf{C}_{max}) = E(\mathbf{C}_{max}|\mathbf{C}_{max} \geq q_\gamma(\mathbf{C}_{max}))$. It is also called γ-Tail Expectation or Expected shortfall (at level γ) of the makespan, and is related to the Value-at-Risk $VaR_\gamma(\mathbf{C}_{max})$ (Bertsimas *et. al.* 2004, Rockafeller 2007): $CVaR_\gamma(\mathbf{C}_{max}) = \frac{1}{1-\gamma}\int_\gamma^1 VaR_\beta(\mathbf{C}_{max})\,d\beta$.
Where $VaR_\gamma(\mathbf{C}_{max}) = inf\{t : prob(\mathbf{C}_{max} \leq t) \geq \gamma\}$ is a measure commonly used in finance which are gaining momentum also in scheduling (Lawrence and Sewell 1997, Sarin *et. al.* 2014). $VaR_\gamma(\mathbf{C}_{max})$ represents a threshold that is exceeded in $(1-\gamma)100\%$ of all cases, while the $CVaR_\gamma(\mathbf{C}_{max})$ represents the expected value of all cases exceeding the threshold $VaR_\gamma(\mathbf{C}_{max})$. Considering the definitions, the following holds for all $\gamma \in [0, 1]$: $VaR_\gamma(\mathbf{C}_{max}) \leq CVaR_\gamma(\mathbf{C}_{max})$. If a decision maker is not only concerned with the frequency of undesirable outcomes, but also with their severity, $CVaR_\gamma$ is recommended instead of VaR_γ (Bertsimas *et. al.* 2004, Sarin *et. al.* 2014). Higher values of γ are chosen by decision makers who are more risk-averse, and $\gamma = 0$ represents the risk-neutral choice. In fact, as γ tends to 1 (i.e., its upper extremum), the $CVaR_\gamma(\mathbf{C}_{max})$ tends to the worst case W; while when γ tends to 0, $CVaR_\gamma(\mathbf{C}_{max})$ tends to the expected value of the makespan $E(\mathbf{C}_{max})$. The features offered by *CVaR* are also useful in planning and scheduling problems based on stochastic activity networks models (Meloni and Pranzo 2020, Sarin *et. al.* 2014).

3 Fuzzy temporal activity networks

A *fuzzy temporal activity network* (*FTAN*) is a temporal activity nertwork with fuzzy valued durations. It can be defined by the pair $(G, \mathbf{D})$, where $G = (N, A)$ is the precedence DAG, the set of nodes N is associated to events, the set A of arcs represents the activities, and $\mathbf{D} = (\mathbf{D}_1, \dots, \mathbf{D}_m)$ is the vector of m fuzzy durations associated to the m arcs in A representing the activities. The network is directed, connected, and acyclic with single source and sink nodes. In a *FTAN*, all quantities that depend on the activity durations have a fuzzy characterization. Therefore, the starting and completion time of any activity, and the makespan $\mathbf{C}_{max}$ are all fuzzy quantities, i.e., fuzzy sets of the real line $\mathbb{R}$. A fuzzy set M of the universe of values X is characterized by a membership function μ_M which takes its value in interval $[0, 1]$. For each element $x \in X$, $\mu_M(x)$ defines the degree to which x belongs to $M = \{x \in X; \mu_M(x) \in [0, 1]\}$. An α-level cut (or α-cut, for short) of M is the crisp set $M_\alpha = \{x \in X|\mu_M(x) \geq \alpha\}$. The support of M is the crisp set $supp(M) = \{x \in X|\mu_M(x) > 0\}$. The duration of all the activities is a fuzzy number which is defined as a bounded support fuzzy quantity whose α-cuts are closed intervals. More specifically, we consider integer fuzzy numbers which are characterized as follows: *i)* the support is a closed integer interval denoted as $[\underline{M}, \overline{M}]$; *ii)* M is normal, i.e., there exists $\hat{x} \in [\underline{M}, \overline{M}]|\mu_M(\hat{x}) = 1$; *iii)* for any $x_1, x_2 \in [\underline{M}, \hat{x}]$, $\mu_M(x_1) \leq \mu_M(x_2)$ holds; and *iv)* for any $x_1, x_2 \in [\hat{x}, \overline{M},]$, $\mu_M(x_1) \geq \mu_M(x_2)$ holds. Given the fuzzy durations $\mathbf{D} = (\mathbf{D}_1, \dots, \mathbf{D}_m)$, the extension principle (Zadeh 1965) provides a powerful technique to extend a real continuous function of the activity durations (such as $CVaR_\gamma(\mathbf{C}_{max})$) to a fuzzy function $F(\mathbf{D})$ of the fuzzy durations $\mathbf{D}$. Moreover, the decomposition by α-cuts can be used to compute that fuzzy function by means of a decomposition method (Nguyen 1978): $[F(\mathbf{D})]_\alpha = F([\mathbf{D}_1]_\alpha, \ldots, [\mathbf{D}_m]_\alpha)$. According to this method, the membership function $\mu_F(x)$ of $F(\mathbf{D})$ can be reconstructed from its α-cuts F_α as follows: $\mu_F(x) = max\{\alpha : x \in F_\alpha\}$. We adopt a piecewise linear model for the membership function of the activity durations to simplify either information collection and computational aspects. We use a representation based on the full breakpoints ordered sequence which is very general and can be easily adapted to the cases of popular models such as triangular, trapezoid, and six-points approximated functions (Fortemps 1997).

4 Evaluation of the $CVaR$ of the makespan in *FTANs*

The case of crisp (i.e., ordinary) interval durations can be considered as a special case of *FTAN*. For this special case, in (Meloni and Pranzo 2020) an algorithmic approach has been proposed and experimentally validated. This approach is based on a counting approach to evaluate the $CVaR_\gamma$ for the makespan. The counting approach, starting from the pessimistic makespan (i.e., the worst possible makespan W), counts backwards how many configurations lead to each possible makespan value. This process is implemented as an iterative procedure which continues until sufficient information has been gathered to compute the $CVaR$ at a desired probability level γ.

On the basis of the computational results reported in (Meloni and Pranzo 2020), in this work we adopt an algorithm configuration for the crisp case which is able to determine a fast and extremely good estimation of $CVaR$ of the makespan in $O(\Gamma^2 m^2)$, where m is the number of arcs of the network, and Γ is the amount of uncertainty of the activity network represented by the size (in terms of number of integers) of the time interval $[\underline{C}_{max}, \overline{C}_{max}]$, where $\underline{C}_{max}$ ($\overline{C}_{max}$) is the makespan when all the activities durations are at their minimum (maximum) value. According to the α-cuts decomposition method, we follow an approximated approach for the evaluation of $CVaR_\gamma$ of the makespan for more general *FTANs*. To this aim, the basic algorithm for $CVaR_\gamma$ evaluation involving ordinary (i.e., non-fuzzy) intervals can be extended to solve the fuzzy cases, by the decomposition of the membership functions of the activity durations into a finite number of α-cuts. In the proposed method the basic algorithm can be applied on the instances associated to the selected α-levels to obtain the corresponding α-cuts of the desired fuzzy $CVaR_\gamma$ evaluation. This finite number of α-cuts are used to obtain an approximated reconstruction of the membership function $\mu_F(x)$ of $CVaR_\gamma$. Applying the general reconstruction rule described in the previous section (i.e., $\mu_F(x) = max\{\alpha : x \in F_\alpha\}$) to a finite set of α-cuts produces a stepped function that can be interpolated with a piece-wise linear function using the extreme points of the α-cuts as breakpoints. More specifically, in the α-cuts decomposition, for each selected level α, each fuzzy duration is cut at level α. This decomposition gives a set of activity networks with interval valued durations, each of which can be solved as a crisp instance. Then, in the successive fuzzy reconstruction procedure, an approximation of the fuzzy membership function of $CVaR_\gamma$ is determined from their α-cuts (e.g., see (Fargier *et. al.* 2000)).

This method is simple to implement but it could be intractable if ran for too many cuts and can be carried out only on a selection of suitably chosen level-cuts. Thus an issue comes from the selection of the relevant α-cuts. Possible solutions include: *i)* to choose α-cuts arbitrarily, e.g., according to a precision degree fixed by the user; *ii)* to use a given number of α-cuts at fixed α levels. Since we consider fuzzy durations represented by piece-wise linear functions, we adopt a more suitable choice of the levels α allowing for an accurate computation of the fuzzy quantities of interest. In fact, the resulting fuzzy quantities will be described by piece-wise linear functions too. Hence, the relevant α-cuts will be those corresponding to the breakpoints (i.e., of the right or left parts) of these fuzzy intervals.

Assuming these levels known (otherwise they can be easily determined in a pre-processing phase), an exact (approximate) interval-based procedure applied to the breakpoint values would compute the actual (approximate) fuzzy *CVaR* values. In the proposed method, the α-cuts decomposition has a preliminary step devoted to determine the α-levels to adopt on the basis of either a specific input or default setting. We can then apply the algorithm to solve the case of interval valued durations to know the corresponding α-cuts of the $CVaR_\gamma$. The fuzzy $CVaR_\gamma$ is then reconstructed from its α-cuts and returned as output. Considering the number K of α-cuts used in the adopted decomposition, the overall complexity of the algorithm is $O(K\Gamma^2 m^2)$. In particular, associating the α-cuts to breakpoints (which is adopted as default scheme in our method) yields to a complexity $O(K_B \Gamma^2 m^2)$, where K_B is the overall number of the different α values in breakpoints in the durations contained in the *FTAN*.

A computational study is conducted to test the proposed approach on a benchmark set of realistic project scheduling problems. The overall speed and quality of the proposed method makes it an enabling tool for the use of *CVaR* as an analysis criterion in fuzzy scheduling problems, while the trade-off between accuracy and computation effort indicates a possible research direction regarding the strategies for choosing the decomposition scheme in terms of both structure and size of the α sample set. Further research directions include the application of the proposed methodology in real contexts, and the improvement in the performance of the algorithm devoted to solve crisp-interval instances.

References

Bertsimas D., Lauprete G.J., Samarov A., 2004, “Shortfall as a risk measure: properties, optimization and applications", *Journal of Economic Dynamics & Control* 28, (7), 1353-1382.

Elmaghraby S.E., 2005, “On the fallacy of averages in project risk management". *European Journal of Operational Research* 165, (2), 307-313.

Fargier H., Galvagnon V., Dubois D., 2000, “Fuzzy PERT in series-parallel graphs". *Ninth IEEE International Conference on Fuzzy Systems. FUZZ-IEEE 2000* vol. 2, 717–722.

Fortemps P., 1997, “Jobshop Scheduling with Imprecise Durations: A Fuzzy Approach". *IEEE Transactions on Fuzzy Systems* 5, (4), 557–569.

Hagstrom J.N., 1988, “Computational Complexity of PERT Problems". *Networks* 18, (2), 139–147.

Lawrence S.R., Sewell E.C., 1997, “Heuristic, optimal, static, and dynamic schedules when processing times are uncertain", *Journal of the Operations Management* 15, 71–82.

Meloni C., Pranzo M., 2020, “Expected shortfall for the makespan in activity networks under imperfect information". *Flexible Services and Manufacturing Journal*, 32, 668-692.

Nguyen H.T., 1978, “A note on the extension principle for fuzzy sets". *Journal of Mathematical Analysis and Applications* vol. 64, 369–380.

Rockafeller R.T., 2007, “Coherent approaches to risk in optimization under uncertainty". *INFORMS Tutorials in Operations Research*, 38â“-61.

Sarin S.C., Sherali H.D., Liao L., 2014, “Minimizing conditional-value-at-risk for stochastic scheduling problems". *Journal of Scheduling* 17, 5–15.

Elmaghraby S.E., 1977, *Activity Networks: Project Planning and Control by Network Models.* Wiley, New York.

Słowiński R., Hapke M., (Eds.), 2010, *Scheduling under Fuzziness.* Physica-Verlag, Heidelberg.

Zadeh L., 1965, “Fuzzy sets". *Journal of Information and Control* vol. 8, 338–353.

A column generation algorithm for the single machine parallel batch scheduling problem

Onur Ozturk[1]

Telfer School of Management, University of Ottawa, 55 Laurier Avenue East, Ottawa, ON, Canada K1N 6N5
oozturk@uottawa.ca

Keywords: Batch scheduling, dynamic programming, column generation, truncated solutions

1 Introduction

Batch scheduling is the type of scheduling where jobs are grouped into batches and processed together. In parallel batch scheduling (p-batch), jobs of the same batch are all processed at the same time and in the same machine. In our problem, jobs have different release dates, r_j, different processing times, p_j, and sizes, v_j. Batches have a fixed capacity B that sum of job sizes in a batch must not exceed. Once their composition is determined, batches may also have different processing times such that each batch must be processed at least as long as the longest processing time of jobs included in that batch. We aim to minimize the total flow time, i.e., $\sum C_j$. Inspired from Graham's notation ((Graham, Lawler, Lenstra & Rinnooy Kan 1979)), our problem can be represented as $1/p-Batch, r_j, p_j, v_j, B/\sum C_j$. Most solution methods for p-batch problems in the literature are metaheuristic methods ((Jia, Zhang, Long, Leung, Li & Li 2018)). In this study, we develop a time indexed column generation model capable of finding high quality solutions within short computational times. Inspired from a 0-1 set partitioning model, each column represents a set of jobs to be processed in the same batch at a given instant. At the end of each iteration, we heuristically obtain primal solutions derived by iteratively rounding the linear programming solution of the column generation model.

2 Set partitioning formulation for the Master Problem

The input of the problem is twofold: n jobs, indexed from 1 to n to be batched complying with batch capacity and a total length T before which all batches must be processed at an instant t such that $1 \leq t \leq T$. Thus for each batch b with processing duration p_b and processing starting time t, time indices $t' \in \{t, t+1, ..., t+p_b-1\}$ indicate the consecutive discrete time instants during which batch b is processed. Let $Set_b \in \{b_1, b_2, ..., b_P\}$ be the set of all feasible job assignments and C_b the cost of processing batch b, i.e., the sum of job completion times of batch b. Let a_{jb} be equal to 1 if job j is assigned to batch b ($\forall j \in \{1, ..., n\}$), 0 otherwise. Also let a_{tb} be equal to 1 if batch b is being processed at instant t ($\forall t \in \{n+1, ..., T\}$), 0 otherwise. Then we have the following conditions for a feasible job assignment and batch processing: $\sum_{j=1}^{n} a_{jb}v_j \leq B$, $p_b = max\{a_{jb} * p_k\}$ ($\forall j \leq n$) and $\sum_{t=n+1}^{n+T} a_{tb} = p_b$. Let C_b be the cost of processing batch b, i.e., total flow time of jobs included in batch b. C_b is calculated as $C_b = \sum_{j=1}^{n} a_{jb} * C_j$ where $C_j = max\{t * a_{tb}\}$).

The decision to take is then if a batch b with a precise processing starting time should be included in the solution. Hence, let the binary decision variable x_b be equal to 1 if batch (column) b is part of the solution and 0 otherwise ($b = 1, ..., P$). We can model the master problem as follows:

$$\text{Min } \sum_{b=1}^{P} x_b * C_b \tag{1}$$

$$\text{s.t.} \tag{2}$$

$$\sum_{b=1}^{P} a_{jb} * x_b = 1 \quad j = 1,..,n \tag{3}$$

$$\sum_{b=1}^{P} a_{tb} * x_b \leq 1 \quad t = n+1,..,n+T \tag{4}$$

$$x_b \in \{0,1\} \quad \forall b = 1,...,P \tag{5}$$

In the MP model above, the objective function is the minimization of total flow time. The first constraint set assigns each job to a single batch. The second constraint set indicates that a time instant can be occupied by the processing of at most one batch. We relax the integrity condition for variable x_b and solve the linear relaxation of the MP model by column generation. The master problem is initiated with a restricted set of columns where initial batches are generated with a heuristic regardless of job release dates and processing times. This heuristic generates intervals of different lengths covering iteratively 1, 2,..., n jobs. Then jobs whose release dates fall into the same interval are batched together with the first fit decreasing heuristic. Finally, batches are scheduled as if there is a single machine to have a sufficiently large value for T.

3 Column generation for solving the sub-problem

The sub-problem aims to search for the column with the most negative reduced cost and not currently included in the master problem. The sub-problem is an optimization problem whose constraints are defined by the three fundamental dimensions of the original problem: batch sizes cannot be greater than B, a batch cannot be processed before the greatest release date of jobs included in that batch, batch processing time is given by the longest processing time of jobs in the batch. The dual values of the master problem and the cost of processing the new generated batch, i.e., new column, help to determine the objective function of the sub-problem. The objective function is expressed as $\sum_{j=1}^{n} C_j - (\sum_{j=1}^{n} y_j * \pi_j - \sum_{j=n+1}^{n+T} y_j * \pi_j)$ where C_j is the completion time of job j in the column, y_j is the binary variable indicating if job j makes part of the column ($\forall j \in \{1,...,n\}$) and if instant t is occupied by the processing of the column ($\forall j \in \{n+1,...,T\}$), finally π_j is the dual variable corresponding to constraint j in the master model ($\forall j \in \{1,...,T\}$). We can present the sub-problem in the form of the following minimization problem:

$$\text{Min } \sum_{j=1}^{n} C_j - (\sum_{j=1}^{n} y_j * \pi_j - \sum_{j=n+1}^{n+T} y_j * \pi_j) \tag{6}$$

$$\text{s.t.} \tag{7}$$

$$\sum_{j=1}^{n} y_j * v_j \leq B \tag{8}$$

$$r_b \geq y_j * r_j \quad j = 1,..,n \tag{9}$$

$$p_b \geq y_j * p_j \quad j = 1,..,n \tag{10}$$

$$r_b \leq t + Q(1 - y_j) \quad j = n+1,..,n+T,\ t = j - n - 1 \tag{11}$$

$$r_b + p_b - 1 \geq y_j * t \quad j = n+1,..,n+T,\ t = j - n - 1 \tag{12}$$

$$\sum_{j=n+1}^{n+T} y_j = p_b \tag{13}$$

$$C_j \geq r_b + p_b - Q(1 - y_j) \quad j = 1, .., T \tag{14}$$

$$y_j \in \{0, 1\}, C_j \geq 0, r_b \geq 0, p_b \geq 0 \tag{15}$$

Constraint 8 is the capacity constraint. Constraint sets 9 and 10 determine the starting time, r_b and the processing duration, p_b, of the new generate batch b. Constraint sets 11, 12 and 13 ensure the continuity of the processing for p_b units of time. Finally, constraint set 14 sets the flow time value for jobs in the batch.

Solving the sub-problem with the above model is time consuming. Thus, we developed a pseudo-polynomial dynamic programming (DP) algorithm which solves the sub-problem faster than the above integer model for the case of integer job sizes. The idea of the DP algorithm is to decide what should be the length of the batch (i.e., processing duration) and at which instant the batch should be processed. Then, jobs inducing the smallest reduced cost can be found complying with the batch capacity. Hence, the DP algorithm is controlled by four parameters: processing beginning instant t, processing duration p_b, jobs in batch j, capacity use noted cap. Let $f(.)$ be a four dimensional array to enumerate recursively the minimum reduced cost. We define $f(t, p_b, j, cap)$ as the reduced cost of including job j in a batch whose used capacity is cap with processing duration p_b and processing starting instant t. Initially we set $f(t, p_b, j, cap)$ to infinity $\forall t \in \{0, ..., T\}, p_b \in \{p_1, ..., p_n\}, cap \in \{0, ..., B\}, j \in \{1, ..., n\}$. Then, for each different processing time p_b and potential batch processing instant t, we set $f(t, p_b, 0, v_{arg(p_j=p_b)}) = C_{arg(p_j=p_b)} - \pi_{arg(p_j=p_b)} + \sum_{j=n+t}^{n+t+p_b-1} \pi_j$ $\forall r_{arg(p_j=p_b)} \leq t$. The interpretation of this calculation is the following: batch processing time p_b is computed for all different job processing times as long as release date of the job determining p_b is available before (or at) batch processing instant t. The forth dimension of function $f(.)$ is then set to the size of the job determining p_b. Then, for all other jobs which are not currently in the batch, we recursively compute the minimal reduced cost value: $f(t, p_b, j, cap) = min_{\forall j' < j, cap' \leq cap} f(t, p_b, j', cap') + t + p_b - \pi_j$

The smallest reduced cost is then equal to $min_{\forall t, p_b, cap} f(t, p_b, n, cap)$. There are n possible different processing times for p_b and the parameter t is bounded by T. For each different value of j, two for loops can be run for each of j' and cap' which are bounded by n and B, respectively. Thus, the DP can be implemented in $O(n^3 BT)$ time.

4 Obtaining primal solutions

We implement a rounding technique similar to the one applied by (Mourgaya & Vanderbeck 2007). Primal solutions are obtained by truncating the relaxed solutions of the MP model. After solving the linear relaxation of the initial MP by column generation, we lookup columns/batches whose decision variable value is equal one, i.e., $x_b = 1$. If there is such a column, then it is registered as a validated batch. If there is no such column, then we select the column having the largest x_b value. (If there are multiple columns with the same x_b value, tie is broken by selecting the column with the smallest C_b value where C_b is the sum of job completion times in batch b.) Then, that column is selected as a validated batch. After validating a batch, jobs of that batch are erased from the original problem instance and a residual problem is obtained. Afterwards, the same solution methodology is applied to the residual problem until all jobs are batched.

Once all batches are obtained, it is now time to schedule them on the single machine. The problem is equivalent to a single machine problem in the presence of jobs with different processing times, release dates and weights since batches can contain different numbers of jobs. Inspired from Graham's notation, it can be noted as $1/r_j/\sum w_j C_j$ which is also an NP-hard problem ((Belouadah, Posner & Potts 1992)). However, this problem can be efficiently solved with a time indexed modeling as suggested by (Unlu & Mason 2010).

5 Numerical experimentation

We tested instances containing 10 to 50 jobs in the presence arbitrary release dates generated in the following three ways: $r_j \in U[0,5]$, $r_j \in U[0,25]$ and $r_j \in U[0,5*n]$. Job processing times were generated with $p_j \in U[1,10]$ distribution and batch capacity B was set to 5, 25 and 50 while integer job sizes were generated with a U[1, B/2] distribution. For each combination of number of jobs, release date type and batch capacity, 5 problem instances were generated and tested. We used CPLEX 12.8 for all numerical tests.

Table 1. Average optimality gaps

	Cap = 5					Cap = 25					Cap = 50				
Nbr jobs:	10	20	30	40	50	10	20	30	40	50	10	20	30	40	50
$r_j \in U[0,5]$	4.5	3.5	5.4	1.6	3.8	0.9	5	10.2	13	22.5	5.6	8.3	10.4	1.9	4.5
$r_j \in U[0,25]$	0.6	4.5	6.3	5.3	9.1	3.5	7.1	14.2	13.4	21	2.7	4.7	6.9	11.1	19.5
$r_j \in U[0,5*n]$	0	0.9	0.4	0.2	0	2.1	0.1	0.1	0.2	0.2	0	0.5	0.5	0	0

We compared the solution quality of our method to the solutions given by a straightforward mixed integer linear programming model of the problem where the solution time limit is set to 1800 seconds. Table 1 shows the average optimality gaps of our solution method. We see that the hardest instances are those with a large batch capacity in the presence of medium release dates, i.e., $r_j \in U[0,25]$. Other than that the results are promising and most of the time the optimality gap is close to zero when $r_j \in U[0,5n]$. Moreover, the solution times with the column generation model are very small. The largest instances, containing 50 jobs with a batch capacity of 25, are solved in the worst case within 400 seconds of CPU time.

6 Conclusion

We presented in this study a column generation solution method for the parallel batch scheduling of jobs with different release dates, processing times and sizes. Numerical tests show that the proposed solution method is able to give high quality solutions within short computational times. For future work, we aim to accelerate the sub-problem to decrease the overall solution time of the algorithm.

References

Belouadah, H., Posner, M. E. & Potts, C. N. (1992), 'Scheduling with release dates on a single machine to minimize total weighted completion time', *Discrete applied mathematics* **36**(3), 213–231.

Graham, R., Lawler, E., Lenstra, J. & Rinnooy Kan, A. (1979), 'Optimization and approximation in deterministic sequencing and scheduling: a survey', *Annals of Discrete Mathematics* **5**, 287–326.

Jia, Z.-h., Zhang, H., Long, W.-t., Leung, J. Y., Li, K. & Li, W. (2018), 'A meta-heuristic for minimizing total weighted flow time on parallel batch machines', *Computers & Industrial Engineering* **125**, 298–308.

Mourgaya, M. & Vanderbeck, F. (2007), 'Column generation based heuristic for tactical planning in multi-period vehicle routing', *European Journal of Operational Research* **183**(3), 1028–1041.

Unlu, Y. & Mason, S. J. (2010), 'Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems', *Computers & Industrial Engineering* **58**(4), 785–800.

Mixed-Integer Programming Formulations for the Anchor-Robust Project Scheduling Problem

Adèle Pass-Lanneau[1,2], Pascale Bendotti[1,2], Philippe Chrétienne[2] and Pierre Fouilhoux[2]

[1] EDF R&D, Palaiseau, France
{adele.pass-lanneau, pascale.bendotti}@edf.fr
[2] Sorbonne Université, CNRS, LIP6, Paris, France
{philippe.chretienne, pierre.fouilhoux}@lip6.fr

Keywords: project scheduling, robust 2-stage optimization, anchored decisions, mixed-integer programming.

1 Introduction

In project management, computing a schedule ahead of time is often necessary to ensure the availability of equipment or staff. When the schedule is about to start, the project data may have changed, e.g., some jobs may be longer than expected in the first place. Hence the schedule computed previously has to be modified accordingly. However, in the context of an industrial complex project, some jobs might be difficult or costly to reschedule. The decision maker thus needs a guarantee over the starting times of these jobs when choosing an initial schedule.

The present work follows the general framework of (Bendotti *et al.* 2017), (Bendotti *et al.* 2019) to integrate such a criterion. We consider a project scheduling problem where jobs must be scheduled while respecting precedence constraints. Processing times of jobs have a nominal value but they may deviate from it by an uncertain amount, which is supposed to be lying in an *uncertainty set.* A *baseline schedule* is a schedule of the instance with nominal processing times that satisfies a given global deadline M. Given a baseline schedule x, subsets of *anchored jobs* are defined as subsets whose starting times in x can be guaranteed whatever the realization of processing times in the uncertainty set. Each job is associated with an anchoring weight, and the Anchor-Robust Project Scheduling Problem (AnchRob) is to find a baseline schedule x and an anchored subset H, so that the total weight of H is maximized.

The (AnchRob) problem was introduced in (Bendotti *et al.* 2019) as a tool for achieving a trade-off between robust-static project scheduling – where a schedule is computed for the worst-case processing times – and the adaptive robust project scheduling problem studied in (Minoux 2008) – where a makespan is computed, but no baseline schedule. (AnchRob) produces a baseline schedule whose makespan is controlled by the deadline M, and the adaptiveness of the solution is controlled through anchored jobs. (AnchRob) was studied in the case of budgeted uncertainty, where at most Γ processing times may deviate from their nominal values at a time. It was proven NP-hard and a MIP formulation was provided: it has a polynomial number of variables and constraints, but it has a poor linear relaxation value due to "bigM" values. Instances up to 200 jobs were solved in up to 4 minutes using this formulation.

Contributions. The present work investigates improved MIP formulations for (AnchRob). We introduce two MIP formulations, that are valid for a wider class of uncertainty sets beyond budgeted uncertainty. Two types of decisions variables are considered: for each job $i \in J$ an anchoring variable $h_i \in \{0, 1\}$ to indicate whether job i is anchored, for each job $i \in \bar{J}$ a continuous variable $z_i \geq 0$ for the starting time of job i in the baseline schedule. The first formulation $\mathcal{F}_{zh}$ uses both types of variables, and it has polynomial size.

The second formulation $\mathcal{F}_h$ uses only anchoring variables. It has an exponential number of constraints for which separation algorithms are proposed. We also prove that $\mathcal{F}_{zh}$ can be projected explicitely on anchoring variables, thus yielding an alternative formulation in anchoring variables only. We establish that formulations $\mathcal{F}_{zh}$ and $\mathcal{F}_h$ are not comparable, and that they both give a better bound than the MIP formulation from (Bendotti *et al.* 2019). From this analysis, we design a customized MIP formulation that takes advantage of $\mathcal{F}_h$ and $\mathcal{F}_{zh}$. In particular polyhedral properties are used to improve the separation of inequalities from $\mathcal{F}_h$ and devise an efficient Branch-and-Cut algorithm.

For the sake of brevity all proofs are omitted.

2 The Anchor-Robust Project Scheduling Problem

Let us define more formally the Anchor-Robust Project Scheduling Problem. A set of jobs J must be scheduled while respecting precedence constraints, represented by a directed acyclic graph G. The vertex-set of G is $\bar{J} = J \cup \{s, t\}$ where s (resp. t) is a source (resp. sink) representing the beginning (resp. the end) of the schedule. Each job $i \in J$ has a processing time $p_i \in \mathbb{R}_+$, and $p_s = 0$ by convention. Given a vector $p \in \mathbb{R}^J$, let $G(p)$ be the weighted digraph obtained from G by weighting every arc (i, j) with p_i. A schedule of $G(p)$ is a vector of starting times $x \in \mathbb{R}_+^{\bar{J}}$ such that $x_j - x_i \geq p_i$ for every arc (i, j) of $G(p)$. We consider a 2-stage framework where the processing times of jobs are uncertain. In first stage, the decision maker has an instance $G(p)$ to solve, called *baseline instance*, where p is the nominal value of processing times. The project is given a deadline $M \geq 0$. A *baseline schedule* is a schedule of $G(p)$ with makespan at most M. In second stage, the instance will be some $G(p + \delta)$, where the vector $\delta \in \mathbb{R}_+^J$ is a *disruption*. In general a baseline schedule will not remain feasible in second stage. As done classically in robust optimization, we consider that the decision maker wants to hedge against a collection $(G(p + \delta))_{\delta \in \varDelta}$ of second-stage instances, where $\varDelta \subseteq \mathbb{R}_+^J$ is the *uncertainty set*. Given a baseline schedule x^0, a subset H of jobs is *anchored* with respect to x^0 if for every $\delta \in \varDelta$, the baseline schedule can be repaired into a feasible solution without changing starting times of jobs in H, i.e., for every $\delta \in \varDelta$ there exists a schedule x^δ of $G(p + \delta)$ such that $x_i^0 = x_i^\delta$ for every $i \in H$. Finally, each job $i \in J$ is associated with an *anchoring weight* $a_i \in \mathbb{R}_+$. (AnchRob) is then to find a baseline schedule x^0 and subset of jobs H anchored w.r.t. x^0, so that the total weight of anchored jobs $\sum_{i \in H} a_i$ is maximized.

Let us now recall a characterization of anchored jobs from (Bendotti *et al.* 2019) that will be used in the sequel. Given $i, j \in \bar{J}$ and $\delta \in \mathbb{R}_+^J$ let $L_\delta(i, j)$ denote the length of the longest i–j path in $G(p + \delta)$ (and $-\infty$ if there is no such path). In particular, $L_0(i, j)$ is the length of the longest i–j path in $G(p)$. Let $L_\varDelta(i, j)$ denote $\max_{\delta \in \varDelta} L_\delta(i, j)$.

Proposition 1. *(Bendotti* et al. *2019) Let x be a schedule of $G(p)$. A set H is anchored w.r.t. x iff $x_j - x_i \geq L_\varDelta(i, j)$ for every $i \in H \cup \{s\}$, $j \in H$.*

In this work, the set $\varDelta$ is supposed to be a subset of $\mathbb{R}_+^J$ such that: for every $\delta \in \varDelta$, for every $J' \subseteq J$, the vector δ' defined by $\delta'_i = \delta_i$ if $i \in J'$, and $\delta_i = 0$ otherwise is still an element of $\varDelta$. We consider that the values $L_\varDelta(i, j)$ for every $i, j \in \bar{J}$ are known and given as input. No other information on the uncertainty set $\varDelta$ is required. The values $L_\varDelta(i, j)$ will appear explicitely in the constraints of the proposed mixed-integer formulations. The computation of the $L_\varDelta(i, j)$ values can be done in polynomial time for some uncertainty sets, such as: budgeted uncertainty sets (Bertsimas and Sim 2004), uncertainty sets defined by a polynomial number of scenarii, and their convex hulls. It can also be done with a pseudo-polynomial algorithm for uncertainty sets with several uncertainty budget constraints introduced in (Minoux 2007). When the computation of the $L_\varDelta(i, j)$ values can be done in a preprocessing step, then the proposed algorithmic framework can be applied using the $L_\Delta(i,j)$ values.

3 Two MIP formulations

3.1 A compact MIP formulation with anchoring and schedule variables

Theorem 1. *A valid formulation for (AnchRob) is*

$$
\begin{array}{lll}
(\mathcal{F}_{zh}) & \max \sum_{i\in J} a_i h_i & \\
& s.t.\ z_j - z_i \geq L_0(i,j) + (L_\Delta(i,j) - L_0(i,j))\, h_j & \forall i \in \bar{J}, j \in \bar{J} \setminus \{t\} \\
& z_t - z_i \geq L_0(i,t) & \forall i \in J \\
& z_t \leq M & \\
& z_j \geq 0 & \forall j \in \bar{J} \\
& h_j \in \{0,1\} & \forall j \in J
\end{array}
$$

Note first that for a feasible pair (z,h) of $\mathcal{F}_{zh}$, the set H associated with h is anchored w.r.t. z, thanks to Proposition 1. The proof of the validity of formulation $\mathcal{F}_{zh}$ relies on a dominance argument: we prove that for any set H anchored w.r.t. some baseline schedule, there exists a baseline schedule z for which H is anchored and (z,h) is feasible for $\mathcal{F}_{zh}$. Note also that this formulation has a polynomial number of variables and constraints.

3.2 An exponential formulation with anchoring variables only

Let $\overline{G}$ be a weighted graph defined as the transitive closure of G, where every arc (i,j) has arc-length $L_\Delta(i,j)$ if $j \neq t$, and $L_0(i,t)$ otherwise. Let $\mathscr{P}$ (resp. $\mathscr{P}^{>M}$) denote the set of s–t paths of $\overline{G}$ (resp. the set of s–t paths of $\overline{G}$ that have length $> M$). Given $P \in \mathscr{P}$, let $V(P)$ denote the jobs of J along path P. We show the following result.

Theorem 2. *A valid formulation for (AnchRob) is*

$$
\begin{array}{llll}
(\mathcal{F}_h) & \max \sum_{i\in J} a_i h_i & & \\
& s.t.\ \sum_{i\in V(P)} h_i \leq |V(P)| - 1 & \forall P \in \mathscr{P}^{>M} & \textit{(PathCov)} \\
& h_j \in \{0,1\} & \forall j \in J &
\end{array}
$$

The family of *path covering* inequalities (PathCov) imposes that there is at least one non-anchored job along any path of $\mathscr{P}^{>M}$. Note that it is necessary: if all jobs along the path were anchored, then with Proposition 1 every associated baseline schedule would have makespan $> M$. Formulation $\mathcal{F}_h$ has an exponential number of constraints, thus we study the separation of (PathCov) inequalities. The separation problem is a constrained longest path problem in $\overline{G}$, that is, the problem of finding a path with length $> M$, and sum of $1 - h_i$ over vertices at least 1. Formally, we show that

Theorem 3. *Separation of (PathCov) is weakly NP-hard and admits a pseudo-polynomial algorithm based on dynamic programming. Separation of (PathCov) is polynomial-time solvable in an integer point $h \in \{0,1\}^J$.*

We mention that, given $h \in \{0,1\}^J$ feasible for $\mathcal{F}_h$, a baseline schedule for which the corresponding set H is anchored can be found in polynomial time. Indeed it is sufficient to compute an earliest schedule that satisfies the precedence constraints from $G(p)$ and the precedence constraints from Proposition 1. In particular, this justifies that we obtained a valid formulation for (AnchRob) with only h variables.

4 Algorithmic framework

4.1 Comparison of formulations for budgeted uncertainty

Consider budgeted uncertainty. Then three formulations for (AnchRob) are available: formulations $\mathcal{F}_{zh}$ and $\mathcal{F}_{h}$, and the formulation from (Bendotti *et al.* 2019) denoted by $\mathcal{F}_{xh}$. It can be shown that the optimal values of the linear relaxation of $\mathcal{F}_{zh}$ and $\mathcal{F}_{h}$ are always better than the optimal value of the linear relaxation of $\mathcal{F}_{xh}$. We thus focused on $\mathcal{F}_{zh}$ and $\mathcal{F}_{h}$; it appears that the two formulations cannot be compared w.r.t. their linear relaxations.

Theorem 4. *$\mathcal{F}_{h}$ and $\mathcal{F}_{zh}$ are not comparable, i.e., there exists instances where the optimal value of the linear relaxation of $\mathcal{F}_{h}$ is better than the optimal value of the linear relaxation of $\mathcal{F}_{zh}$, and vice versa.*

We also provide an explicit formulation of the projection of formulations $\mathcal{F}_{zh}$ and $\mathcal{F}_{xh}$ on the space of anchoring variables. E.g., for $\mathcal{F}_{zh}$ formulation:

Proposition 2. *Let $h \in \{0,1\}^J$. There exists z such that (z,h) is feasible for $\mathcal{F}_{zh}$ if and only if h satisfies the inequalities $\sum_{(i,j)\in P} L_0(i,j) + (L_\Delta(i,j) - L_0(i,j))h_j \leq M \ \forall P \in \mathscr{P}$.*

4.2 Dedicated Branch-And-Cut algorithm

Theorem 4 suggests the use of a combination of $\mathcal{F}_{h}$ and $\mathcal{F}_{zh}$ to solve efficiently larger instances of (AnchRob). We formulate the problem with z and h variables. All constraints from $\mathcal{F}_{zh}$ are enforced in a static way. (PathCov) inequalities from $\mathcal{F}_{h}$ are separated in a heuristic way to strengthen the formulation. Namely, we only separate (PathCov) inequalities in integer points, thus in polynomial time.

Additional features are considered to improve the efficiency of separated (PathCov) inequalities, relying on polyhedral considerations. In particular, it can be shown that if inequality associated with path $P \in \mathscr{P}^{>M}$ is facet-defining, then it must satisfy: for any $i \in P$, the path $P' := P \setminus \{i\}$ is not an element of $\mathscr{P}^{>M}$. We enforce this property during the separation process: an inequality (PathCov) is separated, then vertices are removed from the path until the property is satisfied. We will give numerical results to illustrate the relevance of the proposed Branch-And-Cut algorithm.

References

P. Bendotti, Ph. Chrétienne, P. Fouilhoux, A. Quillot, 2017, “Anchored reactive and proactive solutions to the CPM-scheduling problem", *European Journal of Operational Research*, Vol. 261, pp. 67–74.

P. Bendotti, Ph. Chrétienne, P. Fouilhoux, A. Pass-Lanneau, 2019, “The Anchor-Robust Project Scheduling Problem", *Preprint*, `https://hal.archives-ouvertes.fr/hal-02144834`.

D. Bertsimas and M. Sim, 2004, “The Price of Robustness", *Operations Research*, Vol. 52, pp. 35–53.

M. Minoux, 2007, “Models and Algorithms for Robust PERT Scheduling with Time-Dependent Task Durations". *Vietnam Journal of Mathematics*, Vol. 35, pp 387–398.

M. Minoux, 2008, “Robust linear programming with right-handside uncertainty, duality and application", *Encyclopedia of Optimization*, Springer, pp. 3317–3327.

Search space reduction in MILP approaches for the robust balancing of transfer lines

Aleksandr Pirogov[1,2], André Rossi[3], Evgeny Gurevsky[4], and Alexandre Dolgui[1]

[1] LS2N, IMT Atlantique, Nantes, France
[2] CENTRAL, Lancaster University, UK
[3] LAMSADE, Université Paris-Dauphine (PSL), France
[4] LS2N, Université de Nantes, France

Keywords: transfer line, balancing, robustness, MILP, model enhancements, assignment intervals.

1 Problem description

This abstract deals with the transfer line balancing problem which consists in distributing n non-preemptive production tasks among m linearly ordered machines linked by a conveyor belt such that the load of any machine does not exceed the fixed cycle time T, and precedence constraints are satisfied. At each machine, the corresponding tasks are allocated to blocks, where the tasks are executed simultaneously. Thus, the working time of the block is equal to the longest processing time among the tasks allocated to it. Up to $r_{\max}$ tasks can be attributed to the same block and at most $b_{\max}$ blocks may be arranged at one machine. The blocks of the same machine are activated sequentially. As a consequence, the load time of the machine is equal to the sum of the working time of its blocks. We denote by $U = \{1, \ldots, mb_{\max}\}$ the set of all possible blocks, and by $U(p) = \{(p-1)b_{\max}+1, \ldots, pb_{\max}\}$ the set of blocks for any machine $p \in W = \{1, \ldots, m\}$.

The nominal processing time of task j is t_j for any $j \in V = \{1, \ldots, n\}$. A given nonempty subset $\widetilde{V} \subseteq V$ of tasks is the set of *uncertain tasks*, *i.e.*, the set of tasks whose processing time may vary and can be larger than t_j. The tasks in $V \backslash \widetilde{V}$ are called *certain tasks* and its processing time remains deterministic.

The set of tasks allocated to blocks for a given number of machines and satisfying the mentioned above constraints forms a so-called *feasible line configuration.* For each such configuration, we study a specific robustness measure, named *stability radius.* It is calculated as the maximum increase of the nominal processing time that may affect any uncertain task without compromising the feasibility of the corresponding line configuration by breaking the cycle time constraint. The problem, denoted by P_1 and studied in this abstract, seeks naturally a line configuration, which possesses the maximal value of its stability radius.

The first mixed-integer linear programming (MILP) formulation of P_1 is due to Pirogov (2019). This MILP is given in the next section. In order to solve it more efficiently, we propose some improvements in that formulation, as well as tighter assignment intervals for the tasks. Indeed, because of the precedence constraints, each task j has an assignment interval $[l_j, u_j]$ of indices of blocks which are available to perform this task.

2 Initial MILP formulation

P_1 has originally been formulated as a MILP on the following decision variables: ρ_1 is the stability radius value to maximize; $x_{j,k}$ is a binary variable that is set to one if and only if the task j is allocated to the block k; y_k is equal to 1 if the block k is not empty and 0, otherwise; $\tau_k \geq 0$ determines the working time of the block k; $\Delta^{(p)}_{\min} \geq 0$ represents the minimal value of the save time among all the blocks arranged at the machine p. The save time is defined only for the blocks having uncertain tasks. It is calculated as the difference between the nominal processing time of the longest uncertain task allocated to it and its working time; a_p is a non-negative variable, which is positive if the machine p processes at least one uncertain task; z_k is set to 1 if an uncertain task is allocated to the block k and 0, otherwise. The central idea of the MILP formulation for P_1 consists in maximizing ρ_1, expressed as the minimum idle time over all the machines that process uncertain tasks.

$$\text{Maximise } \rho_1$$

$$\sum_{k\in U} x_{j,k} = 1 \qquad \forall j \in V \tag{1}$$

$$\sum_{j\in V} x_{j,k} \le r_{\max} \qquad \forall k \in U \tag{2}$$

$$x_{j,k} \le y_k \qquad \forall k \in U,\ \forall j \in V \tag{3}$$

$$y_k \le \sum_{j\in V} x_{j,k} \qquad \forall k \in U \tag{4}$$

$$y_{k+1} \le y_k \qquad \forall p \in W,\ \forall k \in U(p) \setminus \{pb_{\max}\} \tag{5}$$

$$t_j \cdot x_{j,k} \le \tau_k \qquad \forall k \in U,\ \forall j \in V \tag{6}$$

$$\Delta^{(p)}_{\min} \le T \cdot (2 - y_k - z_k) + \tau_k - t_j \cdot x_{j,k} \qquad \forall p \in W,\ \forall k \in U(p),\ \forall j \in \widetilde{V} \tag{7}$$

$$x_{j,k} \le z_k \qquad \forall k \in U,\ \forall j \in \widetilde{V} \tag{8}$$

$$\sum_{k\in U(p)} \tau_k \le T \qquad \forall p \in W \tag{9}$$

$$\sum_{q=k}^{|U|-1} x_{i,q} \le \sum_{q=k+1}^{|U|} x_{j,q} \qquad \forall (i,j) \in A,\ \forall k \in U \setminus \{mb_{\max}\} \tag{10}$$

$$x_{j,k} \le a_p \qquad \forall p \in W,\ \forall k \in U(p),\ \forall j \in \widetilde{V} \tag{11}$$

$$\rho_1 \le T \cdot (2 - a_p) - \sum_{k\in U(p)} \tau_k + \Delta^{(p)}_{\min} \qquad \forall p \in W \tag{12}$$

$$\rho_1 \ge 0$$

$$\Delta^{(p)}_{\min} \ge 0,\ a_p \ge 0 \qquad \forall p \in W$$

$$x_{j,k} \in \{0,1\} \qquad \forall j \in V,\ \forall k \in U$$

$$\tau_k \ge 0,\ z_k \in \{0,1\},\ y_k \in \{0,1\} \qquad \forall k \in U$$

Constraints (1) ensure that each task is allocated to exactly one block. Inequalities (2) enforce that each block contains at most $r_{\max}$ tasks. Any block having at least one assigned task is considered as non-empty, as enforced by (3) and (4). Constraints (5) ensures that block $k+1$ has to be empty if block k is empty. The working time of the block is not less than the processing time of any task allocated to it, as provided by (6). Constraints (7) express the definition of $\Delta^{(p)}_{\min}$. Inequalities (8) ensure that z_k is set to one if block k processes an uncertain task. Constraints (9) state that the load of any machine cannot exceed the cycle time, and the precedence constraints are enforced by inequalities (10), where A is the set of all the pairs of tasks involved in the precedence constraints. Constraints (11) – (12) implies that a_p is strictly positive if machine p has at least one uncertain task, and zero otherwise.

3 Reduction of the assignment interval of tasks

Initially, all the tasks have an assignment interval equal to $[1, mb_{\max}]$, but the precedence constraints can help reducing them, which allows to set some $x_{j,k}$ decision variables to 0 in the MILP formulation of P_1. This is achieved by computing the earliest completion time $\theta_j^{(EC)}$ and the latest starting time $\theta_j^{(LS)}$ of task $j \in V$ with the following induction formula initialized with $\theta_0^{(EC)} = 0$ and $\theta_{n+1}^{(LS)} = m \cdot T$ (tasks 0 and $n+1$ are the dummy start and end of the schedule):

$$\theta_j^{(EC)} = t_j + \max_{q \in P(j)} \max \left\{ \theta_q^{(EC)}, \left(\left\lceil \frac{\theta_q^{(EC)} + t_j}{T} \right\rceil - 1 \right) \cdot T \right\},$$

$$\theta_j^{(LS)} = \min_{q \in S(j)} \min \left\{ \theta_q^{(LS)}, \left(1 + \left\lfloor \frac{\theta_q^{(LS)} - t_j}{T} \right\rfloor \right) \cdot T \right\} - t_j.$$

Here, $P(j)$ (resp. $S(j)$) is the set of direct predecessors (resp. successors) of j in the precedence graph. From $\theta_j^{(EC)}$ and $\theta_j^{(LS)}$, the lower and upper bounds of the assignment interval of task j, denoted by l_j and u_j, can be derived:

$$l_j = \left(\left\lceil \frac{\theta_j^{(EC)}}{T} \right\rceil - 1 \right) \cdot b_{\max} + 1, \quad u_j = \left(1 + \left\lfloor \frac{\theta_j^{(LS)}}{T} \right\rfloor \right) \cdot b_{\max}.$$

Since no task can be assigned to the same block as its predecessors or successors, the first rule is to apply the following formula as long as it brings improvements over current bound values:

$$l_j = \max \left\{ l_j, \max_{q \in P(j)} l_q + 1 \right\}, \quad u_j = \min \left\{ u_j, \min_{q \in S(j)} u_q - 1 \right\}.$$

The second rule is to compute $b_{\max}^{(p)} \leq b_{\max}$, an upper bound on the number of non-empty blocks at machine p. Minimizing $b_{\max}^{(p)}$ permits to find empty blocks and allows to set many decision variables to zero. Because of space limitation, this rule is not presented.

Finally, based on a set of tasks that have to be processed (resp. can be possibly processed) by the machine p, noted as $D(p)$ (resp. $V(p)$), the third rule is to assess the maximum remaining working time of the machine p, for all $p \in W$. If $|D(p)| = r_{\max} \cdot b_{\max}^{(p)}$, then no task in $V(p) \backslash D(p)$ can be assigned to the machine p. If $|D(p)| < r_{\max} \cdot b_{\max}^{(p)}$, then the remaining working time of the machine p is upper bounded by $r_{\max} \cdot T - \sum_{j \in D(p)} t_j$. Hence, any task in $V(p) \backslash D(p)$ whose duration is strictly larger than $r_{\max} \cdot T - \sum_{j \in D(p)} t_j$ should be removed from $V(p)$.

4 Improvement of the MILP formulation

From the previous section, variable $x_{j,k}$ is set to zero for all $j \in V$ and for all $k \notin [l_j, u_j]$. Similarly, if $V(p) \subset \widetilde{V}$, then a_p is set to 1.

The following valid inequalities are added to link the y_k and z_k variables (if a block processes an uncertain task, it should be open):

$$z_k \leq y_k, \ \forall k \in U(p), \ \forall p \in W.$$

Constraints (7) can be reinforced by replacing $T(2 - y_k - z_k)$ with $T(1 - z_k)$. In addition, T can be replaced by the constant $\Delta_{\max}^{(p)}$, which is an upper bound on the save time of any block in the machine p:

$$\varDelta_{\max}^{(p)} = \begin{cases} 0, & \text{if } V(p) \cap (V \backslash \widetilde{V}) = V(p) \text{ or } V(p) \cap \widetilde{V} = V(p), \\ 0, & \text{else if } t_{\max}^{(p)} \leq \widetilde{t}_{\min}^{(p)}, \\ t_{\max}^{(p)} - \widetilde{t}_{\min}^{(p)}, & \text{otherwise.} \end{cases}$$

Here, $t_{\max}^{(p)}$ is the maximum processing time among certain tasks that can be processed by machine p, whereas $\widetilde{t}_{\min}^{(p)}$ is the minimum processing time among uncertain tasks that can be processed by machine p: $t_{\max}^{(p)} = \max\limits_{j \in V(p) \cap (V \backslash \widetilde{V})} t_j$ and $\widetilde{t}_{\min}^{(p)} = \min\limits_{j \in V(p) \cap \widetilde{V}} t_j$.

Indeed, when machine p can only process certain tasks, there is no save time, so $\varDelta_{\max}^{(p)}$ has to be set to 0. When machine p can only process uncertain tasks (or when certain tasks are shorter than any uncertain task), $\varDelta_{\min}^{(p)}$ is zero, so $\varDelta_{\max}^{(p)}$ can also be set to 0. In all other cases, the save time is upper bounded by the difference between the longest certain processing time, and the shortest uncertain processing time. Hence, constraints (7) are replaced with:

$$\varDelta_{\min}^{(p)} \leq \varDelta_{\max}^{(p)} \cdot (1 - z_k) + \tau_k - t_j \cdot x_{j,k}, \quad \forall p \in W, \ \forall k \in U(p), \ \forall j \in \widetilde{V}.$$

Constraints (11) can be strengthened to:

$$\sum_{k \in U(p)} x_{j,k} \leq a_p, \quad \forall p \in W, \ \forall j \in \widetilde{V}.$$

The following constraints state that the processing time of an open block cannot be less than the processing time of the shortest task that can be part of this block.

$$\min_{j \in V(p)} t_j \cdot y_k \leq \tau_k, \ \forall p \in W, \ \forall k \in U(p).$$

The following inequalities declare that if a machine processes an uncertain task, then at least one its block has to accommodate an uncertain task:

$$a_p \leq \sum_{k \in U(p)} z_k, \ \forall p \in W.$$

And finally, all the a_p variables should be declared as binary.

5 Conclusion

When applying the improvements proposed in this paper, 829 instances out of 900 from Pirogov (2019) have been solved to optimality within the time limit of 600 seconds per instance. Originally, only 467 instances were solved to optimality with the initial model. These ideas may be applied to address another problem version, where the stability radius is based on a different metric.

Acknowledgements

This work was partially supported by the council of the French Région «Pays de la Loire».

References

Pirogov A., 2019, «Robust balancing of production lines: MILP models and reduction rules». *Ph.D. Thesis, IMT Atlantique, Nantes, France.*

An Inclusion-Exclusion based algorithm for the permutation flowshop scheduling problem

Olivier Ploton[1], Vincent T'kindt[1]

Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée (LIFAT, EA 6300), ERL CNRS 7002 ROOT, Tours, France
{olivier.ploton,vincent.tkindt}@univ-tours.fr

Keywords: flowshop, exponential algorithms, Inclusion-Exclusion.

1 Introduction

In this paper we are interested in minimizing the makespan of a permutation flowshop schedule. Following the notation of Graham et al. (1979), this problem is denoted by $F|prmu|C_{\max}$. In this problem, there are n jobs to be scheduled on m machines. Each job must be processed on machines 1 to m, in this order, and each machine can process only one job at a time. All machines must process jobs in the same order, and a schedule is essentially defined by this order. We note O_{ij}, $i \in \{1 \dots n\}$, $j \in \{1 \dots m\}$, the operation of job i on machine j, which has a non-negative integer processing time p_{ij}. For any given schedule, we define C_{ij} as the completion time of O_{ij} in this schedule. The makespan is the maximum completion time $C_{\max} = \max_{1 \leq i \leq n} C_{im}$. The objective is to find an optimal solution which minimizes the makespan.

We focus on the time and space worst-case complexities of algorithms to solve the $Fm|prmu|C_{\max}$ problem, i.e. when the number of machines is a parameter of the instance. The size of an instance $\mathcal{I}$ is the number of jobs n. The measure of an instance is the sum of its processing times, i.e. $||\mathcal{I}|| = \sum_{i,j} p_{ij}$.

Many algorithms use the branch-and-bound technique, along with specific optimizations. The bounding functions used in these branch-and-bound algorithms are more and more precise as time goes, and some of these algorithms have the best known practical performances (Ladhari and Haouari 2005, Ritt 2016, Gmys et al. 2020). While efficient in practice, they often have a worst-case time complexity bound comparable to the $O^*(n!)$ complexity of the brute-force algorithm.

From a theoretical point of view, few algorithms have been proposed in order to provide better worst-case complexity bounds. Jansen et al. (2013) describe a very general algorithm class, based on a dynamic programming technique on (unordered) sets of jobs. They get time and space worst-case complexities with respect to the number of operations $m \times n$, which translates into $O^*(2^{O(n)}||\mathcal{I}||^{O(1)})$ for each fixed number of machines. Shang et al. (2018) give a more precise result in the particular case of the $F3|prmu|C_{\max}$ problem, by a fine analysis of the number of critical paths in a schedule. They obtain time and space complexities in $O^*(2^n||\mathcal{I}||)$.

Our main contribution in this paper is an algorithm which, for any fixed number of machines, runs with a moderate exponential worst-case time complexity and requires only pseudopolynomial space. More precisely, the time complexity bound is in $O^*(2^n||\mathcal{I}||^m)$ and the space complexity bound is in $O^*(||\mathcal{I}||^m)$.

The algorithms we describe use the Inclusion-Exclusion technique. This rather old combinatorics formula received a pioneer application to computer science by Karp (1982) and Bax (1993). More recently, Inclusion-Exclusion gained in popularity in operational research (Björklund and Husfeldt 2006, Koivisto 2006). Nederlof (2013) showed the interest of this technique to get polynomial or pseudopolynomial space and moderate exponential time algorithms. Yet, the Inclusion-Exclusion technique is not widely used for scheduling algorithms. To the best of our knowledge, the only scheduling problem solved by an Inclusion-Exclusion based algorithm is the $1|r_i, \tilde{d_i}|$-problem (Karp 1982, Nederlof 2008). Some scheduling algorithms can be reduced to well-known classic problems solved by an Inclusion-Exclusion algorithm, e.g. the $P||C_{\max}$ problem reduces to the bin-packing problem (Karp 1982) and the $F|nowait|C_{\max}$ problem reduces to the Asymmetric Traveling Salesman Problem (Bax 1993, Karp 1982).

2 The permutation flowshop decision problem

As the makespan is a regular objective, we can restrict to semi-active schedules, where no operation could be scheduled earlier without changing the job order. So, a schedule S is uniquely represented by the sequence of its jobs: $S = (i_1 \dots i_n)$.

Figure 1 presents an annotated Gantt chart showing a solution of the permutation flowshop problem (one line per machine, one color per job).

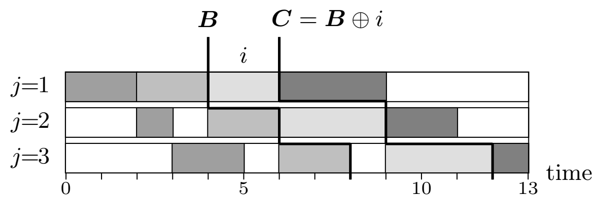


Fig. 1. A permutation schedule

Before executing job i, machines must be done with previous jobs. So, it is useful to consider a more precise version of the problem: machines have release times $(B_j)_{j=1\dots m}$ before which they are busy. Together, they form a time front $\boldsymbol{B} = (B_j)$. We denote by $\boldsymbol{C} = \boldsymbol{B} \oplus i$ the completion times (C_j) of job i on machine j when executed with release times (B_j). These are also the release times of the next job to be processed. With the convention that $C_0 = 0$, we have:

$$C_j = \max(C_{j-1}, B_j) + p_{ij}, \quad \forall\, j = 1 \dots m. \tag{1}$$

Let I be a set of jobs, $\boldsymbol{B}$ a release time front and ε a given threshold on the makespan value. We now want to determine whether or not there exists a schedule using jobs of I, whose makespan is at most ε when run with release times $\boldsymbol{B}$. For that, we shall count the number $N(I, \boldsymbol{B}, \varepsilon)$ of such schedules. We are about to do it using Inclusion-Exclusion.

3 The Inclusion-Exclusion principle

To apply the Inclusion-Exclusion principle, following Fomin and Kratsch (2010), consider a problem of size n in which a solution is represented by a permutation. Relax this problem by allowing any list of length n, with possible duplicates or missing elements. For any $J \subset I$, count valid lists using only elements of J. By Inclusion-Exclusion, we deduce the number of solutions of the initial problem.

We apply this principle to count schedules viewed as lists of jobs. We denote by $\mathfrak{S}_n$ the set of permutations schedules, where all jobs appear once, and I^n is the set of relaxed schedules, where there may be duplicate or missing jobs. Define $E \subset I^n$ as the set of relaxed schedules whose makespan is at most ε when run with release times $\boldsymbol{B}$. We derive:

$$\underbrace{\operatorname{card} E \cap \mathfrak{S}_n}_{N(I,\boldsymbol{B},\varepsilon)} = \sum_{J \subset I} (-1)^{|I|-|J|} \underbrace{\operatorname{card} E \cap J^n}_{N_J(\boldsymbol{B},\varepsilon)} \tag{2}$$

We determine each term $N_J(\boldsymbol{B},\varepsilon)$ by dynamic programming. We compute $N_{J,\varepsilon}[\ell,\boldsymbol{B}]$ as the number of relaxed schedules of length ℓ, using only jobs of J, whose makespans are at most ε when run with release times $\boldsymbol{B}$. We have:

$$N_J(\boldsymbol{B},\varepsilon) = N_{J,\varepsilon}[n,\boldsymbol{B}] \tag{3}$$

$$N_{J,\varepsilon}[\ell,\boldsymbol{B}] = \sum_{i \in J} N_{J,\varepsilon}[\ell-1,\boldsymbol{B}\oplus i] \text{ if } B_m \leq \varepsilon, \text{ 0 otherwise} \qquad \forall\, \ell = 1 \dots n \tag{4}$$

$$N_{J,\varepsilon}[0,\boldsymbol{B}] = 1 \text{ if } B_m \leq \varepsilon, \text{ 0 otherwise.} \tag{5}$$

4 From a feasible makespan value to an explicit solution

The optimal makespan can be deduced from the decision problem. It is:

$$C_{\max}^{\text{opt}} = \min\{\varepsilon \mid N(I,(B_j{=}0),\varepsilon) > 0\} \tag{6}$$

It can be computed by using a dichotomic search.

Once the optimal makespan is known, we can determine an explicit solution step by step. We write (Algorithm 1) the recursive function *Solution*$(\sigma,\boldsymbol{B},I)$, where σ is the sequence of already scheduled jobs, $\boldsymbol{B} = (B_j)_j$ are the completion times of already scheduled jobs, and I is the set of jobs to be scheduled after $\boldsymbol{B}$. We use the decision algorithm as an oracle to systematically choose a job outside σ leading to a feasible solution. We denote by $()$ the empty sequence and by $\cdot$ the concatenation of sequences. We obtain an optimal solution of the $Fm|prmu|C_{\max}$ problem by calling *Solution*$(\sigma{=}(),(B_j{=}0),I{=}\{1\dots n\})$.

```
Function Solution($\sigma$, $\boldsymbol{B}$, $I$):
    if $I \neq \emptyset$ then
        for $i \in I$ do
            if $N(I \setminus \{i\}, \boldsymbol{B} \oplus i, C_{\max}^{\text{opt}}) > 0$ then
                return Solution($\sigma \cdot i$, $\boldsymbol{B} \oplus i$, $I \setminus \{i\}$)
    else
        return $\sigma$
```

Algorithm 1: Computation of a feasible schedule

5 Worst-case complexities

We now evaluate the worst-case time and space complexities of our algorithms:

- Each component of the timefront $\boldsymbol{B}$ involved in the dynamic programming equations (3), (4), (5) is bounded by the sum of the processing times, i.e. $||\mathcal{I}||$. So, the number of involved states is in $O^*(||\mathcal{I}||^m)$.
- The decision problem uses 2^n independent dynamic programming computations and it can be solved in $O^*(2^n||\mathcal{I}||^m)$ time $O^*(||\mathcal{I}||^m)$ space.
- Dichotomic computation of $C_{\max}^{\text{opt}}$ is in $O^*(2^n||\mathcal{I}||^m \log||\mathcal{I}||)$ time and $O^*(||\mathcal{I}||^m)$ space.
- When $C_{\max}^{\text{opt}}$ is known, computation of an optimal solution of the $Fm|prmu|C_{\max}$ problem is in $O^*(2^n||\mathcal{I}||^m)$ time and $O^*(||\mathcal{I}||^m)$ space.
- The global algorithm is in $O^*(2^n||\mathcal{I}||^m \log||\mathcal{I}||)$ time and $O^*(||\mathcal{I}||^m)$ space.

6 Conclusions

In this paper, we study exact algorithms to minimize the makespan of permutation flowshop schedules, and we focus on bounding worst-case time and space complexities. These complexities are evaluated for a fixed number of machines m using job number n as the instance size and the sum of the processing times as an instance measure $||\mathcal{I}||$. The best general time and space complexity bounds proved so far is due to Jansen et al. (2013). It is $O^*(2^{O(n)}||\mathcal{I}||^{O(1)})$, for each fixed m. Shang et al. (2018) proved a more precise bound of $O^*(2^n||\mathcal{I}||)$, for the particular case of 3 machines.

We describe an Inclusion-Exclusion based algorithm for the $Fm|prmu|C_{\max}$ problem, using dynamic programming for enumerations. We prove that, for every fixed m, its worst-case space complexity is pseudopolynomial, with bound $O^*(||\mathcal{I}||^m)$, and its worst-case time complexity is moderately exponential, with bound $O^*(2^n||\mathcal{I}||^m)$.

From this piece of research, several future research directions can be outlined: how to optimize the computation of the sum involved in the Inclusion-Exclusion principle ? How to get tighter bounds on the number of states used in the dynamic programming algorithm ? These questions are of great importance to derive better worst-case complexity bounds.

References

Bax E.T., 1993, “Inclusion and exclusion algorithm for the Hamiltonian path problem”, *Information Processing Letters*, Vol 17(4), pp 203–207.

Björklund A., T. Husfeldt, 2006, “Inclusion-exclusion algorithms for counting set partitions”, *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, pp 575–582.

Fomin F.V., D. Kratsch, 2010, “Exact exponential algorithms”, Springer.

Gmys J., M. Mezmaz, N. Melab, D. Tuyttens, 2020, “A computationally efficient Branch-and-Bound algorithm for the permutation flow-shop scheduling problem”, *European Journal of Operational Research*, Vol 284(3), pp 814–833.

Graham R.L., E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, 1979, “Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey”, *Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications*, Vol 5, pp 287–326.

Jansen K., F. Land, K. Land, 2013, “Bounding the Running Time of Algorithms for Scheduling and Packing Problems”, *Algorithms and Data Structures - 13th International Symposium*, pp 439–450.

Karp R.M., 1982, “Dynamic Processing meets the principle of inclusion and exclusion”, *Operational Research Letters*, Vol 1(2), pp 49–51.

Koivisto M., 2006, “An $O(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion-exclusion”, *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, pp 583–590.

Ladhari T., M. Haouari, 2005, “A computational study of the permutation flow shop problem based on a tight lower bound”, *Computers & Operations Research*, Vol 32, pp 1831–1847.

Nederlof J., 2008, “Inclusion-exclusion for hard problems”, *Master Thesis*, Utrecht University.

Nederlof J., 2013, “Fast Polynomial-Space Algorithms Using Inclusion-Exclusion”, *Algorithmica*, Vol 65, pp 868–884.

Ritt M., 2016, “A branch-and-bound algorithm with cyclic best-first search for the permutation flow shop scheduling problem”, *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pp 872–877.

Ryser H.J., 1963, “Combinatorial mathematics”, *The Carus Mathematical Monographs, No 14*, The Mathematical Association of America.

Shang L., C. Lenté, M. Liedloff, V. T’kindt, 2018, “Exact exponential algorithms for 3-machine flowshop scheduling problems”, *Journal of Scheduling*, Vol 21, pp 227–233.

Decision trees for robust scheduling

Tom Portoleau[1,2], Christian Artigues[1] and Romain Guillaume[2]

[1] LAAS-CNRS, Universite de Toulouse, CNRS, Toulouse, France
tom.portoleau@laas.fr,artigues@laas.fr
[2] IRIT, Toulouse, France
romain.guillaume@irit.fr

Keywords: proactive-reactive scheduling, robust optimization, decision tree

1 Introduction

In the scheduling literature, approaches that mix a proactive phase aiming at issuing a robust baseline schedule and a reactive phase that adapts the baseline scheduling in case of major disturbances have been widely studied under the proactive-reactive scheduling terminology [3]. Recently Davari and Demeulemeester [2] remarked that in this series of approaches, the proactive and the reactive phases were rather treated separately while they should mutually influence each other. In particular the author propose to associate a reaction cost to the reactive scheduling part. As a matter of fact, over-reacting to disturbance, although generally beneficial for the objective function may lead to a destabilization of the scheduling environment. In this paper, we aim also at addressing this issue together with another one arising with the huge increasing amount of available data. We consider that we have a scheduling problem where a part of parameters is known with precision while another part is known under interval-based scenarios. A robust schedule, e.g. minimizing the min max objective function over the scenario set, can be computed and then followed in run time. During the schedule execution, information becomes regularly available that allows to reduce the uncertainty set and possibly react to this change. However the question arise whether the decision maker should use of not the information, or which part of the information should be used and for which reaction. As already mentioned, over-reacting can be costly but there may be also a cost of getting information, especially when the amount of information that can be obtained is large. We proposed an approach for dealing with the information selection and reaction decision issues, inspired by contingent planning approaches that determines alternative schedules at some events where a high probability of schedule break occur [1]. However we assume that contingent schedules can only be computed at some predetermine decision points, corresponding to particular times such as the end of a work day or a shift change. At each such time point, we select a subset of the information that can be available to partition the set of scenarios. For each element of the partition, we compute a new robust schedule compatible to the one followed up to the decision point. This yields what we call a robust decision tree, that prescribes the path to follow in response to a particular scenario. In the remaining of the paper we present the different models and algorithms we propose as well as experimental results. To illustrate our approach we focus on the simple robust $1||L_{\max}$ model with uncertainty on due dates and knwon processing times.

2 Uncertainty, Information and Robust Decision Tree Models

2.1 Uncertainty model

In practice, it is often easier for a decision-maker to establish bounds over uncertain parameters, like processing times or due dates, than to build an accurate probabilistic model

which often requires a large amount of data. In this paper, we consider interval uncertainty. Given an uncertain parameter x we denote by $X = [x_{min}, x_{max}]$ the interval in which it takes its value. We make no assumption about which probabilistic law is followed by x in this interval. Given a set of uncertain parameters $(x_i)_{i\in I}$ we define the set of possible assignments of parameters by $\Omega = \prod_{i\in I} X_i$ (i.e. it is assumed that there is no correlation between them, all realisation x_i are independent). We call a scenario an assignment of each parameter $x_i, \forall i \in I$ such that $(x_i)_{i\in I} \in \Omega$.
From now on, when ω is a scenario, s a schedule and f the objective , we denote by $f(\omega, s)$ the objective value of s in scenario ω (note that for our application case, $f = L_{max}$).

Example 1 *We consider a small instance of the problem $1||L_{max}$ with three tasks : task 1 : $p_1 = 10$, $d_1 \in D_1 = [10, 11]$, task 2 : $p_2 = 6$, $d_2 \in D_2 = [11, 17]$, task 3 : $p_3 = 4$, $d_3 \in D_3 = [13, 20]$. The set of scenarios for this instance is $\Omega = D_1 \times D_2 \times D_3$ and $\omega = (10, 12, 19)$ is a scenario. We will keep this instance all along this paper to exemplify the notions and algorithms we introduce.*

2.2 Information Model

As explained in Section **??** we suppose that at some time during the execution of the schedule, some information become accessible. In our model, an information allows the scheduler to tighten the interval of uncertainty of a future realisation.

Definition 1 *For a given uncertain parameter $x \in X$ and a moment of decision t during the execution of the schedule, we call an information about x a value k^t_x and an operator in $\{\leq, \geq\}$.*

For instance, an information is $x \leq k^t_x$. So the decision maker, from time t on, is able to reduce the set of possible assignments by updating the interval X, $x \in X^{inf}_{k^t_x} = [x_{min}, k^t_x]$ or $x \in X^{sup}_{k^t_x} =]k^t_x, x_{max}]$. Note that the information depends on t, so the scheduler may have to ask for an information about the same data several times during the execution of the planning. Our model aims to make the best use of available information, and select the more relevant ones.
For the problem considered in this paper we suppose that for a given moment of decision t and a task i, we have an information k^t_d if $\min(t + p_i, 2t) \in D_i$. If so, $k^t_d = \min(t + p_i, 2t)$. Otherwise we consider that we have no information about the task i . This hypothesis on the availability of information is arbitrary, but in fact it expresses two natural questions a scheduler may ask about uncertain due dates : "If task i is started now, can it be completed without being late ? If so, is the due date d_i far from now ?" In any case, an answer to these questions allows the scheduler to bound the uncertainty of a due date d_i.

Example 2 *Let us look again at the instance from Example 1. We suppose that we are at a moment of decision $t = 10$ and that task 1 has been scheduled first. Given the hypothesis we made about information availibility, we have:*

- *task 1 is completed, so there is no relevant information about it.*
- $\min(t + p_2, 2t) = 16 \in D_2$, *so* $k^t_{d_2} = 16$.
- $\min(t + p_3, 2t) = 14 \in D_3$, *so* $k^t_{d_3} = 14$.

Therefore, for any $\omega \in \Omega$, the scheduler is able to determine, from time $t = 10$, if $\omega_2 \leq 16$ or if $\omega_2 > 16$, and if $\omega_3 \leq 14$ or if $\omega_3 > 14$.

2.3 Robust Decision Tree

Definition 2 *A robust decision tree T is a tree where the nodes are labeled with a subset of Ω and a partition of this subset, and the arcs are labeled with a partial schedule. If n is a node of T, Ω^n and P^n denote respectively a subset of Ω and a partition of Ω^n, both associated to n. A robust decision tree satisfies the following properties:*

(i) *The node n has exactly $|P^n|$ children.*
(ii) *Let us denote by $(n_j)_{j\leq|P^n|}$ the children of node n. Each n_j is associated with one element of P^n, i.e for all $j \leq |P_n|$, $\Omega^{n_j} \in P^n$ and $\bigcup_{j\leq|P_n|} \Omega^{n_j} = \Omega^n$.*
(iii) *For any path $(n_0, ..., n_m)$ where n_0 is the root of T and n_m is a leaf, the schedule obtained by concatenating all the partial schedules on the arcs along the path is feasible.*
(iv) *Let n and n' be two nodes of T. The partial schedule s' on the arc (n, n') is robust:*

$$s' = \arg\min_{s\in S} \max_{\omega\in\Omega^{n'}} f(\omega, s)$$

where S is the set of admissible partial solutions.

Example 3 *Based on the instance from Example 1, Figure 1 is an illustration of a Robust Decision Tree. In this case, at the first node after the root, Ω is splitted in Ω_1 and Ω_2. Finally, the tree may lead to two distinct solutions according to the ongoing scenario ω : the sequence of task $[1, 2, 3]$ if $\omega \in \Omega_1$, or $[1, 3, 2]$ if $\omega \in \Omega_2$.*

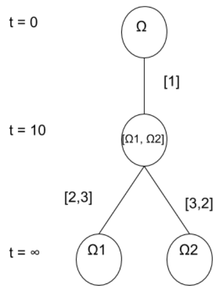


Fig. 1. Illustration of small robust decision tree, for Example 3.

2.4 Partitioning the Scenarios

The core problem of our method is, for a given node n, computing a robust partition P^n, but how do one compare the robustness of two different partitions ? We propose the following criterion. We define the Robustness Score (RS) of a partition P as the sorted vector of the worst case objective values of the optimal robust solution (considering absolute robustness) on each element of P :

$$RS_f(P) = (\min_{s\in S} \max_{\omega\in p} f(\omega, s))_{p\in P}$$

where S is the set of feasible solutions.

As we supposed that we have no information about data distribution, it would be inconsistent to look at the "size" of the element of P. Now, given two partitions P and

P', we say that P is a better partition than P' if $RS(P) <_{lexi} RS(P')$ where $<_{lexi}$ is the lexicographical order. We call Robust Partition Problem (RPP) the problem of finding the best partition with that criterion.

3 Algorithms

Using the previous definitions we now propose a method to build a robust decision tree (see Definition 2). In this paper, we consider that the moments of decision (i.e. moments when the scheduler is able to access new information and change the schedule), denoted by $(t_j)_{j \in J}$ are fixed in advance. This may correspond in practice to special times, such as the end of a working day, or a shift change where the planning can be updated. Every decision moment corresponds to a level in the decision tree, such that t_1 corresponds to the first level, t_2 to the second one, etc... In that respect, the depth of the tree is controlled by the number of decision moments. At each fork at a level j in the tree, a new partial solution, consistent with the partial schedule that has been accomplished until t_j, is proposed according to the current set of scenarios. The root of the tree, that we consider being the level 0 corresponds to the time $t_0 = 0$, the beginning of the schedule. At this point no information is known, so only one robust solution is proposed. Thus, a single node is created at level 1. At this node, we retrieve all the information available at time t_1. Using up to K information, we split the set of scenarios into -at most 2^k- subsets forming a partition P. and obtain a robust partition P'. For each subsets in P' a new solution is proposed and a new branch is set up, leading to a new node at the next level. The set of scenarios considered in this node is the one from which it originated in P'. These steps are repeated until the last decision moment is reached. to solve the the RPP we propose an exhaustive algorithm, that enumerate each combination of information and, provided that the problem admits a global worst case scenario, extracts a dominant partition from this combination. This algorithm has an exponential complexity since that at each iteration of the for loop, we compute a partition P such that $|P| = 2^{|K|}$.

4 Results

The objective of the carried out experiments is to evaluate the robustness of our robust decision tree model, the quality of the selected information used for its construction and its stability in terms of number of reactions for a simulation over 500 scenarios. As our approach uses the notion of information to reduce uncertainties to provide more robust solution, we compare it to a more standard proactive-reactive algorithms. We observe that the robust decision trees provide better solutions (for a given number of tasks) when uncertainty intervals are larger. Intuitively, this can be explained by the fact that decision trees are very constrained by the moments of decision while the reactive algorithms is not. So, larger uncertainties allow robust decision trees to acquire more new information than it does with robust reactive algorithms. Overall our approach obtains better solutions using less information and reactions.

References

1. Davari, M., and Demeulemeester, E. 2017. The proactive and reactive resource-constrained project scheduling problem. *Journal of Scheduling* 1–27.
2. Dearden, R.; Meuleau, N.; Ramakrishnan, S.; Smith, D. E.; and Washington, R. 2003. Incremental contingency planning. In *ICAPS-03 Workshop on Planning under Uncertainty.*
3. Van de Vonder, S.; Demeulemeester, E.; and Herroelen, W. 2007. A classification of predictive-reactive project scheduling procedures. *Journal of scheduling* 10(3):195–207.

A Benders decomposition for the flexible cyclic jobshop problem

Félix Quinton[1], Idir Hamaz[2] and Laurent Houssin[3]

[1] ONERA – The French Aerospace Lab, F-31055, Toulouse, France
felix.quinton@onera.fr
[2] LIRMM UMR 5506, Université de Montpellier, 34932 Montpelier Cedex 5, France
idir.hamaz@lirmm.fr
[3] LAAS-CNRS, Université de Toulouse, CNRS, UPS, Toulouse, France
houssin@laas.fr

Keywords: Cyclic scheduling, Decomposition method, Flexible scheduling.

1 Introduction

This paper tackles the Flexible Cyclic Jobshop Scheduling Problem (FCJSP). We propose a Mixed Integer Linear Programming (MILP) formulation for the FCJSP along with a Benders decomposition algorithm adapted for the FCJSP. The Basic Cyclic Scheduling Problem (BCSP) is a generalisation of the basic scheduling problem where a schedule of elementary tasks $i \in \mathcal{T} = \{1, ..., n\}$ is infinitely repeated. This problem has been studied a lot since it has applications in many fields such as parallel processing ([Hanen and Munier, 1995]), staff scheduling ([Karp and Orlin, 1981]), or robotic scheduling ([Kats and Levner, 1996]). [Hamaz et al., 2018a] studied the BCSP where the processing times are affected by an uncertainty effect.

To specify a BCSP, we are given a set of elementary tasks $i \in \mathcal{T}$ and their respective execution time p_i. Also, we define $t_i(k) \geq 0$ the starting time of occurrence k of task $i \in \mathcal{T}$. The order in which the tasks are to be executed is fixed by precedence constraints. A precedence constraint indicating that task i must be completed before starting the execution for task j can be expressed as a quadruplet (i, j, p_i, H_{ij}), where H_{ij} is called the height of the precedence constraint and represents the occurrence shift between tasks i and j. In simple terms, it states that the $k + H_{ij}$-th occurrence of task j cannot start before the end of the k-th occurrence of task i. Precisely, it states that : $t_j(k + H_{ij}) \geq p_i + t_i(k)$.

2 The cyclic jobshop scheduling problem

In the Cyclic Jobshop Scheduling Problem (CJSP), each elementary task $i \in \mathcal{T} = \{1, ..., n\}$ is assigned to a machine $r_i \in \mathcal{R} = \{1, ..., R\}$, with $R < n$ and elementary tasks linked by precedence constraints constitute jobs. For instance, in a manufacturing context, a job might represent the manufacturing process of a product as a whole, while an elementary task represents only one step of the manufacturing process.

Due to its numerous applications in large-scale production scheduling, the CJSP has received at lot of attention from the research community. [Hanen, 1994] proposed a MILP for this problem and proved some of its properties. [Brucker et al., 2012] also proposed a MILP for a CJSP with transportation by a single robotic cell. [Draper et al., 1999] proposed an alternative formulation to solve CJSP problems as constraint satisfaction problems. [Hamaz et al., 2018b] studied the CJSP with uncertain processing time. The inclusion of machines in the CJSP leads to a lack of resources, since the tasks are competing for the working time of the machines. This lack of resources is represented by disjunction constraints, which state that for a pair of tasks $(i, j) \in \mathcal{T}^2, i \neq j$, that must be executed on the same machine, i.e. $r_i = r_j$, an occurrence of i and an occurrence of j, cannot be executed at the same time. In the following of this paper, we will denote by $\mathcal{D} = \{(i,j)|R(i) \cap R(j) \neq \emptyset\}$ the set of pairs of tasks linked by disjunction constraints. Mathematically, the disjunction between two tasks $(i,j) \in \mathcal{T}^2, i \neq j$ is modeled with the two following disjunction constraints (1):

$$t_j(k+K_{ij}) \geq t_i(k) + p_i \qquad \text{and} \qquad t_i(k+K_{ji}) \geq t_j(k) + p_j. \tag{1}$$

where K_{ij} (resp. K_{ji}) is the height of the disjunction constraint, i.e. the occurrence shift between tasks i and j (resp. j and i). It has been proven by [Hanen, 1994] that a feasible schedule for a CJSP must satisfy $K_{ij} + K_{ji} = 1$.

Note that in this problem the variables are the cycle time α, the starting times of each elementary tasks $(t)_{i\in\mathcal{T}}$, and the heights of the disjunctive constraints, $(K)_{(i,j)\in D}$. A feature of the CJSP is the Work In Process (WIP). It corresponds to the maximum number of occurrences of a job processed simultaneously. Mathematically, the role of the WIP can be modelled as the height of the precedence constraint from fictive task e to fictive task s, and can be explained by the equation :

$$s(k) \geq e(k - WIP).$$

In our study, we aim at minimizing the cycle time α, so the WIP and the $H_{ij}, (i,j) \in \mathcal{E}$ are given. Modelling of the CJSP for the minimisation of the cycle time α with known heights as been proposed by [Hanen, 1994] and is used by [Brucker et al., 2012] to solve a CJSP with transportation.

[Hanen, 1994] proposes to define the variable $\tau = \frac{1}{\alpha}$ and for all $i \in \mathcal{T}$, the variable $u_i = \frac{t_i}{\alpha}$. Then CJSP can then be considered as a MILP in the following form:

$$\max \tau$$

s.t.

$$\tau \leq \frac{1}{p_i}, \quad \forall i \in \mathcal{T} \tag{2a}$$

$$u_j + H_{i,j} \geq u_i + \tau p_i, \quad \forall (i,j) \in \mathcal{E} \tag{2b}$$

$$u_j + K(i,j) \geq u_i + \tau p_i, \quad \forall (i,j) \in \mathcal{D} \tag{2c}$$

$$K(i,j) + K(j,i) = 1, \quad \forall (i,j) \in \mathcal{D} \tag{2d}$$

$$K(i,j) \in \mathbb{Z}, \quad \forall (i,j) \in \mathcal{D} \tag{2e}$$

$$u_i \geq 0, \quad \forall i \in \mathcal{T}. \tag{2f}$$

The CJSP can then be solved by writing the problem as a MILP, which can be solved using mathematical programming or using a dedicated Branch-and-Bound procedure ([Fink et al., 2012], [Hanen, 1994]).

3 The flexible cyclic jobshop scheduling problem

The Flexible Cyclic Jobshop Scheduling Problem (FCJSP) is a CJSP where the elementary tasks are flexible. This means that the execution of a task $i \in \mathcal{T}$, is assigned to exactly one machine r in a set of machines that is a subset of the set of machines $\mathcal{R}$ specific to task i. This subset is denoted $R(i) \subset \mathcal{R}$. We model the assignment of a task $i \in \mathcal{T}$ to a machine $r \in R(i)$ as a decision variable $m_{i,r}$ defined as follows :

$$\forall i \in \mathcal{T}, \forall r \in R(i), \quad m_{i,r} = \begin{cases} 1 \text{ if task } i \text{ is assigned to machine } r \\ 0 \text{ otherwise.} \end{cases}$$

Each assignment of a task $i \in \mathcal{T}$ to a machine $r \in R(i)$ is associated with a given execution time denoted p_{ir}. Also, because we do not know *a priori* on which machine each task will be assigned, we do not know the set $(i,j) \in \mathcal{T}^2, i \neq j, R(i,j) \neq \emptyset$ of pairs of tasks which are connected by a disjunctive constraint.

Based on the model of Section 2 and variables $m_{i,r}$, we have proposed a MILP for the FCJSP. A first model was proposed in [Quinton et al., 2018] but substantial improvements have been made since this first model.

4 A Benders decomposition for the FCJSP

The FCJSP formulated as a MILP can be very hard to solve. Using CPLEX, difficult instances with a large number of tasks or with few robots might exceed any reasonable time limit. To tackle this issue, we propose a Benders decomposition for the FCJSP. In the usual Benders decomposition scheme, two problems coexist: the master problem and the sub-problem. The Master Problem (MP) consists in an integer linear problem composed of the constraints from the model described in Section 3 involving only the integer variables, and the optimality cuts generated at each iteration of the Benders algorithm. The remaining constraints, involving only continuous variables or a combination of continuous and integer variables, compose the primal sub-problem. It can be written as a linear problem. The full description of the algorithm is available in [Quinton et al., 2019].

5 Numerical results

The MILP solving is very efficient for the easiest instances (10 tasks and 5 machines). For those easy instances, it is much more efficient than the Benders decomposition. For instances of average difficulty with 10 tasks and 4 machines, the MILP is still more efficient than the Benders decomposition, but we can remark that the execution time of the MILP is increasing much faster than the execution time of the Benders decomposition. Finally, for the hard instances with 10 tasks and 3 machines, our Benders decomposition is always faster to find the optimal solution than the MILP. From these results, we learn that it is better to use the MILP for easy problems with numerous machines such as our instances with 10 tasks and 5 machines. However, for hard instances with a considerable number of disjunctions, such as the instances with 10 tasks and 3 machines, the execution times of the MILP rocket up and it is much better to use the Benders decomposition to obtain an optimal solution or a heuristic procedure to obtain a feasible solution (not presented here).

6 Conclusion

We have proposed a MILP for the FCJSP where the objective is the minimisation of the cycle time. The problem is highly combinatorial, so we also proposed a Benders decomposition algorithm that is more efficient for difficult instances. The Benders decomposition includes specific cuts to ensure the feasibility of the integer solution. Numerical instances have shown that the MILP becomes inefficient for difficult instances with many disjunctions. Also, if an optimal solution is required, our Benders decomposition is more efficient than the MILP for this type of instances.

References

[Brucker et al., 2012] Brucker, P., Burke, E. K., and Groenemeyer, S. (2012). A mixed integer programming model for the cyclic job-shop problem with transportation. *Discrete Applied Mathematics*, 160(13-14):1924–1935.

[Draper et al., 1999] Draper, D. L., Jonsson, A. K., Clements, D. P., and Joslin, D. E. (1999). Cyclic scheduling. In *IJCAI*, pages 1016–1021. Citeseer.

[Fink et al., 2012] Fink, M., Rahhou, T. B., and Houssin, L. (2012). A new procedure for the cyclic job shop problem. *IFAC Proceedings Volumes*, 45(6):69–74.

[Hamaz et al., 2018a] Hamaz, I., Houssin, L., and Cafieri, S. (2018a). A robust basic cyclic scheduling problem. *EURO Journal on Computational Optimization*, 6(3):291–313.

[Hamaz et al., 2018b] Hamaz, I., Houssin, L., and Cafieri, S. (2018b). The Cyclic Job Shop Problem with uncertain processing times. In *16th International Conference on Project Management and Scheduling (PMS 2018)*, Rome, Italy.

[Hanen, 1994] Hanen, C. (1994). Study of a np-hard cyclic scheduling problem: The recurrent job-shop. *European journal of operational research*, 72(1):82–101.

[Hanen and Munier, 1995] Hanen, C. and Munier, A. (1995). A study of the cyclic scheduling problem on parallel processors. *Discrete Applied Mathematics*, 57(2-3):167–192.

[Karp and Orlin, 1981] Karp, R. M. and Orlin, J. B. (1981). Parametric shortest path algorithms with an application to cyclic staffing. *Discrete Applied Mathematics*, 3(1):37–45.

[Kats and Levner, 1996] Kats, V. and Levner, E. (1996). Polynomial algorithms for scheduling of robots. *Intelligent Scheduling of Robots and FMS*, pages 77–100.

[Quinton et al., 2018] Quinton, F., Hamaz, I., and Houssin, L. (2018). Algorithms for the flexible cyclic jobshop problem. In *14th IEEE International Conference on Automation Science and Engineering, CASE 2018, Munich, Germany, August 20-24, 2018*, pages 945–950.

[Quinton et al., 2019] Quinton, F., Hamaz, I., and Houssin, L. (2019). A mixed integer linear programming modelling for the flexible cyclic jobshop problem. *Annals of Operations Research.*

Exact and heuristic methods for characterizing optimal solutions for the $1||L_{max}$

Tifenn Rault, Ronan Bocquillon and Jean-Charles Billaut

Université de Tours, LIFAT EA 6300, CNRS, ROOT ERL CNRS 7002
64 Avenue Jean Portalis, 37200 Tours
tifenn.rault, ronan.bocquillon, jean-charles.billaut@univ-tours.fr

Keywords: scheduling, single machine, characterization of solutions, deadlines, lattice.

1 Introduction

The $1||L_{max}$ problem can be solved in $\mathcal{O}(n \log n)$ applying EDD rule (Jackson 1955). Still, as many scheduling problems, $1||L_{max}$ may have a huge number of optimal solutions. Our objective is to characterize easily a set of optimal solutions, without enumerating them. Such an approach is useful in a dynamic environment, to react in real time to unexpected events, or to data uncertainty. Some preliminary studies in this direction have been conducted using the lattice of permutations as support (Billaut and Lopez 2011, Billaut *et. al.* 2012, Ta 2018). In such a framework, it is assumed that jobs are renumbered in EDD order and due dates are transformed in deadlines so that sequence EDD is feasible and is the root sequence of the lattice. A property is that all the predecessors of a feasible sequence in the lattice are feasible as well, and therefore optimal sequences for the $1||L_{max}$. These predecessors can be characterized very easily by a partial order between jobs. The distance to the bottom sequence (reverse EDD sequence), also called "level", is denoted by $\sum N_j$. It is expected that a sequence with minimum level will characterize a lot of optimal solutions. In this paper, we study the problem $1|\widetilde{d_j}|\sum N_j$ of finding a sequence with minimum level. We improve the existing branch and bound algorithm introduced in (Ta 2018) thanks to memorization, and we propose a new heuristic method. Our results show that we improve state-of-the-art resolution methods, solving to optimality new instances.

2 Problem description

We consider a set of n jobs $J_j, 1 \leq j \leq n$. To each job is associated a processing time p_j and a due date d_j. For any instance, we apply the following pre-treatment in polynomial time: (1) we apply EDD rule and we obtain L^*_{max}, the optimal maximum lateness value. Then, (2) we modify the due dates in order to obtain deadlines: $\widetilde{d_j} = \min(d_j + L^*_{max}, \sum p_j)$, for any $j \in \{1, 2, .., n\}$. Finally, (3) we renumber the jobs in EDD order, and in case of a tie, in LPT order. At the end we know that sequence $\sigma = (J_1, J_2, ..., J_n)$ is feasible and $i < j \iff (\widetilde{d_i} < \widetilde{d_j})||(\widetilde{d_i} = \widetilde{d_j}, p_i \geq p_j)$.

The lattice of permutations is a digraph constructed as follows. The root node is the EDD sequence, i.e $(J_1, J_2, ..., J_n)$. The children of a sequence σ are created by inverting two jobs J_k and J_l iff J_k is just before J_l in σ, and $k < l$. The process is repeated for newly created sequences until the sink node (i.e reverse EDD sequence) is reached. Each sequence in the lattice can be associated with a *level* which represents the number of inversions from the reverse EDD sequence at level 0, i.e the number of times we have J_i before J_j and $i < j$ (see (Billaut *et. al.* 2012) for more details). The level of a sequence also gives the number of precedence relations characterized by this sequence. One nice property of this lattice is that all predecessors of a feasible sequence are also feasible (and therefore optimal for the $1||L_{max}$).

Finding a sequence as deep as possible in this lattice, or finding a sequence as far as possible from EDD sequence, is the same, and is equivalent to minimising the objective function $\sum N_j$. The variable N_j, the contribution of job J_j to the objective function, is equal to the number of jobs after J_j with and index greater than j. The complexity of problem $1|\widetilde{d_j}|\sum N_j$ remains open (Ta 2018). It has been shown in (Ta *et. al.* 2017, Ta 2018) that the expression $\sum N_j$ is somewhat related to the positions P_j of the jobs, and the problem $1|\widetilde{d_j}|\sum w_j P_j$ is proved strongly NP-hard by reduction from Numerical 3-Dimensional Matching problem.

3 Exact method

The original B&B method for the $1|\widetilde{d_j}|\sum N_j$ problem works as follows. Consider first the initial set of unscheduled jobs in reverse numbering order $S = \{J_n, J_{n-1}, ..., J_1\}$, and $\sigma = \emptyset$ the sequence being build, starting by the end. At each level, add a new unscheduled job J_j in first place of σ. While $S \neq \emptyset$, the job J_j is chosen in S from the smallest to the biggest index so that: the deadlines are respected (put only in the first position of sequence σ a non tardy job), and the dominance conditions too (keep only the sequences satisfying dominance properties).

The initial upper bound is given by a polynomial time heuristic method. The lower bound is computed in $O(1)$. Indeed, when adding a job J_j in front of σ, the contribution of J_j to the level of all unscheduled jobs is exactly its position in S minus 1. Several properties of optimal (or feasible) sequences are presented in (Ta 2018) and exploited by the original Branch-and-Bound. Two new dominance rules are presented hereafter.

Property 1 *Two jobs J_i and J_j with identical deadlines and $i < j$ must be sequenced so that J_i is after J_j in the sequence.*

Sketch of the proof Let us consider two jobs J_i and J_j such that $\widetilde{d_i} = \widetilde{d_j}$ and $i < j$. Due to our renumbering scheme (see Section 2), we know that $p_i \geq p_j$. Suppose sequence is built in a backward manner. If a choice has to be made between J_i or J_j, it is always preferable to place J_i since it will possibly enable to place a job with a smaller index (i.e. a smaller deadline) right after, and J_i has a smallest index than J_j, thus decreasing the objective function.

Property 2 *If two partial solutions contain exactly the same subset of jobs, then the one with the lowest level dominates the other.*

Sketch of the proof Suppose we have two partial solutions σ_u and σ_v such that σ_v contains exactly the same subset of jobs as σ_u (but in different order), and $level(\sigma_u) = \sum_{i \in \sigma_u} N_i \leq \sum_{j \in \sigma_v} N_j = level(\sigma_v)$. We have $S_u = S_v = S$. Let S^* be the optimal way of arranging the jobs in S, s_u^* be the concatenation of S^* and σ_u, and s_v* be the concatenation of S^* and σ_v. We have $level(s_u^*) = level(S^*) + level(\sigma_u)$ and $level(s_v^*) = level(S^*) + level(\sigma_v)$. So, $level(s_u^*) \leq level(s_v^*)$: the partial solution σ_u dominates σ_v.

We denote by B&BP1 the B&B integrating property 1, and we call "Branch-and-Bound with memorization" (denoted B&BM) the B&B integrating properties 1 and 2. To leverage property 2, the set of jobs in σ of visited nodes must be stored, so that dominated nodes can be pruned. In practice, it requires a balance between time gain and memory storage. This method takes as a parameter the maximum cardinality of the recorded sets. Typically, in the results table, B&BM x means that only sets whose cardinality is less than x percent of n are registered. This strategy is motivated by the fact that the more a branch is high, the more pruning it will have an impact.

4 Heuristic method

In addition to the exact method, we investigated a new heuristic strategy based on the Limited Discrepancy Search (LDS). For large search tree, exhaustive search is not tractable. The LDS technique was introduced by Harvey and Ginsberg (Harvey and Ginsberg 1995) to cope with this limitation. A "discrepancy" is a right branch in a heuristically ordered tree. Originally LDS is an iterative procedure. It first generates the leftmost path. Next, it generates those paths with at most one right branch from the root to the leaf. The next set of paths generated by LDS are those with at most two right branches, etc. The process continues until every path has been generated, with the rightmost path being generated last.

This technique seems well suited to our problem since we expect that in the search tree, the feasible solutions with the lowest levels are located among the leaves as far as possible on the left of the search tree. Indeed, the levels of the solutions at the leaves are *almost* distributed from the smallest to the largest, from the left to the right. In LDS-k, we therefore allow during the search the generation of paths with at most k right branches. Note that we use it as a non-iterative heuristic. We can combine LDS-k with property 2 in order to cut branches, and we denote this technique LDS-k M-x, where x means that only sequences whose size is less than x percent of n are registered.

5 Computational results

We compare the original B&B with B&BP1 and with B&BM. We also compare the LDS heuristic with the backward heuristic (denoted BW) introduced in (Ta 2018), which builds the solution by the end, putting in last position the feasible job with the smallest index. We used type I data sets used in (Ta 2018). For each value of n, with $n \in \{10, 20, ..., 100\}$, there are 30 instances. The experiments were run on Intel E5-2670V2 processors running at 2.5GHz (one core per instance). The memorization database is capped at 10^7 entries and the CPU time to solve each instance has been limited to 180 seconds.

Table 1. Performances of exact methods

	(Ta 2018)		**B&BP1**		**B&BM 30**		**B&BM 70**		**B&BM 90**	
n	**opt**	**cpu**	**opt**	**cpu**	**opt**	**cpu**	**opt**	**cpu**	**opt**	**cpu**
10	30	0,00	30	0,00	30	0,00	30	0,00	30	0,00
20	30	0,00	30	0,00	30	0,00	30	0,00	30	0,00
30	30	0,00	30	0,00	30	0,00	30	0,00	30	0,00
40	30	0,00	30	0,00	30	0,00	30	0,00	30	0,00
50	30	0,00	30	0,00	30	0,01	30	0,01	30	0,01
60	30	0,17	30	0,14	30	0,12	30	0,12	30	0,12
70	30	1,88	30	**1,52**	30	**0,61**	30	**0,61**	30	**0,62**
80	28	36,17	**30**	**29,38**	**30**	**6,95**	**30**	**6,81**	**30**	**6,98**
90	12	133,31	12	**128,49**	**17**	**100,26**	**17**	**98,13**	**18**	**99,17**
100	1	175,59	1	175,27	**2**	**170,15**	**2**	**170,21**	**2**	**170,21**

Table 1 compares the performances of the exact methods. It can be seen that we solved to optimality 9 new instances for $n \geq 80$. More precisely, Property 1 allows us to solve two new instances, while B&BM x solves more instances as parameter x increases.

Table 2. Performances of heuristic methods

	BW	LDS-1		LDS-1 M-30		LDS-2		LDS-2 M-30	
n	**gap**	**cpu**	**gap**	**cpu**	**gap**	**cpu**	**gap**	**cpu**	**gap**
10	2,0%	0,00	0,0%	0,00	0,0%	0,00	0,0%	0,00	0,0%
20	20,8%	0,00	1,7%	0,00	1,7%	0,00	0,0%	0,00	0,0%
30	27,3%	0,00	7,4%	0,00	7,4%	0,00	1,1%	0,00	1,1%
40	30,5%	0,00	8,8%	0,00	8,6%	0,00	2,0%	0,00	2,0%
50	42,6%	0,00	15,4%	0,00	14,8%	0,00	4,4%	0,00	4,3%
60	41,0%	0,00	15,3%	0,01	13,9%	0,00	6,6%	0,01	6,5%
70	41,6%	0,03	17,8%	0,10	16,0%	0,01	8,3%	0,07	7,8%
80	45,9%	0,06	23,3%	1,31	18,6%	0,03	12,0%	0,87	10,1%
90	42,0%	5,08	21,2%	28,73	15,5%	2,08	10,6%	23,70	7,4%
100	39,6%	47,18	15,3%	111,04	9,5%	34,93	8,2%	127,61	1,1%

Regarding the heuristic methods, Table 2 shows that the LDS-k approach exhibits lowest gap than the backward heuristic. For $n \geq 70$, this comes at an increased computation time. Still we can point out that LDS-2 and LDS-2 M-30 offer a good trade-off between computation time and solution quality for $n \geq 90$. Indeed, it provides solutions with gaps between $1.1\% - 10.6\%$ faster than the B&B approach.

6 Conclusion

In this paper, we addressed the problem $1|\widetilde{d_j}|\sum N_j$ whose complexity remains open. We introduced two new efficient dominance rules that allowed us to solve new instances to optimality. A new heuristic is presented. Its performances offer a good trade-off between computation time and solution quality.

Future directions include generating larger and harder instances, and studying different policies for choosing which sets to remove from the memorization database when it is full.

References

Billaut J.-C, P. Lopez, 2011, "Characterization of all p-approximated sequences for some scheduling problem", *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA.*

Billaut J.-C., E. Hebrard and P. Lopez, 2012, "Complete Characterization of Near-Optimal Sequences for the Two-Machine Flow Shop Scheduling Problem", *Ninth International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming (CPAIOR'2012), Nantes, France.*

Harvey W. D., M. L. Ginsberg, 1995, "Limited Discrepancy Search", *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 607-613.

Jackson J.R., 1955,"Scheduling a production line to minimize maximum tardiness", *Research report 43, Management Science Research Project, University of California, Los Angeles.*

Ta T.T.T., R. Kivin and J.-C. Billaut, 2017, "New objective functions based on jobs positions for single machine scheduling with deadlines", *7th International Conference on Industrial Engineering and Systems Management (IESM 2017), Saarbrucken, Germany.*

Ta T.T.T., 2018, "New single machine scheduling problems with deadlines for the characterization of optimal solutions", Thèse, *Université de Tours.*

A Comparison of two MILP formulations for the resource renting problem

Max Reinke and Jürgen Zimmermann

Clausthal University of Technology, Germany
`max.reinke@tu-clausthal.de, juergen.zimmermann@tu-clausthal.de`

Keywords: Project scheduling, resource renting, mixed-integer linear programming.

1 Introduction

Typically each activity in a project uses resources during its execution. In many cases those resources have to be rented, for example heavy machinery on construction sites. The renting of resources induces two types of costs namely time-independent procurement costs and time-dependent renting costs. In this paper we present two MILP formulations for the resource renting problem with general temporal constraints (RRP/max). The RRP/max was introduced by Nübel (2001) and extends the well-known resource availability problem by taking time-dependent renting costs into account. The problem aims at minimizing the total resource costs to execute a project while taking general temporal constraints into account. In Section 2 the RRP/max is described formally. Section 3 presents two different time-indexed MILP formulations. Finally, in Section 4 the results of a preliminary computational study are presented where we compare the performance of our two models.

2 Problem description

With given renting and procurement costs for each resource type the resource renting problem (RRP/max) can be modeled as an activity-on-node network, where the nodes represent activities $V = \{0, \dots, n+1\}$ with n real activities and the fictitious project start 0 and project end $n+1$. Each activity $i \in V$ is assigned a duration $p_i \in \mathbb{Z}_{\geq 0}$ and has to be performed without preemption. The arcs of the network given by set $E \subseteq V \times V$ represent temporal constraints between activities. Arc weights $\delta_{ij} \geq 0$ indicate a minimal time lag for arc $\langle i, j\rangle \in E$ while $\delta_{ij} < 0$ gives a maximal time lag between the start times of activities i and j. Here a project has a given deadline $\bar{d}$ introduced by $\langle n+1, 0\rangle$ with $\delta_{n+1,0} = -\bar{d}$. A sequence of start times S_i for all activities $i \in V$ is called a schedule $S = (S_0, S_1, \dots, S_{n+1})$, which is termed time-feasible if $S_j - S_i \geq \delta_{ij}$ for all $\langle i, j\rangle \in E$. Each activity i in a project has an earliest start time ES_i and a latest start time LS_i, so that the possible start times can be limited to the set $W_i = \{ES_i, ES_{i+1}, ..., LS_i\}$. For the execution of activity i an amount $r_{ik} \in \mathbb{Z}_{\geq 0}$ of resource $k \in \mathcal{R}$ is needed. The RRP/max considers renewable resources which have to be rented. The usage of one unit of resource k for t periods incurs procurement costs c^p_k and time-dependent renting costs of $t \cdot c^r_k$. To carry out a project according to a given time-feasible schedule S, the number of available (rented) resources has to be equal to or exceed the resource demand for all $k \in \mathcal{R}$ and $t \in T$. Thus, a renting policy $\varphi_k(S, t)$ is needed, which specifies when resources are procured or released, so that $\varphi_k(S, t) \geq \sum_{i \in V | S_i \leq t < S_i + p_i} r_{ik}$ for all $k \in \mathcal{R}$ and $t \in T$. For the case $c^p_k < c^r_k$ an optimal renting policy is to procure additional resources when the resource demand has a positive step discontinuity at t and to release idle resources immediately, since procuring new resources is less expensive than keeping unused resources. In general $c^p_k > c^r_k$ holds, here procurement costs of a resource k are higher than renting costs for one period. In this case, we can define *span* $= \lfloor c^p_k / c^r_k \rfloor$ as the maximum number of time periods for which it is beneficial to hold unused resources if they are used later in the project. The objective of the RRP/max is to find a schedule which minimizes the total costs incurred by renting resources. The problem can be stated as follows.

$$\begin{aligned}
\text{Min} \quad & F(S) = \sum_{k\in\mathcal{R}} \left[c_k^r \int_0^{\bar{d}} \varphi_k(S,t)dt + c_k^p \sum_{t\in J_k} \Delta^+ \varphi_k(S,t) \right] & \\
\text{s.t.} \quad & S_j - S_i \geq \delta_{ij} & (\langle i,j\rangle \in E) \\
& \varphi_k(S,t) \geq \sum_{i\in V | S_i \leq t < S_i + p_i} r_{ik} & (t\in T, k\in\mathcal{R}) \\
& S_0 = 0 & \\
& S_i \in \mathbb{Z}_{\geq 0} & (i \in V)
\end{aligned}$$

3 Mixed-integer linear programs

Both MILP formulations presented in this paper are based on well-known time-indexed formulations for the RCPSP. The first formulation based on the formulation by Pritsker *et al.* (1969) uses binary variables x_{it} for all $i \in V$ and $t \in T$ where $x_{it} = 1$ if activity i starts at point in time t. To model the resource demand, we use the variable z_{kt} representing the amount of resource $k \in \mathcal{R}$ needed at $t \in T$. The renting policy $\varphi_k(S,t)$ is modeled using positive variables a_{kt} and w_{kt}, which represent the number of units of resource k added or withdrawn at t. With these variables the model (RRP-SP) can be formulated.

$$\begin{aligned}
\text{Min} \quad & \sum_{k\in\mathcal{R}} c_k^p \sum_{t\in T} a_{kt} + \sum_{k\in\mathcal{R}} c_k^r \sum_{t\in T} t \cdot (w_{kt} - a_{kt}) & & (1.1) \\
\text{s.t.} \quad & \sum_{t\in W_i} x_{it} = 1 & (i\in V) \quad & (1.2) \\
& \sum_{t\in W_j} t\cdot x_{jt} - \sum_{t\in W_i} t \cdot x_{it} \geq \delta_{ij} & (\langle i,j\rangle \in E) \quad & (1.3) \\
& \sum_{i\in V} r_{ik} \sum_{\tau=\max\{ES_i, t-p_i+1\}}^{\min\{t, LS_i\}} x_{i\tau} \leq z_{kt} & (k\in\mathcal{R}, t\in T) \quad & (1.4) \\
& \sum_{\tau=0}^{t} (a_{k\tau} - w_{k\tau}) \geq z_{kt} & (k\in\mathcal{R}, t\in T) \quad & (1.5) \\
& \sum_{t\in T} a_{kt} - \sum_{t\in T} w_{kt} = 0 & (k\in\mathcal{R}) \quad & (1.6) \\
& a_{kt}, w_{kt}, z_{kt} \in \mathbb{Z}_{\geq 0} & (k\in\mathcal{R}, t\in T) \quad & (1.7) \\
& x_{it} \in \{0,1\} & (i \in V, t\in W_i) \quad & (1.8)
\end{aligned}$$

Constraints (1.2) state that every activity has to be started exactly once. With $S_i = \sum_{t\in W_i} t \cdot x_{it}$ (1.3) ensure all temporal restrictions between activities are satisfied. Lower bounds on the minimal value for every k and t are assigned to variables z_{kt} by constraints (1.4). The available (rented) resources at time t are given by $\sum_{\tau=0}^{t}(a_{k\tau} - w_{k\tau})$, hence inequalities (1.5) ensure that a feasible renting policy $\varphi_k(S,t) \geq z_{kt}$ for all $k \in \mathcal{R}$ and $t \in T$ is obtained. Also all added resources have to be withdrawn by the end of the project, see (1.6). Alternative constraints to ensure temporal relations, are the disaggregated precedence constraints proposed by Christofides *et al.* (1987). Here we use the formulation,

$$\sum_{\tau=t}^{LS_i} x_{i\tau} + \sum_{\tau=ES_j}^{\min\{LS_j, t+\delta_{ij}-1\}} x_{j\tau} \leq 1 \qquad (\langle i,j\rangle \in E, t\in T) \quad (1.9),$$

by Rieck *et al.* (2012) to take minimal and maximal time lags into consideration. The second MILP formulation is based on binary on/off variables where $\mu_{it} = 1$ if activity i is in progress at t and otherwise $\mu_{it} = 0$. Here we adapt the model from Artigues (2013) to the RRP/max. By defining $K_{it} = \lfloor t/p_i \rfloor$ for all $i \in V | p_i > 0$ and otherwise $K_{it} = 0$, as the number of potential time windows of length p_i in which an activity could be executed, our second MILP model (RRP-OO) for the RRP/max can be stated as follows.

$$
\begin{array}{lll}
\text{Min} & \sum_{k\in\mathcal{R}} c_k^p \sum_{t\in T} a_{kt} + \sum_{k\in\mathcal{R}} c_k^r \sum_{t\in T} t\cdot(w_{kt}-a_{kt}) & (2.1)\\
\text{s.t.} & \sum_{\lambda=0}^{K_{it}} \mu_{i,t-\lambda\cdot p_i} - \sum_{\lambda=0}^{K_{i,t-1}} \mu_{i,t-\lambda\cdot p_j -1} \geq 0 \quad (i\in V|p_i>0, t\in T\setminus\{0\}) & (2.2)\\
& \sum_{\lambda=0}^{K_{i,LC_i-\phi_i}} \mu_{i,LC_i-\phi_i-\lambda\cdot p_i} = 1 \quad (i\in V) & (2.3)\\
& \sum_{\lambda=0}^{K_{i,t-\delta_{ij}}} \mu_{i,t-\lambda\cdot p_i-\delta_{ij}} - \sum_{\lambda=0}^{K_{jt}} \mu_{j,t-\lambda\cdot p_j} \geq 0 \quad (\langle i,j\rangle\in E|\delta_{ij}\geq 0, t\in T) & (2.4)\\
& \sum_{\lambda=0}^{K^\delta_{ijt}} \mu_{i,\bar{d}-\lambda\cdot p_i-\Delta_{ijt}} \leq 1-\mu_{jt} \quad (\langle i,j\rangle\in E|\delta_{ij}<0, t\in T) & (2.5)\\
& \sum_{i\in V} r_{ik}\cdot\mu_{it}\leq z_{kt} \quad (k\in\mathcal{R}, t\in T) & (2.6)\\
& \sum_{\tau=0}^{t}(a_{k\tau}-w_{k\tau})\geq z_{kt} \quad (k\in\mathcal{R}, t\in T) & (2.7)\\
& \sum_{t\in T} a_{kt} - \sum_{t\in T} w_{kt} = 0 \quad (k\in\mathcal{R}) & (2.8)\\
& \mu_{00}=1 & (2.9)\\
& \mu_{it}=0 \quad (i\in V, t\in T\setminus W_i') & (2.10)\\
& a_{kt}, w_{kt}, z_{kt}\in\mathbb{Z}_{\geq 0} \quad (k\in\mathcal{R}, t\in T) & (2.11)\\
& \mu_{it}\in\{0,1\} \quad (i\in V, t\in W_i) & (2.12)
\end{array}
$$

Constraints (2.2) state that each activity has to be performed during p_i consecutive periods. For all potential time windows between the project start and the latest completion LC_i of activity i, an activity can only be in progress during one of them (2.3). Here $\phi_i = 1$ if $p_i \geq 1$ else $\phi_i = 0$. Constraints (2.4), are modifications of the disaggregated precedence constraints used by Artigues (2013) and model all time lags where $\langle i,j\rangle \in E|\delta_{ij} \geq 0$, by ensuring an activity j can only be processed if activity i has been in progress at least δ_{ij} periods before. To model time lags, where $\delta_{ij} < 0$, we define $K^\delta_{ijt} = \lfloor(\bar{d} - t + p_i + \delta_{ij})/p_i\rfloor$ for all $i \in V|p_i > 0$ and $K^\delta_{ijt} = 0$ otherwise. Moreover $\Delta_{ijt} = mod([\bar{d} - t + \delta_{ij}]/p_i)$ for all $i \in V|p_i > 0$ and $\Delta_{ijt} = 0$ otherwise, is defined. Inequalities (2.5) ensure that for every time lag $\langle i,j\rangle \in E|\delta_{ij} < 0$, if activity i is executed during the time interval $[t + p_i + |\delta_{ij}|, ..., \bar{d}]$ activity j can not be in progress at t, since this would result in $S_j - S_i < \delta_{ij}$ and, therefore, in an infeasible schedule. The calculation of the resource demand is straightforward for this formulation, see (2.6). Constraints (2.7) and (2.8) ensure a feasible renting policy is used. (2.9) state that the project starts at $t = 0$ and (2.10) set all infeasible μ_{it} to 0 where $W_i' = \{ES_i, ES_{i+1}..., LC_i - 1\}$.

4 Lower Bounds for variables

To reduce the number of potential solutions and, therefore, limit the search space, when solving the RRP/max, additional restrictions are devised. We establish a lower bound for the minimal necessary number of resources to procure for the whole project, by defining $\sum_{t\in T} a_{kt} \geq LB^a_k$ for all $k \in \mathcal{R}$ (3.1) where $LB^a_k = \max(LB^a_{1,k}, LB^a_{2,k}, LB^a_{3,k})$. $LB^a_{1,k} = \max_{i\in V}(r_{ik})$ gives the maximum resource demand of k for any activity i. For $LB^a_{2,k} = \lceil\sum_{i\in V} r_{ik}\cdot p_i/\bar{d}\rceil$ the total workload for every resource k is distributed equally over the time horizon of the project. The third lower bound is given by $LB^a_{3,k} = \max_{t\in T}(\sum_{i\in V^{nc}_t} r_{ik})$. Here we use the resource demand of near-critical activities $V^{nc}_t = \{i \in V|LS_i \leq t < ES_i + p_i\}$ to determine the minimal number of resources needed for the project at a given point of time. For a second type of restriction, the near-critical resource demand is used to define a bound $z_{kt} \geq \sum_{i\in V^{nc}_t} r_{ik}$ for all $(t \in T)$ (3.2), where the minimal number of units needed of resource k for every t is determined. By substituting constraints (1.3) with (1.9) and adding the restrictions (3.1) and (3.2), models (RRP-SPC-LB) and (RRP-OO-LB) are obtained.

5 Performance analysis

To compare the performance of the devised formulations a computational study was conducted. The four models were implemented in GAMS v.25.1 and solved, by using *IBM CPLEX* v.12.8.0, on a computer with an Intel Core i7-7700 with 4.2 GHz and 64 GB RAM under Windows 10. As problem instances we used adaptations of the well-known benchmark test set UBO (Schwindt 1998) where we introduced a project deadline $\bar{d} = \alpha \cdot ES_{n+1}$ with $\alpha = \{1, 1.25, 1.5\}$ and procurement costs $c^r_k = CQ \cdot c^p_k$ with $CQ = \{0.5, 0.25, 0.1\}$. For every combination of parameters α and CQ, 30 instances with $n = \{10, 20\}$ and an order strength of 0.5 were used. For the computational study a run time limit of 600 seconds was employed. The results of our study are given in Tab. 1, where for all four models the average *gap* [%] (relative deviation from the best lower bound), the average solution time [s] and the number of instances solved to optimality (max. 30) are given. Instances with a larger time horizon have more possibilities to schedule activities and therefore are harder to solve. For most parameter combinations the presented additional restrictions lead to smaller gaps and lower solution times for both formulations. Especially the larger instances with $n = 20$ show improvements. Preliminary tests for instances with greater numbers of activities $n = \{50, 100\}$, showed a decreasing performance of the OO-models compared to the SP-models.
In conclusion, when comparing the two models with additional constraints (RRP-SPC-LB) and (RRP-OO-LB) for all instances with $n = 20$, we found the solver is able to obtain better solutions in a shorter amount of time with the OO-formulation. In future research, the two formulations presented in this paper could be compared to other possible MILP formulations for the (RRP/max).

Table 1. Results of the computational study

			RRP-SP			*RRP-SPC-LB*			*RRP-OO*			*RRP-OO-LB*		
n	α	CQ	*gap*	*time*	*#opt*	*gap*	*time*	*#opt*	*gap*	*time*	*#opt*	*gap*	*time*	*#opt*
10	1	0,25	0,0	1,0	30	0,0	1,0	30	0,0	0,4	30	0,0	0,4	30
10	1,25	0,25	0,29	74,0	28	0,19	55,9	29	0,0	12,7	30	0,0	12,9	30
10	1,5	0,25	1,69	138,4	25	1,55	144,5	25	0,22	61,4	29	0,22	59,9	29
20	1	0,25	0,0	25,4	30	0,0	25,0	30	0,0	5,9	30	0,0	6,0	30
20	1,25	0,25	8,82	562,6	3	6,9	527,2	5	3,56	465,0	10	2,81	451,0	12
20	1,5	0,25	17,81	600,0	0	15,14	600,0	0	12,11	592,3	1	11,62	591,9	1

References

Artigues C., 2015, “A note on time-indexed formulations for the resource-constrained project scheduling problem”, Technical Report 13206, LAAS, CNRS, Toulouse.

Christofides N., Álvarez-Valdés R., Tamarit J.M., 1987, “Project scheduling with resource constraints: A branch and bound approach”, *European Journal of Operational Research*, Vol. 29, pp. 262-273.

Nübel H., 2001, “The resource renting problem subject to temporal constraints”, *OR Spectrum*, Vol. 23, pp. 359-381.

Pritsker A., Watters L., Wolfe P., 1969, “Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach”, *Management Science*, Vol. 16, pp. 93-108.

Rieck J., Zimmermann J., Gather T., 2012, “Mixed-integer linear programming for resource leveling problems”, *European Journal of Operational Research*, Vol. 221, pp. 27-37.

Schwindt C., 1998, “Generation of resource-constrained project scheduling problems subject to temporal constraints”, *Technical Report WIOR-543*, University of Karlsruhe.

Minimizing the costs induced by perishable resource waste in a chemotherapy production unit

Alexis ROBBES[1], Yannick KERGOSIEN[1], Virginie ANDRÉ[2] and Jean-Charles BILLAUT[1]

[1] Université de Tours, LIFAT EA 6300, CNRS, ROOT ERL CNRS 7002, 64 avenue Jean Portalis, 37200 Tours
{alexis.robbes, yannick.kergosien, jean-charles.billaut}@univ-tours.fr
[2] CHRU de Tours, Hôpital Bretonneau, 2 boulevard Tonnellé, 37044 Tours Cedex 9
v.andre@chu-tours.fr

Keywords: Scheduling, Bin Packing, Healthcare, Stock management, Perishable resource.

1 Introduction

The UBCO (Bio pharmaceutical Unit of Oncology Clinic of the hospital of Tours) produces around 150 chemotherapy drugs per day for three hospital units of Tours (France). Perishable resources (molecules) are needed to prepare the drugs, precisely one molecule type per Chemotherapy drug. Each molecule type has its own stability time (shelf life after opening) and cost. The UBCO aims to respect a production lead-time so that the patient waiting time is reduced and to minimize the production cost. Around 40 molecules types are used for the production. The cost per vial of each molecule type varies between few euros and more than 1200€. The production is a three-step process: sterilization of the resources, preparation of the drug inside an isolator by an operator and quality control of a sample by an automatic analyzer. In (Robbes *et. al.* 2019), a chemotherapy drug production is modelized as a Hybrid Flow-shop scheduling problem and minimizing the production and delivery delay is solved by a heuristic. However the heuristic presented does not take account of the costs induced by the waste of perishable resources. In this paper we propose a parallel machine modelization of the chemotherapy production scheduling problem (simplified model) and a matheuristic algorithm. The goal is to find a schedule for the chemotherapy production which minimizes the total cost of the perishable resources.

2 Problem definition

A chemotherapy drug production plant is composed of a set I of parallel identical isolators, each one is composed by m work stations (machines). This production can be modelized as a parallel machines scheduling problem where machines are gathered into groups corresponding to the isolators. A job can be processed on any isolator and on any machine of the isolator. The chemotherapy drugs are the jobs and the molecules used for the preparation are perishable resources used by the jobs. The preparation of one chemotherapy drug j uses only one type of molecule. The same vial can be used for several jobs. When one vial is assigned to two jobs on two different isolators, the sterilization time σ has to be taken into account. Each job j is also characterized by a preparation processing time p_j and a required resource quantity q_j of molecule type μ_j. Each perishable resource $r \in R$ is stored in vials of volume V_r with a price of $cost_r$ and has a stability time γ_r. We consider a planning horizon of $|D|$ consecutive days, knowing that no drug can be produced during the week-end. The scheduling variables are the starting time s_j (day and hour) and completion time C_j of each job j. The vial index assigned to the job j is noted f_j. We denote by z_r the cost induced by the perishable resource r and LB_r is the lower bound of z_r. The problem is to assign the jobs to the machines, to set their starting times and to assign a vial of perishable resource to each job, in order to minimize the total cost.

3 Matheuristic method

We propose a matheuristic method to solve this problem. We first generate an initial solution where jobs are assigned to the machines. Then, the algorithm is based on a Gradient Descent algorithm with a Tabu list, which explores neighborhoods for the assignment of jobs, and for the sequencing of jobs on the machines. The assignment of the vials to the jobs is done, molecule type per molecule type, by solving a Bin Packing problem with Conflicts. To compute the initial solution, we sort the molecule types by decreasing $cost_r$. Then, for each molecule type in this order, we assign all the jobs using the molecule type to the minimum possible number of machines. This behaviour tends to allow more possibilities of vial assignment by reducing the number of jobs using the same molecule type at the same time. Grouping the jobs using a same molecule type on few days reduces the number of vial assignment incompatibilities due to the molecule stability time.

3.1 Assignment to vials: Bin Packing problem with Conflicts

The Bin Packing problem with Conflicts (BPC) is a variant of the Bin Packing problem, where a conflict graph defines which pair of jobs cannot be packed together in the same bin. The one-dimensional version was introduced in (Jansen and Öhring 1997).

A conflict graph is generated from conflict rules. An edge corresponds to an incompatibility between two jobs to use the same vial. There are 3 conflicts cases for two jobs j and j' using the same type of perishable resource ($\mu_j = \mu_{j'}$):

1. **Machine overlap conflict**: the jobs are performed by different machines at the same time, i.e. $\max(s_j, s_{j'}) \; < \; \min(C_j, C_{j'})$
2. **Isolator overlap conflict**: the jobs are performed in different isolators and the time gap is smaller than the sterilization time, i.e. $\max(s_j, s_{j'}) \; - \; \sigma \; < \; \min(C_j, C_{j'})$
3. **Stability conflict**: the jobs are performed with a time gap greater than the stability time of the molecule type, i.e. $\max(C_j, C_{j'}) \; - \; \min(s_j, s_{j'}) \; > \; \gamma_{\mu_j}$

The BPC instance built from this conflict graph is solved using an Integer Linear Programming (ILP) modelization based on the new formulation of the Bin Packing Problem proposed in (Hadj Salem and Kieffer 2019).

3.2 Local permutation: Non Destructive Permutation

The Gradient Descent algorithm with a Tabu list executes as much as possible pairwise permutations on the schedule. Evaluating the schedule after each permutation is time consuming. To avoid a costly neighbourhood exploration, we choose to focus the on permutations which are consistent with the vials assignments (BPCs solutions) and do not need to solve the BPCs again.

We define a Non Destructive Permutation (NDP) as a job pairwise permutation respecting some rules. Let consider two jobs j and j' and a vial f containing the molecule type μ_j that is assigned to other jobs. The goal to assign vial f ($f \neq f_j$) to j in order to reduce the number of vials of μ_j molecule type. The remaining volume of vial f should be greater than q_j.

The idea is to swap job j with j' in the schedule, in order to modify the conflict graph of the molecule type μ_j. A permutation between j and j' is Non Destructive if the swap between j and j' does not modify the vial assigned to j' (the assignment of f'_j to j' remains consistent). The following constraints have to be satisfied.

– $z_{\mu_j} \neq LB_{\mu_j}$ i.e. it is possible to reduce the number of vials.
– the vial f_j used for j is not totally used.
– $p_j = p_{j'}$ i.e. the permutation does not impact the rest of the schedule.
– $\mu_j \neq \mu_{j'}$ i.e. the two jobs are not using the same molecule type.
– for all j" as $f_{j"} = f_{j'}$, swapping j and j' does not generate conflict between j' and j".

To avoid any cyclic behaviour, all jobs permuted are added in a Tabu list. Any new NDP that implies a job in the Tabu list is forbiden.

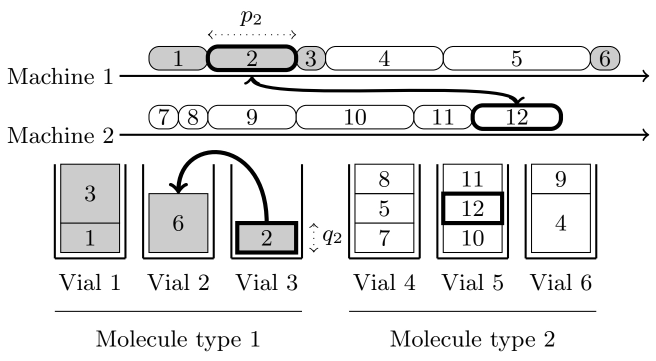


Fig. 1. Illustration of a Non Destructive Permutation

Fig. 1 shows a possible NDP between the jobs 2 and 12 by changing the assigned Vial $f_2 = 3$ to $f_2 = 2$. This NDP reduces the number of opened vial of Molecule type 1. Note that even if the job 12 is swapped with 2, no conflict will be created between jobs assigned to Vial 5.

4 Computational Experiments

Instances have been generated based on the real-life application at UBCO. The study have been limited to 6 types of perishable resources. For each perishable resource r, the vial volume is set to $V_r = 10$. For each job j, q_j is a random value in $\{1, \ldots, 10\}$ and p_j is a random value between 5, 10 and 15 minutes. The sterilization time σ is set to 15 minutes. A production day is 8 hours of work and a week has 5 production days. We consider a time horizon $|D|$ of 2 and 3 and 4 weeks. The number of isolators $|I|$ belongs to 2,3 and the number of machines per isolator is equal to $m = 2$.

Table 1. Perishable resources characteristics

r	1	2	3	4	5	6
Stability time γ_r	1 day	7 days	9 days	30 days	30 days	30 days
Cost per vial (€)	380	35	1195	25	405	740
% of the jobs requiring r	9	13	2	56	9	11

Table 1 presents the 3 characteristics of the 6 perishable resources: the stability times, the cost per vial and the percentages of job using the resource. We fix the computation times limit to solve an ILP to 120 seconds, with Gurobi Optimizer. The experimentation was performed on 10 instances for each instance size. The size of an instance is defined by the schedule horizon (number of days or weeks to schedule), the number of isolators $|I|$ and m the number of work stations per isolator. The number of jobs depends on the jobs processing times, and we have $\sum_{j=1}^{n} p_j = 0.9 \times |D| \times 8 \times 60 \times |I| \times m$.

Table 2. Average Gap and average number of NDP depending of the initialization

Instance size		Gap%				Nb NDP	
Horizon $\|D\|$	$\|I\|$	Grouping		Random		Grouping	Random
		before GD	after GD	before GD	after GD		
2 weeks	2 isolators	0.21	**0.17**	0.37	0.36	0.7	0.7
2 weeks	3 isolators	0.97	0.81	0.77	**0.62**	4.2	2.4
3 weeks	2 isolators	0.95	**0.78**	1.39	1.02	2.4	7.3
3 weeks	3 isolators	2.12	**1.53**	2.93	1.85	10.3	15.3
4 weeks	2 isolators	2.04	**1.57**	3.58	1.99	6.4	21.3
4 weeks	3 isolators	2.29	1.65	2.92	**1.62**	13.3	24.8

In Table 2, the gap is defined by $100 * \frac{\sum z_r - \sum LB_r}{\sum LB_r}$, two initialisation methods are compared, the proposed Grouping scheduling and a Random one. The column "before GD" represents the average gap of the initial solution and the column "after GD" represents the average gap at the end of the Grandient Descent. "Nb NDP" is the average number of NDP performed by the Gradient Descent algorithm. The average gap between our matheuristic and the lower bound is very low. It seems that the initialisation method does not have an important impact on the results after the GD but the number of NDP is greater for a random initialisation than the Grouping heuristic. The gap of a random initialisation is quite small even before the GD, which means that for our instances, the most important part to reduce the total cost is not the scheduling but the vial assignments. Further experimentations have to be done to confirm this claim.

5 Conclusion and perspectives

In this paper, we propose a matheuristic algorithm to minimize the costs induced by perishable resources waste in a chemotherapy drugs production unit. We modelize the production as a parallel machine scheduling problem and present a Gradient Descent algorithm with a Tabu list where the perishable resources assignment is done by solving several Bin Packing problem with conflicts. The computational results shows that with a short production time horizon, we obtain solutions with a production cost near to its lower bound. However, a large planning horizon needs multiple Gradient Descent steps before finding a local minimum. The perspectives are to use the branch and price proposed in (Sadykov and Vanderbeck 2012) to decrease the computation time to solve each Bin Packing problem.

References

Hadj Salem K., Y. Kieffer, 2019, “Nouvelle formulation en PLNE pour le problème classique du Bin Packing", *ROADEF 2019*.

Jansen K., S. Öhring, 2012, “Approximation Algorithms for Time Constrained Scheduling", *Information and Computation*, Vol. 132, pp. 85-108.

Robbes A., Y. Kergosien and J-C. Billaut, 2019, “Multi-level heuristic to optimize the chemotherapy production and delivery", *Health Care Systems Engineering: HCSE 2019*.

Sadykov R., F. Vanderbeck, 2012, “Bin Packing with Conflicts: A Generic Branch-and-Price Algorithm", *INFORMS Journal on Computing*, Vol. 25.

A comparison of proactive and reactive scheduling approaches for the RCPSP with uncertain activity durations

Pedram Saeedi, Erik Demeulemeester

KU Leuven, Faculty of Economics and Business, Department of Decision Sciences and Information Management, Leuven (Belgium)
pedram.saeedi@kuleuven.be
erik.demeulemeester@kuleuven.be

Keywords: RCPSP, proactive and reactive scheduling, uncertain activity durations.

1 Abstract

In this paper, we study the performance of state-of-the-art robust solution approaches towards solving the resource-constrained project scheduling problem (RCPSP) with uncertain activity durations. This study addresses approaches in search of a stable project plan that usually use a two-stage procedure, which first creates an (optimal) baseline schedule and then provides changes in the schedule whenever the realized duration of an activity forces us to deviate from the baseline schedule. We discuss the fundamentals of these approaches and possible directions for future research.

2 Introduction

Although the RCPSP has been studied for decades, most of the work done in this field investigated the problem in a static and deterministic environment (Padalkar & Gopinath 2016). In reality, however, the activity durations in a project are subject to significant uncertainty due to various internal or external factors (Zhu et al. 2005, Atkinson et al. 2006): variations in resource requirements, activities taking more or less time than their initial estimation, changes in the budget, etc. This may result in a considerable project overcost and/or overtime (Flyvbjerg 2013). The efforts to address uncertainty in the study of the RCPSP are relatively small compared to those for the deterministic version. In recent decades there have been several studies discussing project scheduling dealing with uncertainties (one can refer to Herroelen & Leus (2005), Demeulemeester & Herroelen (2011) and Hazir & Ulusoy (2019) for an overview of these studies).

Different approaches are proposed to address the uncertainty in project scheduling (Herroelen & Leus 2005). From a general point of view, we can divide most of these efforts into two categories: the stochastic RCPSP (SRCPSP, also referred to as dynamic scheduling) and proactive and reactive project scheduling. The SRCPSP uses scheduling policies or scheduling strategies to dynamically make certain decisions at certain moments. This implies that no baseline schedule is created prior to the start of the project and the schedule is generated gradually by using the aforementioned policies. This results in a lack of robustness in the SRCPSP approach. In the proactive and reactive approaches, on the other hand, a two-stage process is used. In the first stage, a robust baseline schedule is produced, meaning that this approach tries to generate a baseline schedule that tolerates a certain type of uncertainty as well as possible. In the second stage, reactive measures are taken whenever a conflict occurs that cannot be absorbed by the baseline schedule. The main drawback of this two-stage approach is that the final schedule (and its *quality*) is heavily based on the initial baseline schedule. However, Davari & Demeulemeester (2019*a*) have recently introduced an integrated proactive and reactive approach for the RCPSP to address this drawback.

3 Methodology

The goal of the research in this paper is to study the performance of the proactive and reactive scheduling approaches towards solving the RCPSP. There have been several robustness measures introduced in the scheduling literature to evaluate the robustness of a project schedule. Herroelen & Leus (2005) divide these measures into the two categories of *solution robustness* and *quality robustness*. Solution robustness (also referred to as schedule stability) is concerned with the difference between the baseline schedule and the realized schedule. Quality robustness, on the other hand, is concerned with the insensitivity of the objective value of the baseline schedule against distortions. For an overview and comparison of different robustness measures used in the literature we refer to (Herroelen 2007) and (Khemakhem & Chtourou 2013).Many papers take advantage of proactive and reactive scheduling to deal with uncertainty in the RCPSP. Leus & Herroelen (2004) developed a linear programming model that allowed an increase in the duration of a single activity with having only a single resource type in the problem. Al-Fawzan & Haouari (2005) studied the optimization of the makespan and the solution robustness by introducing a bi-objective model for the RCPSP. Deblaere et al. (2007) consider the RCPSP with uncertainty in activity durations and try to solve the problem by providing solutions on resource allocation that would maximize schedule robustness. Van de Vonder et al. (2008) developed new heuristics such as the *starting time criticality* heuristic to find solutions to the proactive RCPSP. Lambrechts et al. (2008) focus on uncertainty in resource availability and propose eight proactive and three reactive strategies to solve the problem. They apply two approaches for generating a baseline schedule, including using the highest *cumulative instability weight* (CIW) measure to determine an ordered list of activities for scheduling. Lamas & Demeulemeester (2016) introduced a MIP formulation for the chance-constrained RCPSP (C-C RCPSP). Bruni et al. (2017) take advantage of the adjustable robust optimization approach. In the adjustable robust optimization, part of the variables in the problem are determined before the realization of the uncertainty and the others can be adjusted based on the realization of the uncertainty Davari & Demeulemeester (2019*a*) developed an integrated proactive and reactive approach towards the RCPSP and they continued their study in Davari & Demeulemeester (2019*b*) by investigating different classes of reactions and evaluating their contributions in optimal proactive and reactive policies (PR-policies).

This study is conducted in two phases. In the first phase, a thorough literature review is done to address different proactive and reactive approaches and solutions. This phase also covers different single or composite measures of robustness introduced in the literature as well as different policies on activity starting times and schemes on resource allocation. Considering the various combinations of these factors alongside different proactive and reactive approaches in the RCPSP literature, there has not been a clear comparison of how well different studies perform compared to each other. This is where the second phase of this study comes to action. We will conduct a computational comparison of the proactive and reactive RCPSP approaches to see how well they perform in the same environment and with the same input parameters. This will give us the opportunity to see the current gaps in the literature, for instance, to see on what type of project *settings* proactive and reactive approaches cannot still achieve satisfactory results or cannot provide a good solution in terms of robustness. Another contribution would be getting an overview of promising proactive and reactive approaches and expansions to achieve further better results for different RCPSP problems, for instance, to identify promising (or optimal) scheduling policies for particular versions of the RCPSP and to compare the *quality* of different scheduling policies. This helps us to identify the potential directions towards future studies.

References

Al-Fawzan, M. A. & Haouari, M. (2005), 'A bi-objective model for robust resource-constrained project scheduling', *International Journal of production economics* **96**(2), 175–187.

Atkinson, R., Crawford, L. & Ward, S. (2006), 'Fundamental uncertainties in projects and the scope of project management', *International journal of project management* **24**(8), 687–698.

Bruni, M. E., Pugliese, L. D. P., Beraldi, P. & Guerriero, F. (2017), 'An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations', *Omega* **71**, 66–84.

Davari, M. & Demeulemeester, E. (2019*a*), 'The proactive and reactive resource-constrained project scheduling problem', *Journal of Scheduling* **22**(2), 211–237.

Davari, M. & Demeulemeester, E. (2019*b*), 'Important classes of reactions for the proactive and reactive resource-constrained project scheduling problem', *Annals of Operations Research* **274**(1-2), 187–210.

Deblaere, F., Demeulemeester, E., Herroelen, W. & Van de Vonder, S. (2007), 'Robust resource allocation decisions in resource-constrained projects', *Decision Sciences* **38**(1), 5–37.

Demeulemeester, E. & Herroelen, W. (2011), 'Robust project scheduling', *Foundations and Trends® in Technology, Information and Operations Management* **3**(3–4), 201–376.

Flyvbjerg, B. (2013), 'Over budget, over time, over and over again: Managing major projects'.

Hazir, O. & Ulusoy, G. (2019), 'A classification and review of approaches and methods for modeling uncertainty in projects', *International Journal of Production Economics* p. 107522.

Herroelen, W. (2007), Generating robust project baseline schedules, *in* 'OR Tools and Applications: Glimpses of Future Technologies', INFORMS, pp. 124–144.

Herroelen, W. & Leus, R. (2005), 'Project scheduling under uncertainty: Survey and research potentials', *European journal of operational research* **165**(2), 289–306.

Khemakhem, M. A. & Chtourou, H. (2013), 'Efficient robustness measures for the resource-constrained project scheduling problem', *International Journal of Industrial and Systems Engineering* **14**(2), 245–267.

Lamas, P. & Demeulemeester, E. (2016), 'A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations', *Journal of Scheduling* **19**(4), 409–428.

Lambrechts, O., Demeulemeester, E. & Herroelen, W. (2008), 'Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities', *Journal of scheduling* **11**(2), 121–136.

Leus, R. & Herroelen, W. (2004), 'Stability and resource allocation in project planning', *IIE transactions* **36**(7), 667–682.

Padalkar, M. & Gopinath, S. (2016), 'Six decades of project management research: Thematic trends and future opportunities', *International Journal of Project Management* **34**(7), 1305–1321.

Van de Vonder, S., Demeulemeester, E. & Herroelen, W. (2008), 'Proactive heuristic procedures for robust project scheduling: An experimental analysis', *European Journal of Operational Research* **189**(3), 723–733.

Zhu, G., Bard, J. F. & Yu, G. (2005), 'Disruption management for resource-constrained project scheduling', *Journal of the Operational Research Society* **56**(4), 365–381.

Towards the Optimisation of the Dynamic and Stochastic Resource-Constrained Multi-Project Scheduling Problem

Ugur Satic, Peter Jacko and Christopher Kirkbride

Lancaster University, United Kingdom
u.satic@lancaster.ac.uk

Keywords: Dynamic programming, resource constraint, project scheduling, approximate dynamic programming

1 Introduction

Project scheduling is a rich and widely studied research area, most of the literature focuses on deterministic problems such as the *resource-constrained project scheduling problem* (RCPSP) and the *resource-constrained multi-project scheduling problem* (RCMPSP). The goals of RCPSP and RCMPSP typically minimise the total or average project completion times, which can be overly simplistic as it ignores the fact that projects may have different importance or rewards and may have deadlines with associated tardiness penalties. Solutions for such problems are deterministic schedules that show planned starting times of tasks.

The project execution frequently gets affected by many uncertainties; and project completion times deviate from the planned schedule. One uncertain element of project scheduling is task durations. Another uncertain element is stochastic resource availability. Resource availability could be affected by maintenance, breakdowns, personnel days off. In the literature, the stochastic equivalents of RCPSP and RMCPSP are called the *stochastic* RCPSP and the *stochastic* RCMPSP respectively. Most of the research in this field concerns only stochastic task durations e.g. Bruni, Pugliese, Beraldi & Guerriero. (2018). Only a few researched stochastic resource availability e.g. Wang, Chen, Mao, Chen & Li (2015).

Companies usually work on multiple projects at the same time to use their resources more effectively. On the arrival of a new project it should be added to execution as soon as possible without waiting for the completion of the previous schedule. Thus the arrival of a new project disrupts the previous schedules. The RCMPSP with random project arrivals is called *dynamic* RCMPSP (DRCMPSP). In the literature, the DRCMPSP considers the random arrival of new projects assuming all other project elements are deterministic.

Only a limited number of research considered the both dynamic project arrivals and stochastic task durations together e.g. Satic, Jacko & Kirkbride (2020). This problem is called the *dynamic and stochastic* RCMPSP. The dynamic and stochastic RCMPSP aims to find optimal schedules or scheduling policies that maximise the expected total discounted or time-average project reward minus the costs. In this paper we will focus on the former. We consider the dynamic and stochastic RCMPSP with project arrivals and stochastic task durations. We model the problem as an infinite-horizon discrete-time *Markov decision process* (MDP).

2 Modelling Framework

In this study, we assume the dynamic and stochastic RCMPSP contains J project types where the project type, j, determines the characteristics such as a new project arrival

Table 1. Example decision state for a problem with two project types, each with three tasks.

	Tasks	Remaining due date
Project type 1 :	$x_{1,1}$ $x_{1,2}$ $x_{1,3}$	d_1
Project type 2 :	$x_{2,1}$ $x_{2,2}$ $x_{2,3}$	d_2

probability (λ_j), number of tasks (I_j), project network, resource requirement per unit time of a task i ($b_{j,i}$), task completion probabilities ($\gamma_{j,i}$), minimal possible completion time ($t^{min}_{j,i}$), maximal possible completion time ($t^{max}_{j,i}$), project due date (F_j), reward (r_j) and tardiness cost (w_j). We model this problem as an infinite horizon *Discrete Time Markov Decision Process* (DT-MDP) which is defined by five elements: time horizon, decision state space, action set, transition function and profit function.

In a DT-MDP, the decision maker takes an action a for a decision state s at a decision epoch that occur at fixed intervals. The period between two consecutive decision epochs is a single unit of time, called a period.

The system information at a decision epoch is called a decision state (s). An example decision state for a two project-type problem is given in Table 1, where $x_{j,i}$ is the remaining task processing time to the latest possible task completion time (t_j, i^{max}) and d_j is the remaining time until the due date. For tasks awaiting processing we set $x_{j,i}$ to -1.

The action a represents the processing decision of pending tasks ($x_{j,i} = -1$) of the decision state (s). An action must satisfy two conditions: there must be enough resources (B) available to begin processing these new tasks ($\sum_{j=1}^{J}\sum_{i=1}^{I_j} b_{j,i}(\mathcal{I}\{a_{j,i}=1\} + \mathcal{I}\{x_{j,i} > 0\}) \leq B$) and all predecessor tasks ($\mathcal{M}_{j,i}$) of task i must be completed ($\sum_{m\in\mathcal{M}_{j,i}} x_{j,m} = 0$).

After the selected action a is applied in the decision state s, the system transforms from one state to another (s') at the next decision epoch according to a transition function $P(s'|s,a)$.

$$P(s'|s,a) = \prod_{j=1}^{J}\prod_{i=1}^{I_j} P(x'_{j,i}|x_{j,i}+a_{j,i}) \tag{1}$$

$$P(x'_{j,i}|x_{j,i}+a_{j,i}) = \begin{cases} \lambda_j\gamma_{j,i}(x_{j,i}+a_{j,i}), & \text{for } 1 \leq x_{j,i}+a_{j,i} \leq 1 + t^{max}_{j,i} - t^{min}_{j,i}, \\ & x'_{j,i} = -1, i = I_j \\ (1-\lambda_j)\gamma_{j,i}(x_{j,i}+a_{j,i}), & \text{for } 1 \leq x_{j,i}+a_{j,i} \leq 1 + t^{max}_{j,i} - t^{min}_{j,i}, \\ & x'_{j,i} = 0, i = I_j \\ \gamma_{j,i}(x_{j,i}+a_{j,i}), & \text{for } 1 \leq x_{j,i}+a_{j,i} \leq 1 + t^{max}_{j,i} - t^{min}_{j,i}, \\ & x'_{j,i} = 0, i < I \\ 1-\gamma_{j,i}(x_{j,i}+a_{j,i}), & \text{for } 1 \leq x_{j,i}+a_{j,i} \leq 1 + t^{max}_{j,i} - t^{min}_{j,i}, \\ & x'_{j,i} = x_{j,i}+a_{j,i}-1 \\ \lambda_j, & \text{for } x_{j,i}+a_{j,i} = 0, x'_{j,i} = -1, i = I_j \\ 1-\lambda_j, & \text{for } x_{j,i}+a_{j,i} = 0, x'_{j,i} = 0, i = I_j \\ 1, & \text{for } x_{j,i}+a_{j,i} = 0, x'_{j,i} = 0, i < I_j \\ 1, & \text{for } x_{j,i}+a_{j,i} > 1 + t^{max}_{j,i} - t^{min}_{j,i}, \\ & x'_{j,i} = x_{j,i}+a_{j,i}-1 \\ 1, & \text{for } x_{j,i}+a_{j,i} = -1, x'_{j,i} = -1 \end{cases} \tag{2}$$

The profit function ($R_{s,a}$) is the sum of rewards (r_j) of completed projects in the current period minus the tardiness cost of any late completions.

$$\begin{aligned} R_{s,a} =& \sum_{j=1}^{J} r_j \mathbb{E}\Big[\mathcal{I}\big\{\big(x_{j,I} \geq 1 \vee (x_{j,I} = -1 \wedge a_{j,I} = 1)\big) \wedge x'_{j,I} \leq 0\big\}\Big] \\ & - \sum_{j=1}^{J} w_j \mathbb{E}\Big[\mathcal{I}\big\{\big(x_{j,I} \geq 1 \vee (x_{j,I} = -1 \wedge a_{j,I} = 1)\big) \wedge x'_{j,I} \leq 0 \wedge d_j = 0\big\}\Big]. \end{aligned} \quad (3)$$

3 Solution Methods

We compare six different solution approaches which are: a dynamic programming algorithm (DP), an approximate dynamic programming algorithm (ADP), an optimal reactive baseline algorithm (ORBA), a genetic algorithm (GA), a rule-based algorithm (RBA) and a worst decision algorithm (WDP).

The dynamic programming value iteration algorithm is used to determine an optimal policy that maximises the discounted long-time profit. We tested computational limitations of DP in the dynamic and stochastic RCMPSP.

ADP replaces the true value function of the Bellman's equation with an approximate one to overcome the curse of dimensionality problem of DP. We built a linear approximate value function with two state information as decision variables and weighted them using coefficients. The decision variables are the number of the period spent on processing each type of projects and the number of allocated resources between project types.

We used three reactive scheduling heuristics which are ORBA, GA and RBA. GA and RBA methods are popular for both dynamic and static RCMPSP problems thus we added them our comparison to evaluate their performance. Reactive scheduling methods do not consider the future uncertainties while generates schedules; then they fix these schedules at each distribution (Rostami, Creemers & Leus 2018).

Reactive scheduling methods generate a new baseline schedule and convert it to an action for each state. ORBA and GA seek to maximise the profit and uses the total completion time as tiebreakers between the schedules with equal reward. If several schedules have equal rewards and equal completion times, the models prioritise smallest numbered project type. We used a population of 100 with 100 generations in GA. RBA uses the longest processing time first rule to schedule. If several tasks have equal duration, RBA selects one at random.

WDP, using a dynamic programming value iteration algorithm, seeks a non-idling policy to minimise the average profit per unit time. We used this method in our comparison to show the profit of the worst non-idling policy.

4 Algorithm evaluation

All tests are performed on a desktop computer with Intel i5-6500T CPU with 2.50 GHz clock speed and 32 GB of RAM. JuliaPro 1.3.1.2 is used for coding the model, solution approaches and problems. Three dynamic and stochastic RCMPSPs are generated and tested consecutively from 1% to 90% project arrival probabilities, incremented by 10%.

Table 2 shows the discounted long-term profits of six algorithms with 95% discount rate. The policies of reactive scheduling algorithms are closer to optimum with low project arrival rates such as 1% where system is closer to static and they diverge from the optimum as arrival probability increases. ADP suffers at low arrival probabilities but produces equal or better results to ORBA after %30 arrival probability.

Table 2. Discounted long-term profits

Two projects and two tasks problem										
λ_j	1%	10%	20%	30%	40%	50%	60%	70%	80%	90%
DP	6.16	16.23	22.55	26.32	28.76	30.46	31.70	32.64	33.38	33.96
ADP	3.42	13.45	18.02	22.04	26.57	27.91	28.86	29.54	30.05	30.44
ORBA	5.99	15.64	21.31	24.54	26.55	27.89	28.83	29.51	30.02	30.41
GA	5.74	14.70	18.54	20.73	19.64	24.97	20.60	22.74	21.62	20.85
RBA	5.49	13.45	16.36	16.99	16.84	16.44	16.00	15.59	15.26	15.04
WDA	5.38	12.96	15.37	15.73	15.50	15.16	14.78	14.26	13.68	12.96
Two projects and three tasks problem										
DP	10.17	18.79	23.08	25.43	26.91	27.92	28.65	29.20	29.63	29.98
ADP	6.85	18.51	22.70	25.12	26.53	27.47	28.14	28.64	29.03	28.89
ORBA	10.15	18.68	22.86	25.12	26.53	27.47	28.14	28.64	29.03	29.34
GA	10.15	18.68	22.86	25.12	26.53	27.47	28.14	28.64	29.03	29.34
RBA	10.11	18.42	22.37	24.44	25.69	26.50	27.05	27.45	27.75	27.99
WDA	9.79	17.13	20.07	21.44	22.20	22.66	22.95	23.15	23.29	23.40
Three projects and two tasks problem										
DP	15.14	28.36	32.63	34.92	36.98	38.74	40.19	41.37	42.33	43.12
ADP	10.50	25.48	29.15	31.83	34.23	36.28	37.99	39.39	41.23	42.03
ORBA	14.88	27.48	30.73	31.75	32.31	32.73	33.10	33.43	33.74	34.03
GA	14.63	26.79	29.37	29.84	29.88	29.97	30.65	30.60	31.29	31.62
RBA	14.61	25.96	28.64	29.96	31.21	32.48	33.73	34.93	36.06	37.11
WDA	13.82	22.59	23.31	23.26	23.14	23.05	23.04	23.10	23.23	23.40

5 Conclusion

We consider the RCMPSP with uncertain project arrivals and stochastic task durations as an infinite-horizon DT-MDP. We used six approaches and compared their results. We also tested the computational limits of the DP on the dynamic and stochastic RCMPSPs. We observed that DP suffers from the curse of dimensionality even for the small size problems and results of reactive scheduling methods deteriorate compared to optimum results as stochasticity increases. ADP performs similar or better than ORBA, which is the second best method, after 30% arrival probability. More detailed description of the model and more extensive results can be found at Satic et al. (2020).

References

Bruni, M. E., Pugliese, L. D. P., Beraldi, P. & Guerriero., F. (2018), A two-stage stochastic programming model for the resource constrained project scheduling problem under uncertainty, *in* 'Proceedings of the 7th International Conference on Operations Research and Enterprise Systems (ICORES)', Vol. 1, INSTICC, SciTePress, pp. 194–200.

Rostami, S., Creemers, S. & Leus, R. (2018), 'New strategies for stochastic resource-constrained project scheduling', *Journal of Scheduling* **21**(3), 349–365.

Satic, U., Jacko, P. & Kirkbride, C. (2020), 'Performance evaluation of scheduling policies for the dynamic and stochastic resource-constrained multi-project scheduling problem', *International Journal of Production Research*, Advanced online publication, doi = 10.1080/00207543.2020.1857450.

Wang, X., Chen, Q., Mao, N., Chen, X. & Li, Z. (2015), 'Proactive approach for Stochastic RCMPSP based on multi-priority rule combinations', *International Journal of Production Research* **53**(4), 1098–1110.

An FPTAS for Scheduling with Piecewise-Linear Nonmonotonic Convex Time-Dependent Processing Times and Job-Specific Agreeable Slopes

Helmut A. Sedding

Institute of Data Analysis and Process Design
ZHAW Zurich University of Applied Sciences, Winterthur, Switzerland

Keywords: Time-dependent scheduling, Piecewise-linear convex nonmonotonic processing times with agreeable ratios of basic processing time and slopes, FPTAS

1 Introduction

The time-dependent scheduling branch is concerned with processing times that are a function of job start time (Gawiejnowicz 2020a, 2020b). These effectively superimpose an additional layer of complexity compared to fixed processing times. For example, an interchange of adjacent jobs typically yields a change in their processing time. Additionally, this change affects all subsequent jobs' processing time. Thus, applying an adjacent job interchange argument is more involved. Hence, it can be challenging already to sequence a set of jobs without idle time on a single machine with the objective of minimizing the makespan $C_{\max}$. This problem is considered in this study for a processing time function that can attain a nonmonotonic convex shape. This shape distinguishes it from most existing literature, which considers monotonic convex shapes (Gawiejnowicz 2020a, 2020b).

Time-dependent effects of a job's start time can be additive, multiplicative, or combined (Strusevich and Rustogi 2017). The effect studied in this study is additive, extending the earliest work in this field in Shafransky (1978). Here, each job j has a *basic processing time* ℓ_j, and a *penalty function* f_j of start time t. They are added to yield processing time

$$p_j(t) = \ell_j + f_j(t). \tag{1}$$

This study considers the nonmonotonic piecewise-linear job-specific penalty function

$$f_j(t) = \max\{-a_j\,(t-\tau),\, b_j\,(t-\tau)\} \tag{2}$$

for a given common *ideal start time* τ and rational valued *slopes* $0 \le a_j \le 1$ and $b_j \ge 0$. The jobs are required to have *agreeable ratios of basic processing time and slopes*, i.e., there must exist a sequence all jobs that fulfills condition

$$\ell_i a_j \ge \ell_j a_i \quad \text{and} \quad \ell_i b_j \ge \ell_j b_i \quad \text{for any job } i \text{ sequenced before any job } j. \tag{3}$$

The main challenge of this problem is to decide which jobs shall start before τ, and which jobs shall start at or after τ. This decision is NP-hard already for uniform slopes $a = a_j$, $b = b_j$, as shown in Sedding (2020b, 2020c) by reduction from Even-Odd Partition.

In this study, a fully polynomial time approximation scheme (FPTAS) is given for this problem, which enables to sequence the jobs both quickly and with an error bound.

In practice, this enables fast reaction times in computationally sequencing a worker's tasks at the moving conveyor line, e.g., in car assembly. Here, each operation involves leaving the work point, walking along the line to a supply point, and return. The occurring walking time is, according to Sedding (2020a), adequately depicted by the studied penalty function (2). Individual slopes with agreeable ratios (3) reflect task-specific walking velocities that occur, e.g., when carrying additional weight.

2 Related problems

Existing time-dependent scheduling literature with an additive effect mostly considers penalty functions that are monotonic, hence either nondecreasing (often called *deterioration effect*) or nonincreasing (sometimes called *learning effect*). The main advantage of monotonic penalty functions is that their monotonic effect is recursive. Starting some job earlier does not increase its own completion time, and neither those of successors.

An interesting discovery is that it is possible for any shape of uniform ($f = f_j$) monotonic penalty functions to find an optimal sequence in polynomial time: by sorting the jobs with respect to their basic processing time (Melnikov and Shafransky 1979).

A well-known polynomial case with job-specific penalty functions has the non-increasing proportional-linear $f_j(t) = -a_j\, t$ with $0 \le a_j \le 1$ (hence, $b_j = 0$ and $\tau = 0$ in (2)). Here, any sequence that fulfills condition (3) is optimal (Ho, Leung and Wei 1993). The symmetric case is the non-decreasing proportional-linear $f_j(t) = b_j\, t$ with $b_j \ge 0$ (hence, $a_j = 0$ and $\tau = 0$ in (2)). Here, an optimal sequence fulfills condition

$$\ell_i a_j \le \ell_j a_i \quad \text{and} \quad \ell_i b_j \le \ell_j b_i \quad \text{for any job } i \text{ sequenced before any job } j \tag{4}$$

(Shafransky 1978, Gupta and Gupta 1988, Browne and Yechiali 1990, Gawiejnowicz and Pankowska 1995).

These results are the basis for the monotonic piecewise-linear case, expressed by penalty functions as in (2) but restricting slopes either to $a = a_j = 0$, or to $b = b_j = 0$. Then, the jobs need to be partitioned into two sides around τ, which is an NP-hard problem (Kononov 1997, Kubiak and van de Velde 1998, Cheng, Ding, Kovalyov, Bachman and Janiak 2003) that permits FPTASs (Kovalyov and Kubiak 1998, Kovalyov and Kubiak 2012, Cai, Cai and Zhu 1998, Woeginger 2000, Ji and Cheng 2007, Halman 2020).

The non-monotonic penalty function case in (2) with symmetric slopes $a_j = b_j < 1$ is covered by the model in Kononov (1998) for all-zero basic processing times $\ell_j = 0$, and solved by ordering the jobs non-decreasingly with respect to a_j. Nonnegative $\ell_j \ge 0$ are first considered in Sedding and Jaehn (2014) for uniform symmetric slopes $a = a_j = b_j < 1$. A similar model is studied in Jaehn and Sedding (2016), a much more general model in Kawase, Makino and Seimi (2018). In Sedding (2020b, 2020c), the uniform slopes $a = a_j$, $b = b_j$ case is shown to be NP-hard. This model is extended to agreeable slope ratios (3) in this study, which is expanded in Sedding (2020b).

3 Sorting criteria

In the given problem, optimal sequences exhibit a certain sort order for the jobs that complete before or at the ideal start time τ (they are denoted by partial sequence S_1), and another for the jobs starting at or after τ (denoted by S_2). Then, the jobs in S_1 and S_2 effectively have proportional-linear penalty functions. Hence, the respective sum of processing times in S_1 and S_2 is minimized if the sorting criteria as described in section 2 hold, i.e., if S_1 fulfills condition (3) and S_2 fulfills (4) (Sedding 2018a, 2018b).

The sorting criteria on both S_1 and S_2 have implications on the construction of the FPTAS. In the monotonic case, an arbitrary sorting is possible for either S_1 (if all $a_j = 0$), or S_2 (if all $b_j = 0$) because the processing times in one of them are not time-dependent. Assuming these unchanging processing times are integer, one can also assume an integral, pseudopolynomial sum of processing times. Note that the known FPTAS for these problems utilize both properties, which leaves them unsuitable for the studied problem.

At least, the sort criteria in S_1 and in S_2 are related as follows. A sequence for condition (3) exists, and it is found in polynomial time. With this, the jobs are *agreeably* *renumbered* such that (3) holds for the job sequence $1, 2, \dots, n$. Then, it follows that there exists an optimal S_1 where the jobs are increasingly numbered, and an optimal S_2 where they are decreasingly numbered. Hence, S_1 and S_2 can be symmetrically sorted.

Please note that an exception to these sorting criteria might exist with a *straddler job* that starts before or at τ and completes at or after τ. In particular, such a job might not be the last job according to these criteria, i.e., the job number with number n.

4 Dynamic programming algorithm

The following dynamic programming algorithm solves the given problem exactly if a straddler job exists (if not, the instance corresponds to a proportional-linear penalty function case) and is already given. To choose it, the algorithm is repeatedly started with each of the jobs as a straddler job, then returning the best feasible schedule. In the following, straddler job χ has been chosen, the others are agreeably renumbered to $1, 2, \dots, n$.

Then, the dynamic program consists of stages 1 to n. Each stage $j \in \{1, \dots, n\}$ generates a set V_j of partial solutions. To generate this set, job j is inserted into all partial solutions of the preceding stage V_{j-1}, beginning with an empty solution in the first stage. Each partial solution represents two sequences S_1, S_2. Sequence S_1 represents the jobs to be completed before or at τ, and S_2 the jobs to be started at or after τ. A partial solution in V_{j-1} includes the jobs $1, \dots, j-1$ and is encoded by a nonnegative real vector $[x, y, z]$ of

- x, which specifies sequence S_1's completion time,
- y, which specifies the proportional increase (i.e., the value of the partial derivative) of sequence S_2's completion time for increasing its start time, and
- z, which specifies sequence S_2's sum of processing times if it is started at τ.

The initial partial solution set is $V_0 = \{[0, 1, 0]\}$. There are two ways to add job j to a partial solution: either appending j to S_1 (if possible), or prepending j at S_2. In this way, condition (3) is always upheld for S_1, and (4) for S_2. From any vector $[x, y, z] \in V_{j-1}$, appending job j to S_1 is possible if $x + p_j(x) \le \tau$, and it adds vector $[x + p_j(x),\ y,\ z]$ to V_j. Prepending j to S_2 adds another vector $[x,\ y \cdot (1 + b_j),\ y \cdot \ell_j + z]$ to V_j.

After the final stage n, the straddler job χ is inserted between S_1 and S_2. Given a vector $[x, y, z] \in V_n$ of the final stage, let χ start at x. If the completion time $C_\chi = x + p_\chi(x)$ of the straddler job is less than τ, the vector is discarded. Otherwise, the makespan $C_{\max} = \tau + y \cdot (C_\chi - \tau) + z$ of the solution vector is obtained. The smallest $C_{\max}$ of all $[x, y, z] \in V_n$ is the optimum makespan value $C^*_{\max}$ (assuming χ is an optimal straddler job). The according job sequence can be reconstructed by traveling back the corresponding partial solutions.

5 A fully polynomial time approximation scheme

The dynamic program is turned into an FPTAS by trimming the states using exponentially growing value bins as described in Woeginger (2000). For a given maximum relative error $\varepsilon \in (0, 1]$, define $\Delta = 1 + \frac{\varepsilon}{2n}$, and function $h(x) = \Delta^{\lceil \log_\Delta x \rceil}$ for any positive real x, which satisfies $x/\Delta < h(x) \le x \cdot \Delta$. After each stage j, the set of partial solutions V_j is trimmed: for any disjoint pair of vectors $[x, y, z] \in V_j$, $[x', y', z'] \in V_j$ where $x \le x'$, $h(y) \le h(y')$, and $h(z) = h(z')$, the vector $[x', y', z']$ is discarded. It can be shown that the number of vectors in V_n remains polynomially bounded in input size and $1/\varepsilon$ by $\mathcal{O}(n^3 \cdot \log(1 + b_{\max}) \cdot (\log \max\{\ell_{\max}, 1/b_{\max}\} + n \log(1 + b_{\max}))/\varepsilon^2)$ for $\ell_{\max} = \max_j \ell_j$, $b_{\max} = \max_j b_j$; and a bounded minimum makespan $C^{\text{approx}}_{\max} \le C^*_{\max} \cdot (1 + \varepsilon)$ is achieved.

References

Browne, S. and Yechiali, U.: 1990, Scheduling deteriorating jobs on a single processor, *Operations Research* **38**(3), 495–498.

Cai, J.-Y., Cai, P. and Zhu, Y.: 1998, On a scheduling problem of time deteriorating jobs, *Journal of Complexity* **14**(2), 190–209.

Cheng, T. C. E., Ding, Q., Kovalyov, M. Y., Bachman, A. and Janiak, A.: 2003, Scheduling jobs with piecewise linear decreasing processing times, *Naval Research Logistics* **50**(6), 531–554.

Gawiejnowicz, S.: 2020a, *Models and algorithms of time-dependent scheduling*, Monographs in Theoretical Computer Science, second edn, Springer, Berlin, Heidelberg.

Gawiejnowicz, S.: 2020b, A review of four decades of time-dependent scheduling: main results, new topics, and open problems, *Journal of Scheduling* **23**(1), 3–47.

Gawiejnowicz, S. and Pankowska, L.: 1995, Scheduling jobs with varying processing times, *Information Processing Letters* **54**(3), 175–178.

Gupta, J. N. D. and Gupta, S. K.: 1988, Single facility scheduling with nonlinear processing times, *Computers & Industrial Engineering* **14**(4), 387–393.

Halman, N.: 2020, A technical note: fully polynomial time approximation schemes for minimizing the makespan of deteriorating jobs with nonlinear processing times, *Journal of Scheduling* **23**(6), 643–648.

Ho, K. I.-J., Leung, J. Y.-T. and Wei, W.-D.: 1993, Complexity of scheduling tasks with time-dependent execution times, *Information Processing Letters* **48**(6), 315–320.

Jaehn, F. and Sedding, H. A.: 2016, Scheduling with time-dependent discrepancy times, *Journal of Scheduling* **19**(6), 737–757.

Ji, M. and Cheng, T. C. E.: 2007, An FPTAS for scheduling jobs with piecewise linear decreasing processing times to minimize makespan, *Information Processing Letters* **102**(2-3), 41–47.

Kawase, Y., Makino, K. and Seimi, K.: 2018, Optimal composition ordering problems for piecewise linear functions, *Algorithmica* **80**(7), 2134–2159.

Kononov, A. V.: 1997, On schedules of a single machine jobs with processing times nonlinear in time, *Discrete Analysis and Operational Research* **391**, 109–122.

Kononov, A. V.: 1998, Problems in scheduling theory on a single machine with job durations proportional to an arbitrary function, *Diskretnyĭ Analiz i Issledovanie Operatsiĭ* **5**(3), 17–37.

Kovalyov, M. Y. and Kubiak, W.: 1998, A fully polynomial approximation scheme for minimizing makespan of deteriorating jobs, *Journal of Heuristics* **3**(4), 287–297.

Kovalyov, M. Y. and Kubiak, W.: 2012, A generic FPTAS for partition type optimisation problems, *International Journal of Planning and Scheduling* **1**(3), 209.

Kubiak, W. and van de Velde, S. L.: 1998, Scheduling deteriorating jobs to minimize makespan, *Naval Research Logistics* **45**(5), 511–523.

Melnikov, O. I. and Shafransky, Y. M.: 1979, Parametric problem in scheduling theory, *Cybernetics* **15**(3), 352–357.

Sedding, H. A.: 2018a, On the complexity of scheduling start time dependent asymmetric convex processing times, *Proceedings of the 16th International Conference on Project Management and Scheduling*, Università di Roma “Tor Vergata”, Rome, Italy, pp. 209–212.

Sedding, H. A.: 2018b, Scheduling non-monotonous convex piecewise-linear time-dependent processing times, *2nd International Workshop on Dynamic Scheduling Problems*, Adam Mickiewicz University, Poznań, Poland, pp. 79–84.

Sedding, H. A.: 2020a, Line side placement for shorter assembly line worker paths, *IISE Transactions* **52**(2), 181–198.

Sedding, H. A.: 2020b, Scheduling jobs with a V-shaped time-dependent processing time, *Journal of Scheduling* **23**(6), 751–768.

Sedding, H. A.: 2020c, *Time-dependent path scheduling: Algorithmic minimization of walking time at the moving assembly line*, Springer Vieweg, Wiesbaden.

Sedding, H. A. and Jaehn, F.: 2014, Single machine scheduling with nonmonotonic piecewise linear time dependent processing times, *Proceedings of the 14th International Conference on Project Management and Scheduling*, TUM School of Management, Munich, Germany, pp. 222–225.

Shafransky, Y. M.: 1978, On optimal ordering in deterministic systems with tree-like partial serving order, *Proceedings of the Academy of Sciences of Belarus, Physics&Mathematics* **1978**(2), 120.

Strusevich, V. A. and Rustogi, K.: 2017, *Scheduling with time-changing effects and rate-modifying activities*, Springer, Cham.

Woeginger, G. J.: 2000, When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)?, *INFORMS Journal on Computing* **12**(1), 57–74.

An analysis of critical alternatives in the RCPSP-AS

Tom Servranckx[1] and Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
tom.servranckx@ugent.be, mario.vanhoucke@ugent.be
[2] Technology and Operations Management Area, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, UK

Keywords: Project scheduling, Alternative project structures, Network analysis.

1 Introduction

The most well-known problem in the field of project scheduling is the resource-constrained project scheduling problem (RCPSP) (Brucker *et al.* 1999). In the RCPSP, activities are scheduled as soon as possible given technological and renewable resource constraints with the objective of a minimal project makespan. It is assumed that the project structure is completely fixed and known prior to project scheduling, while the projects and the project environment are becoming increasingly complex. This raises the question whether a fixed project structure is realistic or preferred in a highly variable project environment. Therefore, several researchers have investigated scheduling problems in which there exist alternative ways to execute subsets of activities in the project, so-called alternative project structures (Kellenbrink and Helber 2015, Tao and Dong 2017, Tao and Dong 2018, Servranckx and Vanhoucke 2019a). Such problems typically consist of two subproblems: a selection and a scheduling subproblem. In the *selection subproblem*, a choice should be made between the different alternatives in the project structure. Subsequently, the activities corresponding to the selected alternatives should be scheduled in the *scheduling subproblem.* In this abstract, we will focus on the RCPSP with alternative project structures (RCPSP-AS) that extends the RCPSP by defining alternative ways to execute work packages (WPs) in the project (Servranckx and Vanhoucke 2019a).

The current studies in this research field present meta-heuristic solution approaches to solve both subproblems in a sequential or integrated way in order to rapidly generate high-quality schedules for case studies and/or analyse the impact of alternatives on the project scheduling objective. In general, these studies show that the existence of alternative execution modes for WPs improves the flexibility of project scheduling in highly complex and variable project environments. However, the added value of the inclusion of alternatives will be limited in case that the research efforts are limited to project scheduling alone. This is because the project scheduling process will reveal the best set of alternatives in the project structure, but will neglect the non-selected alternatives. In the complex and dynamic project environment, however, all alternatives (both the selected and non-selected alternatives in the best found solution) could be important during project rescheduling since the alternatives in the project structure provide innovative ways to deal with project disruptions. Servranckx and Vanhoucke (2019b) have proposed to construct a set of back-up schedules, so-called alternative schedules, that can be mutually switched in case that unexpected disruptions occur during project execution. As a result, the authors do not only determine the best set of alternatives in the baseline schedule, but also dynamically adjust this set of alternatives.

In this research, we will contribute to the field of research by proposing a technique to reduce the complexity of the selection subproblem of the RCPSP-AS. More precisely, we will analyse the frequency of selection of the alternatives during the project scheduling process in order to determine whether we can fix certain alternatives in the project structure with only a limited (negative) impact on the scheduling objective as a result. By fixing alternatives in the project structure, we reduce the number of possible combinations between alternatives and thus reduce the complexity in the selection subproblem. Our contributions are threefold: (1) We present a technique to analyse the impact of alternatives on the solution quality of project instances. (2) We analyse the impact of two criteria on the reduction of the number of alternatives in the project structure. (3) We validate the proposed technique on both artificial project instances and empirical case studies.

2 Problem description

In the RCPSP-AS, Servranckx and Vanhoucke (2019a) define alternative ways to execute a subset of interrelated activities in the project. Such a subset of activities is referred to as a WP, called an *alternative subgraph*, and an alternative way to execute a WP is called an *alternative branch*. For the sake of simplicity, we will refer to the former as work packages and the latter as alternatives in the remainder of this abstract. The objective of the RCPSP-AS is to select for each WP exactly one alternative such that the resulting precedence, resource and logical feasible schedule has a minimal project makespan. In the RCPSP-AS, two types of dependencies between alternatives are modelled. *Linked alternative branches* indicate that (part of) the activities in one alternative branch should be selected when another alternative branch is selected. *Nested alternative subgraphs* imply that the selection of one alternative branch from an alternative subgraph is triggered by the selection of the enclosed alternative branch. Servranckx and Vanhoucke (2019a) define two parameters to model the dependencies between alternatives: the degree of linked alternative branches (%linked) and the degree of nested alternative subgraphs (%nested).

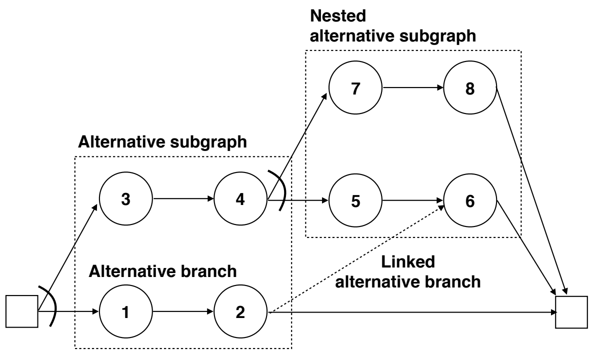


Fig. 1. Illustrative example of the RCPSP-AS

In figure 1, we show a simple example to illustrate the different concepts. We observe two alternative subgraphs with two alternative branches each, of which one alternative subgraph is nested. For the sake of simplicity, each alternative branch only consists of two activities in sequence. Each curved line ')' in figure 1 marks a choice between alternative branches in an alternative subgraph.

3 Solution approach

Due to the complexity of the RCPSP-AS, the search for a (near-)optimal solution is hard. Furthermore, a single best-found solution corresponds with a single set of selected alternatives, while significant information can be embedded in other alternative schedules. In this study, we therefore consider a set of (high-quality) solutions to analyse the impact of the selected certain alternatives on the solution quality. More precisely, we use the Tabu Search (TS) developed by Servranckx and Vanhoucke (2019a) to generate the set of high-quality solutions within a limited computational time.

In our research, we define two important criteria to analyse the set of generated solutions:

1. **SOLUTION QUALITY (Q)**: Since we should balance between a general approach (i.e. focus on all schedules generated in the TS) or a restrictive approach (i.e. focus on the best solutions generated in the TS), we will analyse a subset of schedules in the population. More precisely, we will select the best solutions with respect to the project makespan (i.e. scheduling objective) observed throughout the search process. A small subset will result in a high solution quality, while a larger subset will increasingly consist of schedules with a lower solution quality.
2. **CRITICALITY (C)**: In order to determine when an alternative can be fixed in the project structure, we measure for each schedule in our selected set of schedules, how many times each alternative is selected. The higher the frequency of occurrence, the more likely it is that this alternative must be chosen, and that it can therefore be fixed. When the frequency of an alternative exceeds a specified threshold, we fix it and do not consider the other alternatives. A higher (lower) threshold makes it harder (easier) to fix alternatives and thus corresponds with a higher (lower) criticality.

In our research, we will consider three thresholds (LOW, MED and $HIGH$) for both criteria. Using these two criteria, we can compare the frequency of selection of an alternative in the subset (Q) with the required frequency (C). This results in two possible outcomes. On the one hand, the frequency of one alternative for a WP in the subset Q is higher than the threshold C and thus this alternative can be fixed in the project structure. On the other hand, the frequency of all alternatives for a WP in the subset Q is lower than the threshold C. In this case, the selection of an alternative for this WP remains part of the selection subproblem. We prefer a higher number of fixed alternatives as this implies that a larger part of the selection subproblem can be considered solved. Therefore, the number of fixed alternatives is an important metric throughout the analysis of the artificial data instances and empirical case studies in the next section.

4 Computational results

In order to analyse the impact of the proposed technique on the complexity of the selection subproblem (i.e. the degree of alternatives that can be fixed in the project structure for different settings of Q and C), we will analyse a set of 3,600 artificial project instances presented in Servranckx and Vanhoucke (2019a). Furthermore, we will also validate these results based on three case studies. In general, we can conclude that the number of fixed alternatives indeed increases as the solution quality increases and the criticality decreases. As expected, this behaviour is observed in both the artificial dataset and the three case studies. However, we observe that the relative number of fixed alternatives is lower in the artificial projects compared to the case studies since the artificial dataset consists of projects with more or less similar alternatives. Furthermore, the experiments on the artificial dataset show us that the number of fixed alternatives increases in case that more dependencies exist between the alternatives in the project structure (i.e. %linked and %nested increase). Some preliminary results are summarised in table 1. In figure 2, the im-

		%nested					%linked				
		0	0.25	0.5	0.75	1	0	0.25	0.5	0.75	1
Solution quality	LOW	0.38	0.39	0.39	0.41	0.42	0.34	0.36	0.37	0.39	.42
	MED	0.40	0.40	0.42	0.43	0.44	0.35	0.38	0.40	0.43	0.46
	HIGH	0.41	0.41	0.45	0.46	0.48	0.38	0.41	0.44	0.47	0.51
Criticality	LOW	0.43	0.43	0.45	0.48	0.51	0.41	0.42	0.44	0.46	0.53
	MED	0.29	0.31	0.32	0.34	0.35	0.28	0.29	0.33	0.33	0.36
	HIGH	0.21	0.21	0.23	0.24	0.26	0.19	0.20	0.22	0.24	0.27

Table 1. Relative number of fixed alternatives for different settings of %nested and %linked

pact of both criteria on the number of fixed alternatives is illustrated for one of the three case studies. A darker color in the 3D-graph corresponds with a larger relative number of fixed alternatives. Similar results are obtained for the other case studies as well.

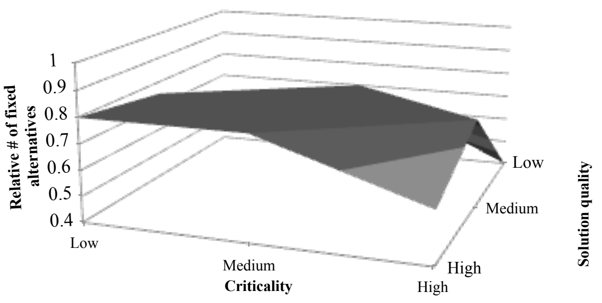


Fig. 2. Number of fixed alternatives for different settings of Q and C - Empirical data

In conclusion, we show that the elimination of certain alternatives allows us to identify key alternatives in the project structure. As a result, it can help project managers to focus their efforts during project rescheduling on important alternatives, while neglecting certain fixed alternatives.

References

Brucker, P., A. Drexl, R. Mohring, K. Neumann, E. Pesch, 1999, "Resource-constrained project scheduling: notation, classification, models, and methods.", *European Journal of Operational Research*, Vol. 122, pp.3-41

Kellenbrink C., S. Helber, 2015, "Scheduling resource-constrained projects with a flexible project structure", *European Journal of Operational Research*, Vol. 246 (2), pp. 379-391.

Servranckx, T., Vanhoucke, M., 2019a, "A tabu search procedure for the resource-constrained project scheduling problem with alternative subgraphs". *European Journal of Operational Research*, Vol. 273 (3), pp. 841-860.

Servranckx, T., Vanhoucke, M., 2019b, "Strategies for project scheduling with alternative subgraphs under uncertainty: similar and dissimilar sets of schedules". *European Journal of Operational Research*, Vol. 279 (1), pp. 38-53.

Tao, S., Dong, Z. S., 2017. "Scheduling resource-constrained project problem with alternative activity chains". *Computers & Industrial Engineering*, Vol. 114, pp. 288-296.

Tao, S., Dong, Z. S., 2018. "Multi-mode resource-constrained project scheduling problem with alternative project structures". *Computers & Industrial Engineering*, Vol. 125, pp. 333-347.

Reference Class Forecasting to improve time and cost forecasts: Empirical and statistical analysis

Tom Servranckx[1] and Mario Vanhoucke[1,2,3] and Tarik Aouam[1,4]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
tom.servranckx@ugent.be, mario.vanhoucke@ugent.be, tarik.aouam@ugent.be
[2] Technology and Operations Management Area, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, UK
[4] International University of Rabat, BearLab, Rabat Business School, Morocco

Keywords: Time and cost forecasting; Reference class forecasting; Empirical study.

1 Introduction

In highly complex and uncertain projects, project control techniques are used to monitor the progress and take corrective actions if necessary. However, these techniques require an ambitious, yet realistic, baseline schedule as a point-of-reference during project control. In practice, the construction of such a realistic baseline schedule in terms of realistic time and cost estimates is difficult since project risks are systematically underestimated, which might result in large time delays and cost overruns. As a result, the initial underestimation of risk, and the corresponding underestimation of time and budget, is problematic as it hinders effective and efficient project control actions. The possible explanations or root causes of the risk underestimation are twofold: deliberate and unintentional underestimation (Cantarelli *et al.* 1999, Flyvbjerg *et al.* 2002). In the former case, project managers are part of the problem, not the solution, and the installation of improved governance is advised (Flyvbjerg *et al.* 2004). In the latter case, it is recommended to use more advanced project management techniques. These techniques need to consider an outside view, rather than an inside view, to project risk estimation in order to overrule the project manager's perspective on the project risk. Using an inside view, the project manager will provide time and cost estimates based on past experience and gut-feeling, potentially overestimating the project team's capabilities and/or underestimating the project's risk. Using an outside view, the project estimations are (partially) based on historical information and objective metrics in order to cancel out systematic biases.

In this research, we investigate the ability of the reference class forecasting (RCF) method to improve the project forecasts for both the time and cost dimension. This method does not rely on specific estimates of the project manager, but instead compares the project to a statistical distribution of similar historical projects (Flyvbjerg 2006). Two important aspects of RCF are (1) the size of the reference classes and (2) the properties used to construct the reference classes (e.g. industry, project size, nationality). First, the reference classes cannot be too large as this might have a negative impact on the similarity between the projects in a reference class, but they cannot be too small either because the results might become statistically insignificant. Secondly, the selection of the correct properties of similarity is important to ensure that projects are compared with other, similar projects. Although these aspects of RCF are crucial to ensure a good outcome of the approach, they are often neglected in RCF studies in the existing literature. More precisely, the properties that are used to construct the reference classes are pre-defined and derived from other studies without critical assessment. Also, the choice of properties is often left to the project manager, which might again result in a subjective inside view, rather than an objective outside view. Therefore, we present a research study that aims to empirically investigate the important properties for RCF in a wide variety of industries as well as statistically analyse the ability of RCF to improve the project forecasts. A summary of the RCF method is shown in Figure 1. It is clearly indicated that RCF requires both a high similarity between the projects in a single reference class and a high dissimilarity between the different reference classes.

The contributions of our research study are threefold. First, project managers from Belgium and Italy were questioned about the ranking of different project properties for their ability to identify similar projects. In contrast to existing research studies, project managers were involved in the RCF method to give their opinion about properties of similarity, rather than that the properties of similarity were pre-defined or selected by the researchers themselves. Secondly, we investigate the individual and combined impact of the six best project properties on the forecasting accuracy of the RCF method using a real-life dataset of 52 projects collected from the interviewed project managers. In the literature, there already exists a lot of research on the improvement of project forecasts (for time and cost) using the earned value management method (Vandevoorde and Vanhoucke 2006, Vanhoucke and Vandevoorde 2007, Vanhoucke and Vandevoorde 2008), however, we extend these research efforts to the RCF methodology. Finally, we provide insights on the optimal number of project properties that should be used to create reference classes.

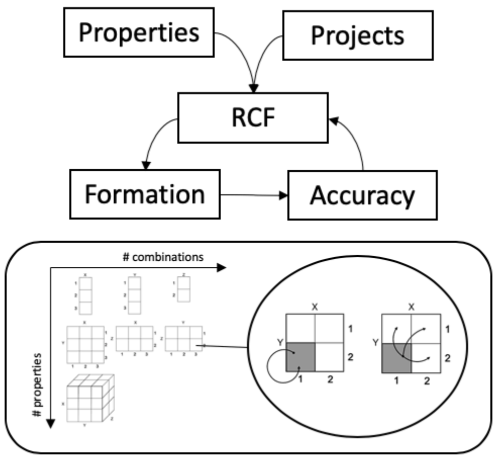


Fig. 1. General overview of RCF method

2 Methodology

The RCF method consists of four steps in order to obtain an accurate estimate of the duration and cost of the project prior to its start. First, a reference class of projects that is similar to the new project should be identified. In this study, we use a dataset of 52 real-life projects and we interview 76 project managers about good properties to measure the similarity between projects. Based on these interviews and data collection, we are able to construct different reference classes of historical projects. The interviews were conducted and the data was collected by Vandoorne *et al.* (2018). As shown in Figure 2 (Step 1), a new project will be identified based on the selected similarity properties and assigned to a specific reference class. Since the discrepancy between the initial estimated and final actual cost (i.e. the forecast error) is known for the historical projects in the dataset, a probability distribution of the forecast errors of the historic projects in this reference class is subsequently determined (Batselier and Vanhoucke 2016). We might observe that the forecasts for the new project are too optimistic (or pessimistic) based on the historical information captured in the reference class (see Figure 2 (Step 2)). Furthermore, this distribution of the reference class is transformed into a cumulative probability distribution (see Figure 2 (Step 3)). Finally, the original forecast for the new project (e.g. using a traditional inside view) should be changed (increased or decreased) based on the recommended uplift. In Figure 2 (Step 4), we show the inverse cumulative distribution to compare the willingness to accept risk for the new project and the required uplift of the estimated project duration and/or cost.

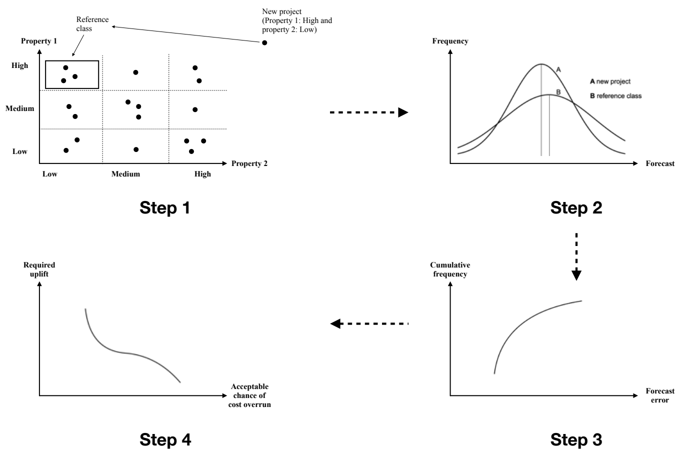


Fig. 2. Step-wise approach of RCF method

3 Computational results

In the computational experiments, we investigate the following features of RCF:

1. The individual and combined impact of the similarity properties on the forecasting accuracy of the reference classes.
2. The difference between time and cost forecasting.
3. The impact of computing the expected uplift with and without variance.
4. The impact of the number of similarity properties on the forecasting accuracy.

The preliminary results indicate that the RCF method results in more accurate forecasts for both the time and cost dimension. Table 1 shows that considering the variance in the probability distribution of the forecast errors in the reference class does not result in an improved performance of the RCF method. This might be due to the fact that the inclusion of variance in the computation of the uplift results in an overcorrection of the initial forecasts and, hence, too pessimistic (rather than optimistic) project duration and cost estimates, resulting again in forecasting inaccuracy. However, more extensive research is needed to investigate this specific observation. Finally, we notice that there exists an optimal number of properties to consider in the RCF method.

	Time		**Cost**	
	No Var	**Var**	**No Var**	**Var**
1	2.24	-9.15	1.13	-7.37
2	2.45	-7.75	1.20	-5.78
3	3.56	-7.46	2.65	-4.67
4	4.12	-5.97	3.52	-2.63
5	**6.89**	-4.21	**3.82**	-2.02
6	6.56	-3.60	3.15	-1.86
AVG	4.30	-6.36	2.58	-4.06

Table 1. Average percentage points improvement with/without variance for different numbers of properties and time/cost analysis

4 Conclusion

Based on a statistical analysis, we conclude that the RCF method results in robust improvements of the forecasting accuracy. While some combinations of properties result in a better performance of RCF than other combinations, a general observation is that a larger number of properties improves the RCF performance. In this case, the increased similarity between the projects outweighs the lower number of projects in each reference class. Finally, the results for the cost analysis (most common in the existing RCF literature) are validated for the time analysis.

References

Batselier, J., M. Vanhoucke, 2016, "Practical Application and Empirical Evaluation of Reference Class Forecasting for Project Management.", *Project Management Journal*, Vol. 47 (5), pp.36-51.

Cantarelli, C., B. Flyvbjerg, B. Wee, E. Molin, 2010, "Lock-in and its influence on the project performance of large-scale transportation infrastructure projects: Investigating the way in which lock-in can emerge and affect cost overruns.", *Environment and Planning B: Planning and Design*, Vol. 37, pp.792-807.

Flyvbjerg, B., M. Skamris, S. Buhl, 2002, "Underestimating costs in public works: Error or lie?", *Journal of the American Planning Association*, Vol. 68 (3), pp.279-295.

Flyvbjerg, B., C. Glenting, A. RÃ¸nnest, 2004a. "Procedures for Dealing with Optimism Bias in Transport Planning", *London: The British Department for Transport, Guidance Document.*

Flyvbjerg, B., 2006, "From Nobel Prize to project management: Getting risks rights.", *Project Management Journal*, Vol. 37, pp.5-15.

Vandevoorde, S., M., Vanhoucke, 2006, "A comparison of different project duration forecasting methods using earned value metrics.", *International Journal of Project Management*, Vol. 24, pp. 289-302.

Vandoorne, W., M. De Smyter, T. Servranckx, M. Vanhoucke. "Practical Application of Reference Class Forecasting: Identifying the Drivers of Similarity Between Projects". *Master thesis.*

Vanhoucke, M., S., Vandevoorde, 2007, "A simulation and evaluation of earned value metrics to forecast the project duration.", *Journal of the Operational Research Society*, Vol. 58, pp. 1361-1374.

Vanhoucke, M., S., Vandevoorde, 2008, "Earned value forecast accuracy and activity criticality.", *The Measurable News*, Summer, pp. 13-16.

Buffer Sizing in Critical Chain Project Management with Network Decomposition

Bingling She[1], Bo Chen[1] and Nicholas G. Hall[2]

[1] Warwick Business School, The University of Warwick
`b.she@warwick.ac.uk, b.chen@warwick.ac.uk`
[2] Fisher College of Business, The Ohio State University
`hall.33@osu.edu`

Keywords: critical chain project management, buffer sizing, network decomposition

1 Introduction

Approximately 30% of global economic activity is organized using project management, which implies an annual value of about \$27 trillion (Hu et al., 2015; Zhao et al., 2020). An important methodological development in project management is critical chain project management (Goldratt, 1997), or CCPM for short. The critical chain he defines generalizes the critical path (Kelley and Walker, 1959) used in traditional project planning, by incorporating the issue of resource availability. The critical chain is protected by three types of buffer: a project buffer, one or more feeding buffers, and one or more resource buffers. However, buffers that are too small result in replanning and expensive emergency procedures to avoid late delivery of the project. Whereas, buffers that are too large result in uncompetitive bidding for projects and loss of potentially valuable contracts. Hence, accurate buffer sizing is essential to the economic success of project companies.

Previous buffer sizing research, focused predominantly on the critical chain, typically results in excessive buffer sizing, and critical chains being challenged by feeding buffers during planning, as well as inconsistent performance in, e.g., makespan estimation.

We propose a new procedure for buffer sizing through analytical decomposition of the project network, which offers logical advantages over previous ones. The buffers are determined based on uncertainties of all associated chains and comparisons between parallel critical and noncritical parts. Our work also addresses the concerns in the literature that there is no systematic analysis of the project network structure and its relationship with buffers. In addition, we resolve the problem of a challenged critical chain, while simultaneously addressing issues with multiple critical chains. Computational testing on a case study of a real project and extensive simulated data shows that our procedure delivers much greater accuracy in estimating project makespan, and smaller feeding buffers, while the resulting critical chain is never challenged. Additional benefits include delayed expenditure, and reductions in work-in-process, rework, and multitasking.

2 Problem Description and Buffer Sizing Procedure

We consider a project consisting of a task set $V = \{1, 2, \ldots, n\}$, a set $E \subseteq V \times V$ of precedence relationships between tasks, and a renewable resource set $R = \{1, 2, \ldots, m\}$. Tasks 1 and n are dummy, with no duration or resource requirement, representing the start and end points of the project. The constant availability of resource k is r_k, $k \in R$, throughout the project horizon. We assume every non-dummy task i has a non-preemptive stochastic lognormal duration D_i with mean d_i, and a constant resource demand $r_{ik} \leq r_k$ of resource $k \in R$ in execution. In CCPM, we assume that task i consumes exactly one type of resource k_i (Leach, 2014, p.170), i.e., $r_{ik} = r_{ik_i}$ if $k = k_i$, $r_{ik} = 0$ if $k \neq k_i$. For each task i, let $\Gamma_i^{-1} \subseteq V \setminus \{i\}$ denote the set of its immediate predecessors. The set E of precedence relationships is given as: $E = \{(i,j) : i \in \Gamma_j^{-1}, j \in V\}$. For every pair $(i,j) \in E$, task i must be finished before the start of j. All the tasks and precedence relationships form an acyclic task-on-node network PN(V,E). In the network, precedence relationships (i,j) are denoted by $i \to j$ when referring to chains, and tasks are topologically numbered, meaning that if $(i,j) \in E$, then $i < j$. A *resource contention* is a situation where the total resource demand exceeds the resource availability during some time period of the project.

Figure 1 illustrates the main steps of our buffer sizing procedure.

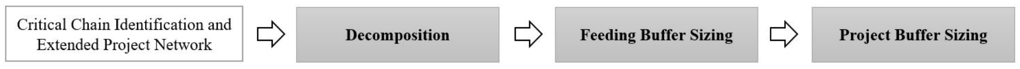


Figure 1. The Proposed Buffer Sizing Procedure

The preparation step in buffer sizing is critical chain identification, i.e, the identification of a baseline schedule without resource contentions, given mean task durations d_i. This problem is a special case of the classical resource constrained project scheduling problem (RCPSP, Demeulemeester and Herroelen (2002, p. 203)), where every task requires only one resource in R. This special case is strongly *NP*-hard, even when $m = 2, d_i = 1$ and $r_{ik_i} = r_{k_i} = 1$ (Bernstein et al., 1989). The literature describes several heuristics to solve the classical RCPSP, with the objective of minimizing the makespan. However, we propose a new heuristic for this special case of the RCPSP, which is designed to produce few additional precedence relationships from breaking resource contentions, and also short project makespans. The output is an extended precedence relationship set that defines the extended project network for our buffer sizing procedure. From the extended network, denoted by PN$(V, \widetilde{E})$, we identify the critical chain and other noncritical chains via the critical path method. Tasks on the critical chain are *critical tasks* and others are *noncritical tasks.*

Next, a project buffer and feeding buffers need to be located, respectively, at the end of the critical chain and wherever a feeding chain joins the critical chain, with well-defined size. Our buffer sizing procedure consists of three main steps.

(1) Decomposition of the network PN$(V, \widetilde{E})$ based on the identified critical chain.
(2) Feeding buffer sizing to account for nonciritical chain uncertanties and avoid the critical chain being exceeded by noncritical chains with the insertion of feeding buffers, using graph theory and linear programming techniques.
(3) Project buffer sizing to absorb uncertainties on the entire network with feeding buffers inserted, by aggregating safety margins of individual components derived from the first step.

For a single project with 150 tasks, the running time of our procedure with Matlab R2018a is less than 10 seconds on a personal computer with processor Intel(R) Core(TM) i5-7400 CPU @3.00GHZ and installed RAM of 8.00GB. Since the procedure is used at the planning stage of projects, this computation time is small enough for practical use.

3 Computational Analysis

We compare our buffer sizing procedure, PP, with five methods in the literature on four performance indicators via simulation. These five benchmarks are: Cut and Paste Method (C&PM, Goldratt (1997)), Root Square Error Method (RSEM, Newbold (1998)), Adaptive Procedure with Density (APD, Tukel et al. (2006)), Monte Carlo simulation method (SMC, Tenera (2008)), and the Method of Yu et al. (2013). The four performance indicators are: the accuracy of project makespan estimation - P_1, the reliability of the estimated project makespan - P_2, the average feeding buffer size - P_3, and the indicator of whether or not the identified critical chain is challenged by the insertion of feeding buffers - P_4.

In addtition to the data of a real project provided on the website of the Operations Research & Scheduling Research Group (OR&S, 2019b; Batselier and Vanhoucke, 2015; Vanhoucke et al., 2016), we use the data of 90 projects with 150 tasks each, randomly generated with RanGen2 software (OR&S, 2019a; Demeulemeester et al., 2003; Vanhoucke et al., 2008, 2016).

The numerical results on the four performance indicators show that our procedure provides more suitable buffer sizing and more accurate project makespan estimation, as well as much smaller feeding buffers while the critical chains are not challenged. Table 1 presents an exemplar set of our consistent results.

Performance Indicator	PP	C&PM	RSEM	APD	SMC	Yu2013
P_1	0.16	0.50	0.56	0.57	1.00	0.67
P_2	0.83	1.00	1.00	1.00	1.00	1.00
P_3	1.75	5.72	6.44	8.23	10.56	2.35
P_4	0	1	1	1	1	0

Table 1. Comparative Performance of Six Methods

4 Concluding Remarks

Because of the large economic value at stake, the sizing of buffers is centrally important to CCPM. Despite extensive research, previous approaches suffer from two significant deficiencies, which occur at the planning stage of projects: erroneous buffer sizing leads to inaccurate estimation of project makespan, and the insertion of feeding buffers overrides the critical chain. To resolve these issues, we have developed a buffer sizing procedure, which analyzes the entire project network by decomposing it, to obtain more accurate information about the relative lengths of critical and noncritical chains, and about interactions between buffers, over complex network structures. Hence, the size of a buffer is determined using global information about the project network instead of local information about the longest chain. Based on extensive computational testing for both real and simulated data, our procedure provides much more suitable buffer sizing and more accurate project makespan estimation, as well as much smaller feeding buffers, than five widely used benchmark methods. Additional benefits of our procedure include delayed expenditure, and reduced work-in-process, rework, and multitasking.

Our work should be of direct value to project management companies, for several reasons. First, all the information required for the network decomposition is immediately available in every well documented project. Second, the algorithmic steps required can easily be implemented as an add-on to commercial project planning software, such as Microsoft Project. Third, the elimination of the issue of challenging the critical chain simplifies project planning and reduces replanning. Fourth, the flexibility in using safety margins enables a project company to adjust its service level, in order to take into account strategic issues that frequently influence project choice and prioritization. Fifth, our buffer sizing procedure enables significantly more accurate and robust estimation of project makespan than earlier methods, thereby helping project companies to avoid the problems of underestimation and overestimation, and their significant costs or opportunity costs. Sixth, by enhancing CCPM, the choice that project companies face between using traditional and

CCPM planning may be clarified. Overall, we hope that the following comment will be helpful to project companies: our work enables significant reduction in buffers relative to other methods, but because the buffers we design are more accurately sized and located, the result is an improvement in project estimation.

References

Batselier, J. and Vanhoucke, M. (2015). Construction and evaluation framework for a real-life project database. *International Journal of Project Management*, 33(3):697–710.

Bernstein, D., Rodeh, M., and Gertner, I. (1989). On the complexity of scheduling problems for parallel/pipelined machines. *IEEE Transaction on Computers*, 38(9):1308–1313.

Demeulemeester, E. and Herroelen, W. (2002). *Project scheduling: a research handbook.* Kluwer Academic Publishers.

Demeulemeester, E., Vanhoucke, M., and Herroelen, W. (2003). RanGen: A random network generator for activity-on-the-node networks. *Journal of Scheduling*, 6:17–38.

Goldratt, E. M. (1997). *Critical Chain.* USA: The North River Press.

Hu, X., Cui, N., and Demeulemeester, E. (2015). Effective expediting to improve project due date and cost performance through buffer management. *International Journal of Production Research*, 53(5):1460–1471.

Kelley, J. E. and Walker, M. R. (1959). Critical-path planning and scheduling. In *Proceedings of the Eastern Joint Computer Conference*, pages 160–173. ACM.

Leach, L. P. (2014). *Critical Chain Project Management.* USA: The Artech House Publishers, 3rd edition.

Newbold, R. C. (1998). *Project Management in the Fast Lane – Applying the Theory of Constraints.* USA: CRC Press.

OR&S (2019a). Operations Research and Scheduling Research Group - RanGen. `http://www.projectmanagement.ugent.be/research/data/RanGen`. Last accessed on Feb 18, 2020.

OR&S (2019b). Operations Research and Scheduling Research Group - Real data. `http://www.projectmanagement.ugent.be/research/data/realdata`. Last accessed on Feb 18, 2020.

Tenera, A. B. (2008). Critical chain buffer sizing: A comparative study. In *2008 PMI research conference: Defining the future of project management*, pages 1–14, Warsaw, Poland. PMI.

Tukel, O. I., Rom, W. O., and Eksioglu, S. D. (2006). An investigation of buffer sizing techniques in critical chain scheduling. *European Journal of Operational Research*, 172:401–416.

Vanhoucke, M., Coelho, J., and Batselier, J. (2016). An overview of project data for integrated project managment and control. *The Journal of Modern Project Management*, 3(3):6–21.

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., and Tavares, L. (2008). An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research*, 187(2):511–524.

Yu, J., Xu, Z., and Hu, C. (2013). Buffer sizing approach in critical chain project management under multiresource constraints. In *Proceedings of 2013 6th International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII)*, volume 3, pages 71–75. IEEE.

Zhao, W., Hall, N. G., and Liu, Z. (2020). Project evaluation and selection with task failures. *Production and Operations Management*, 29(2):428–446. doi: 10.1111/poms.13107.

A new solution procedure for multi-skilled resources in resource-constrained project scheduling

Jakob Snauwaert[1], Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
`jakob.snauwaert@ugent.be` `mario.vanhoucke@ugent.be`
[2] Operations and Technology Management Centre, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, United Kingdom

Keywords: Project Management, Resource-Constrained Project Scheduling, Multi-skilled Resource Assignment, Metaheuristic.

1 Introduction

Creating a versatile and flexible workforce has become a paramount goal in the long term strategies of today's corporations. Investing in the development of a set of proficient human resources is a key concept in the success of companies that deal with projects (Riley, *et. al.* 2017). Moreover, the developing need for a team of complementary skillful resources is motivated by the current customer-specific wishes that have become regular for companies. This move from large production firms to job-shop companies with shorter product cycles creates a demand for workers who frequently learn to master new skills and adapt to new product requirements and specifications (Nembhard and Uzumeri 2000). Since a lot of projects are executed by workforces that consist of multi-skilled resources (Haas, *et. al.* 2001), this paper will study their impact on the project's characteristics and objectives.

This abstract addresses the *multi-skilled resource-constrained project scheduling problem*, further abbreviated as MSRCPSP. The MSRCPSP is an extension to resource-constrained project scheduling problem (RCPSP), which is NP-hard (Blazewicz, *et. al.* 1983). When dealing with multi-skilled resources in projects, two key decisions need to be taken. The first decision considers the workforce composition. Different from the RCPSP, the workers are not assumed to be uniform or to master the complete skillset, rather the multi-skilled resources are individually different and master a subset of the complete skillset. During the workforce composition stage, the goal is to create a workforce with the right size and available skills that match the skill requirements of the project at hand. The second decision to be taken involves simultaneously determining the activity schedule and the resource assignment. Using the assembled workforce, the aim is to construct resource-feasible solutions for the MSRCPSP that minimise two objectives. Additionally, we investigate the impact of skills on the final project schedule and resource assignment.

In our view of the problem, the multi-skilled resources are characterised by two key concepts known as breadth and depth. In this problem, the breadth of resources indicates the number of skills that a resource masters. Depth is linked to a skill that a resource masters, and signifies the efficiency at which the resource can perform the skill. The more proficient a resource is at a skill, the higher its depth is for that skill. In this study, the depth of a skill will impact the duration of the activities in the project. More specifically, the value of the depth can positively or negatively affect the variable duration of the activities dependent on the efficiency of the skills of the resources that are assigned to the activity.

In the literature, the MSRCPSP has been researched in several studies. Néron (2002) was the first to extend the RCPSP with resource skillsets, in this study two lower bounds were proposed. Furthermore, Bellenguez and Néron (2004) extended this research with hierarchical skill levels. This concept is similar to the depth of the resources, but is handled differently. In their research, the hierarchical skill levels are incorporated to specify a different required efficiency level for each skill requirement, which means that it only impacts the resource assignment and not the duration of the activities. Furthermore, various heuristics and other solutions were proposed for this problem (Bellenguez and Néron (2007), Li and Womer (2009), Correia, *et. al.* (2012)).

2 Problem Definition

In the MSRCPSP, a project can be represented by an acyclic activity-on-the-node network $G = (N, A)$, in which the minimum precedence relationships with a time-lag of zero are characterised by the arc set A and the activities by the node set N. The project consists of $|N|$ real activities and two additional dummy activities 0 and $|N| + 1$, which have no resource usage and a duration of zero. The activities are topologically ordered, that is, an activity always has a higher label than all of its predecessors. The project is said to be scheduled without pre-emption within the precedence relations and with a set of renewable multi-skilled resources R, with $R = 1, ..., |R|$, resulting in a baseline schedule with an activity starting time s_i and finishing time f_i for each activity i that minimises the makespan and resource idle time.

Each activity has a certain predefined duration and a demand for renewable resources, that are defined differently than for the RCPSP due to the presence of skills in the problem formulation. While the resource constraints for the RCPSP are defined as a set of renewable resource types (with index $k = 1, ..., |K|$), and each activity i is linked to a resource by its resource demand r_{ik} for each resource type k, the resource requirements are adjusted for skills in the MSRCPSP. Each activity $i \in N$ requires r_{ij} resources that master skill j of the set of available skills J for project G. More specifically, an activity i has a certain resource demand that is defined as a skill requirement for the skill j of a resource k. Note that a resource k can only be assigned to no more than one activity every time unit t. Every resource k from the set R masters at least one skill of the set of skills J, with $J = 1, ..., |J|$, defined by the breadth of the resource b_k. The depth d_{jk} of a skill j is the efficiency level of the resource k, and is defined as a rational number with a default standard value equal to 1. A resource k with a higher (lower) efficiency level is represented by a depth higher (lower) than 1.

An individual activity $i \in N$ has a predefined non-preemptive processing time of p_i, which is defined as the time required to execute the activity by workers $k \in R$ with skills of the default depth level, $d_{jk} = 1$. Higher depth values will reduce the processing time of an activity, while lower levels of depth will increase the processing time. To the best of the author's knowledge, there is no standardised equation in the literature that calculates the adjusted processing time of an activity i taking into account the depth d_{jk} of the skills ($j \in J$) of the assigned workers ($k \in R$). As proposed in Heimerl and Kolisch (2010) and Kolisch and Heimerl (2012), the reciprocal of the average depth will be used to calculate the impact of different efficiency levels on the activity duration. Moreover, we will investigate the impact of this variable activity duration on the activity scheduling and the resource assignment.

3 General Methodology

The MSRCPSP under study is an NP-hard problem (Bellenguez and Néron 2004), and is, therefore, solved using a genetic algorithm (GA). A schedule in the GA of this paper is represented by two lists which give a solution to the activity scheduling problem and the resource assignment problem (Figure 1). An activity list (AL) represents the activity schedule. In such an activity list of length $|N|$, activities with a higher priority to be scheduled in the schedule generation scheme are earlier in the list than activities with a lower priority. A new priority rule list (PL) is proposed that represents the resource assignment problem of the MSRCPSP. In this list of length $|N|$, every activity is assigned a priority list value PL_i that corresponds to a resource priority rule, which consists of |R| values. A priority rule can be appointed to multiple activities and the value PL_i determines the priority rule of activity i. A set of resource priority rules is carefully assembled to support the assignment of resources to the activities.

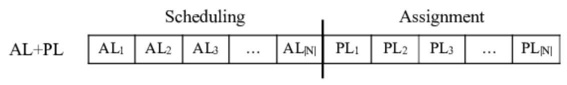


Fig. 1. Solution representation

To initialise the genetic algorithm several different initial solutions are created based on the skill requirements, the skill availabilities and a random approach. Afterwards, two parent solutions are selected from this initial population. These two parents will undergo a crossover to generate a new child solution. In this algorithm, two different crossovers are presented. The random two-point crossover randomly selects two points in the activity list or the priority rule list and creates a new child solution by copying the values outside the two points from the first parent, and inserting the values between the two points using the order of the second parent. The split low demand ranges crossover is specifically designed for the MSRCPSP and utilises ranges, or sections, of activities in the activity list that have a lower resource requirement than the average resource requirement. The middle points of the two longest ranges are then applied as in a two-point crossover to split the sections of activities with low resource requirements. Furthermore, two basic mutators are incorporated in this algorithm, which are the swap- and modify-mutator.

The generated solutions are further improved through several local searches. A local search is created that improves the schedule based on the forward-backward iterative scheduling technique of Li and Willis (1992). To improve the resource assignments of the generated schedules, two local searches are proposed that, respectively, focus on the availability of the skills in the workforce and the depth of the resources.

4 Computational experiments

The experimental results will consists of 3 different insightful analyses.

- A comparison of the solution quality of the algorithm against a benchmark to investigate the competitiveness of our algorithm.
- Investigating the quality of the set of resource priority rules used for the resource assignment problem.
- Managerial insights on the ideal workforce size and the availability of the skills. Preliminary results have shown that, in most cases, both the ideal workforce and the skill-pool size depend on the objective and the seriality of the project network (Table 1).

Table 1. Preliminary managerial insights

Objective	SP	Skill-pool size	Workforce size
Makespan	Low	Mid	Mid-High
Makespan	Mid	Low-Mid	No preference
Makespan	High	Low-Mid	No preference
Idle Time	Low	Mid	Low
Idle Time	Mid	Low	Low
Idle Time	High	Low	Low

Acknowledgements

The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, FWO and the Flemish Government - department EWI.

References

Bellenguez, O., Néron, E., 2004. “Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills", *International Conference on the Practice and Theory of Automated Timetabling*, Springer, pp. 229-243.

Bellenguez-Morineau, O., Néron, E., 2007. “A branch-and-bound method for solving multi-skill project scheduling problem", *RAIRO-Operations Research*, 41 (2), pp. 155-170.

Blazewicz, J., Lenstra, J. K. and Kan, A. R., 1983. “Scheduling subject to resource constraints: classification and complexity", *Discrete applied mathematics*, 5 (1), pp. 11-24.

Correia, I., Lampreia-Lourenc Ì§o, L., Saldanha-da Gama, F., 2012. “Project scheduling with flexible resources: formulation and inequalities", *OR spectrum*, 34 (3), pp. 635-663.

Gutjahr W.J., Katzensteiner, S., Reiter, P., Stummer, C., and Denk, M., 2010, “Multi-objective decision analysis for competence-oriented project portfolio selection", *European Journal of Operational Research*, 205 (3), pp. 670-679.

Haas, C. T., Rodriguez, A. M., Glover, R. and Goodrum, P. M., 2001. “Implementing a multiskilled workforce", *Construction Management & Economics*, 19 (6), pp. 633-641.

Heimerl, C., Kolisch, R., 2010. “Scheduling and staffing multiple projects with a multi-skilled workforce.", *OR spectrum*, 32 (2), pp. 343-368.

Kolisch, R., Heimerl, C., 2012. “An efficient metaheuristic for integrated scheduling and staffing it projects based on a generalized minimum cost flow network.", *Naval Research Logistics*, 59 (2), pp. 111-127.

Li, K. Y., Willis, R. J., 1992. “An iterative scheduling technique for resource-constrained project scheduling", *European journal of operational research*, 56 (3), pp. 370-379.

Li, H., Womer, K., 2009. “Scheduling projects with multi-skilled personnel by a hybrid MILP/CP benders decomposition algorithm", *Journal of Scheduling*, 12 (3), pp. 281.

Nembhard, D. A., and Uzumeri, M. V., 2000. “Experiential learning and forgetting for manual and cognitive tasks", *International journal of industrial ergonomics*, 25 (4), pp. 315-326.

Néron, E., 2002. “Lower bounds for the multi-skill project scheduling problem", *Proceeding of the Eighth International Workshop on Project Management and Scheduling*, pp. 274-277.

Riley, S. M., Michael, S. C. and Mahoney, J. T., 2017. “Human capital matters: Market valuation of firm investments in training and the role of complementary assets", *Strategic Management Journal*, 38 (9), pp. 1895-1914.

The impact of limited budget on the corrective action taking process

Jie Song[1], Annelies Martens[1] and Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
jieson.song@ugent.be, annelies.martens@ugent.be, mario.vanhoucke@ugent.be
[2] Technology and Operations Management, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, United Kingdom

Keywords: Project Management, Project Control, Simulation, Earned Value Management.

1 Introduction

The main goal of project control is to identify the deviations between the baseline schedule and the actual progress of the project by measuring the project performance in progress, and using the project control methodologies to generate warning signals that act as triggers for corrective actions to bring the project back on track. To that purpose, tolerance limits are set on the required project performance, such that if the warning signals exceed these limits, they should result in appropriate corrective actions. However, the corrective actions always directly result in additional costs (reassignment of highly-skilled personnel, extra equipment, improvement of manpower, etc.) during project progress. The budget used for taking corrective actions is always limited in practice. In order to make the best use of the limited budget, they should be allocated in an efficient and effective way to repair the project delays. In this paper, four different approaches are proposed to allocate the limited budget to take corrective actions with the aim of improving the expected project outcome.

2 Problem formulation

The research on controlling projects with Earned Value Management (EVM) has grown in the last decades, and the topic has been investigated from different angles. The basic and more detailed aspects about EVM can be found in several studies (Anbari 2003, Fleming and Koppelman 2010). Although these performance measures in the EVM methodology detect deviations from the project plan, they do not notify the project manager whether these deviations are acceptable or not. Therefore, tolerance limits should be set up in conjunction with the EVM metrics in order to support the project manager to carry out corrective actions when necessary. Tolerance limits prescribe whether the measured project performance is acceptable or not, such that the project manager can decide whether corrective action is necessary. These tolerance limits can be classified into three types in literature according to their complexity (by the type and amount of data required together with the statistical tool used to analyze these data) and their abilities in predicting the need for actions, namely the *static tolerance limits*, *statistical tolerance limits* and *analytical tolerance limits*. For a detailed comparison and evaluation for these three types of tolerance limits for project control, the reader is referred to Vanhoucke (2019). When the tolerance limits are exceeded, the project manager should decide whether to take corrective actions or not. Vanhoucke (2010) presents a simulation study with corrective actions on the selected highly sensitive activities by reducing the activities' delay to half of their baseline duration. However, reducing an activity's duration normally leads to increased cost and this is known as *activity crashing.* The computational results have shown that corrective actions taken on highly sensitive activities are more reliable for parallel projects, but lead to poor contribution for serial projects. In order to provide a better alternative to the poor behavior of these corrective actions for more serial projects, Vanhoucke (2011) extends this research by introducing the concept of control efficiency for corrective actions and compares two alternative methods in a simulation study. The author still relies on activity crashing as the only way of taking corrective actions. Moreover, the author shows that EVM is more reliable for serial projects than for parallel projects. Hu *et. al.* (2016) implement corrective actions on selected sensitive activities by reducing the baseline activity duration proportional to activity sensitivity information to revise the project delay.

Despite the growing amount of literature on project control with corrective actions, to the best of our knowledge, none of these studies discussed previously have explicitly taken the limited availability of budget for taking corrective actions into account. Due to the presence of a budget constraint, not all the corrective actions are able to be taken timely when the current project performance is not acceptable, and this might have a negative impact on the project outcome. Consequently, the central question of this paper is how the limited budget can be best allocated, such that it can support the project manager to take efficient corrective actions to improve the expected project outcome.

This paper presents four different approaches to allocate the limited budget according to different project characteristics. More precisely, the *Earned Value* (EV) approach makes use of the earned value methodology to allocate the limited budget according to the cost information of each project phase. The *Earned Schedule* (ES) approach allocates the limited budget using the earned schedule methodology which measures the time and cost information of the project. The *Earned Duration* (ED) approach uses the earned duration methodology to allocate the limited budget based on the time information of the project. The *Activity Risk* approach (AR) uses the risk information of each individual activity to allocate the limited budget.

3 Methodology

3.1 Data generation

In order to test the impact of limited budget on the corrective action taking process, a set of 900 fictitious project networks with topological structure are generated by a project network generator RanGen (Demeulemeester *et. al.* 2003, Vanhoucke *et. al.* 2008). The dataset is extensively applied in previous project control simulation studies (Ballesteros-Pérez *et. al.* 2019, Elshaer 2013). The topological structure of these fictitious project networks are presented by the serial/parallel (SP) indicator, which is used to measure the closeness of a project network, and contains 100 projects for SP = 0.1, 0.2,...,0.9. More specifically, project networks with low (high) SP values are close to parallel (serial) projects. Each project network consists of 30 activities. For each activity, the fixed cost is uniformly sampled between €10 and €90, a variable cost is uniformly sampled between €100 and €900.

3.2 Setting tolerance limits

The tolerance limits are constructed by assuming that the project buffer of a certain percentage of the planned duration is consumed proportionally to the PV accrue of each project phase. More specifically, when the project is completed at x% of the BAC, x% of the project buffer is allowed to be consumed for that phase. This approach is developed by Martens and Vanhoucke (2017). In this study, the tolerance limits are set for project buffer sizes of 15%, 25% and 35% of the PD to simulate the frequency of warning signals in the project.

3.3 Simulated project execution

With the aim of generating a large set of fictitious project executions, Monte Carlo simulations are employed to generate project information with the presence of uncertainty. We will use the lognormal distribution which is skewed to the right to model the real activity duration (Hu *et. al.* 2016, Bie *et. al.* 2012, Kotiah and Wallace 1973), with $\mu = 1.1$ and $\delta = 0.3$.

3.4 Corrective actions

When a warning signal is generated during project execution, an action will be taken at that control phase. In this case, the estimated remaining duration of the eligible activity will be reduced under strict predefined limits. In the simulation experiments, three reduced levels of duration are considered and compared (i.e. 30%, 50% and 70% of the estimated activity duration) in order to simulate different degrees of corrective actions.

4 Results

In the experiment, the time efficiency is used to assess the ability of each approach to reduce the real project duration within the same control effort, which has been introduced in Vanhoucke (2010). The time efficiency is measured as a ratio between the total reduction in the project real duration after taking corrective actions ($RD^{no} - RD^{yes}$) and the sum of reduction in the activities due to crashing ($RD_i^{no} - RD_i^{yes}$), as described in Eq. (1).

$$time\,efficiency = \begin{cases} 0,\, if\, denominator = 0 \\ \frac{1}{nrs_w} \sum_{k=1}^{nrs_w} (\frac{RD^{no} - RD^{yes}}{\sum_{i=1}^{nract} (RD_i^{no} - RD_i^{yes})}),\, otherwise \end{cases} \tag{1}$$

With nrs_w the total number of projects with warning signals. As can be seen from this formula, the time efficiency equals to 1 when the reduction in real project duration is equal to the total change in all evaluated activities, which is a desirable state. Moreover, the time efficiency can be also equal to 0, when the total change in all the activities has no effect on the real project duration (numerator = 0) or when no corrective actions are taken when warning signals generated, due to a lack of available control budget (denominator = 0).

First, the computational results show that the ED approach (39.12%, Avg.) outperforms on average the other three approaches (EV: 29.13%, ES: 37.36%, AR: 38.11%). This ED approach allocates the limited budget according the to earned duration metric, and generates warning signals using tolerance limits set of the duration performance index DPI. It has already been shown in Batselier and Vanhoucke (2015) that this ED-DPI approach performs well for predicting the total project duration, but now it shows that it also performs well when taking corrective actions under a limited budget. Hence, our study confirms previous results, and shows that using earned duration management performs well both for project duration forecasting and for taking corrective action to bring projects back on track.

Second, the results show that the time efficiency decreases when the buffer size grows. When a relatively small buffer size is added at the end of the project, the project performance in progress will be frequently exceeded by the dynamic tolerance limits, and even the presence of small delays in the projects will result in more corrective actions in dynamic project progress. However, in case of a relatively large buffer size, the project performance measures seldom drop below the tolerance limits to generate warning signals to take corrective actions. In these cases, a false signal (i.e. project delay due to serious delay in non-critical activities which has little impact on the project duration) will have a larger influence (negative) on the time efficiency.

References

Anbari F., 2003, “Earned value project management method and extensions", *Project Management Journal*, Vol. 34, pp. 12-23.

Ballesteros-Pérez P., A. Cerezo-Narvaéz, M. Pastor-Fernández, and M. Vanhoucke, 2019, “Performance comparison of activity sensitivity metrics in schedule risk analysis", *Automation in Construction*, Vol. 106, 102906.

Batselier J., M. Vanhoucke, 2015, “Evaluation of deterministic state-of-the-art forecasting approaches for project duration based on earned value management", *International Journal of Project Management*, Vol. 33(7), pp. 1558-1596.

Bie L., N. Cui and X. Zhang, 2012, “Buffer sizing approach with dependence assumption between activities in critical chain scheduling", *International Journal of Production Research*, Vol. 50, pp. 7343-7356.

Demeulemeester E., M. Vanhoucke and W. Herroelen, 2003, “Rangen: A random network generator for activity-on-the-node networks", *Journal of Scheduling*, Vol. 6, pp. 17-38.

Elshaer R., 2013, “Impact of sensitivity information on the prediction of project's duration using earned schedule method", *International Journal of Project Management*, Vol. 31, pp. 579-588.

Fleming Q., J. Koppelman, 2010, “Earned value project management", Project Management Institute, Newton Square, Pennsylvania, 3rd edition.

Hu X., N. Cui, E. Demeulemeester, and L. Bie, 2016, “Incorporation of activity sensitivity measures into buffer management to manage project schedule risk", *European Journal of Operational Research*, Vol. 249, pp. 717-727.

Kotiah T., N. D. Wallace, 1973, “Another look at the pert assumptions", *Management Science*, Vol. 20(1), pp. 44-49.

Martens A., M. Vanhoucke, 2017, “A buffer control method for top-down project control", *European Journal Of Operational Research*, Vol. 262, pp. 274-286.

Vanhoucke M., J. Coelho, D. Debels, B. Maenhout and L. Tavares, 2008, “An evaluation of the adequacy of project network generators with systematically sampled networks", *European Journal of Operational Research*, Vol. 187, pp. 511-524.

Vanhoucke M., 2010, “Using activity sensitivity and network topology information to monitor project time performance", *Omega The International Journal of Management Science*, Vol. 01, pp. 1-5.

Vanhoucke M., 2011, “On the dynamic use of project performance and schedule risk information during project tracking", *Omega The International Journal of Management Science*, Vol. 39, pp. 416-426.

Vanhoucke M., 2019, “Tolerance limits for project control: An overview of different approaches", *Computers & Industrial Engineering*, Vol. 127, pp. 467-479.

A New Lower Bound Approach for the Multi-mode Resource Constrained Project Scheduling Problem

Christian Stürck[1]

Helmut Schmidt University Hamburg, Germany
christian.stuerck@hsu-hh.de

Keywords: MRCPSP, project scheduling, lower bounds.

1 Introduction and problem description

The multi-mode resource constrained project scheduling problem (MRCPSP) is an extension of the resource-constrained project scheduling problem (RCPSP). Besides the decision of starting time, a mode has to be chosen for each activity.

The objective of the MRCPSP is to find the minimum feasible makespan of the project. The project consists of a set of activities $A = \{0, ..., n+1\}$. Each activity i can be executed in different modes. Therefore, for each activity i a set of modes M_i is given. Mode m has a duration $d_{i,m} \in \mathbb{Z}^+$ as well as a resource consumption $r_{i,m,k} \in \mathbb{Z}^+$ for each resource $k \in \mathcal{R} \cup \mathcal{R}^n$. The duration and the resource consumption of an activity varies with respect to the chosen mode. On the one hand, a set of renewable resources $\mathcal{R}$ is given which are available per time unit. On the other hand, a set of non-renewable resources $\mathcal{R}^n$ exists which are available through out the whole project. Activities have precedence constraints between each other. Set $E = \{(i,j) : i,j \in A\}$ indicates the precedence relations. Every activity has to be set to a mode and a starting time, while respecting all precedence and resource constraints. A mathematical model for the MRCPSP was first described by Talbot (1982):

$$\min \sum_{ES_{n+1}}^{LS_{n+1}} x_{n+1,1,t} \cdot t \tag{1}$$

$$\text{s.t.} \sum_{m \in M_i} \sum_{t=ES_i}^{LS_i} x_{i,m,t} = 1 \qquad \forall i \in A \tag{2}$$

$$\sum_{m \in M_i} \sum_{t=ES_i}^{LS_i} x_{i,m,t} \cdot (t + d_{i,m}) \leq \sum_{m \in M_j} \sum_{t=ES_j}^{LS_j} x_{j,m,t} \cdot t \qquad \forall (i,j) \in E \tag{3}$$

$$\sum_{i \in A} \sum_{m \in M_i} \sum_{t=ES_i}^{LS_i} x_{i,m,t} \cdot r_{i,m,k} \leq R_k \qquad \forall k \in \mathcal{R}^n \tag{4}$$

$$\sum_{i \in A} \sum_{m \in M_i} \sum_{q=\max(ES_i, t-d_{i,m}+1)}^{\min(t, LS_i)} x_{i,m,q} \cdot r_{i,m,k} \leq R_k \qquad \forall k \in \mathcal{R}, \forall t \in T \tag{5}$$

$$x_{i,m,t} \in \{0,1\} \qquad \forall i \in A, \forall m \in M_i, t = ES_i, \ldots, LS_i \tag{6}$$

The binary decision variables $x_{i,m,t}$ are set to one if and only if the activity i is executed in mode m and if it starts at time period t. We consider the non-preemptive case of the MRCPSP, which means that an activity cannot be interrupted or change its mode, once it is started. Note that in the model above the index t denotes the starting period and not the completion period as introduced by Talbot.

The objective function (1) minimizes the starting time of the dummy activity $n+1$. As this denotes the end of the project, the makespan of the project is minimized. Term (2) ensures that each activity is assigned to exactly one mode and to one starting time. Constraints (3) guarantee the precedence relations, i.e., an activity i must be finished before activity j can start if $(i,j) \in E$. Inequalities (4) and (5) restrict the resource consumptions for the non-renewable and renewable resources, respectively. The binary decision variables are defined in (6).

The MRCPSP is a $\mathcal{NP}$-hard problem, even finding a feasible mode assignment is $\mathcal{NP}$-complete if the instance has more than one non-renewable resource (Kolisch and Drexl (1997)). Lower bounds can be used either to evaluate the quality of a solution or as bounds in an exact approach to reduce the solution space. Therefore this work presents new lower bound approaches for the MRCPSP.

2 A new lower bound approach

Several lower bound procedures for the MRCPSP already exist in the literature. The most common one is the Critical Path Lower Bound (CP-LB) which only considers the precedence constraints and relaxes all resource constraints (Kelley (1963)). Since every activity has more than one mode the mode with the shortest duration is always chosen for the CP-LB.

More complex lower bounds were presented as well. Maniezzo and Mingozzi (1999) presents several LP relaxations. Pesch (1999) uses an adaptation of the Talbot (1982) algorithm for generating lower bounds. The approaches of Zhu *et al.* (1997) and Stürck and Gerhards (2018) are based on calculating new earliest starting times for the activities which lead to new lower bounds. But there are a lot more lower bound procedures for the RCPSP. For an example the work of Klein and Scholl (1999) alone presents 17 different approaches for lower bounds for the RCPSP.

This work will present a new lower bound approach for the MRCPS. It is based on the *Capacity Bound* for the RCPSP in Klein and Scholl (1999). Klein and Scholl (1999) described the *Capacity Bound* as follows: for each activity the duration is multiplied with the renewable resource consumption of the activity. These products are summed up and divided by the available resource per time period:

$$\textit{Capacity Bound} = \max\{\lceil \sum_{i \in A} r^r_{i,k} \cdot d_i / R_k \rceil : k \in \mathcal{R}^r\} \ . \tag{7}$$

This operation is done for every renewable resource. The maximal rounded up quotient determines the lower bound for the RCPSP.

To use this approach for the MRCPSP it has to be adapted. We combine the *Capacity Bound* with a feasible mode assignment. Therefore we call this bound the *Feasible Mode Capacity Bound*. The used feasible mode assignment is inspired by the MIP approach of Toffolo *et al.* (2016). The problem of the mode selection is redefined to a multidimensional knapsack problem:

$$\min \sum_{i \in A} \sum_{m \in M_i} y_{i,m} \cdot d_{i,m} \tag{8}$$

$$\text{s.t.} \sum_{m \in M_i} y_{i,m} = 1 \qquad \forall i \in A \tag{9}$$

$$\sum_{i \in A} \sum_{m \in M_i} y_{i,m} \cdot r^n_{i,m,k} \leq R^n_k \qquad \forall k \in \mathcal{R}^n \tag{10}$$

$$y_{i,m} \in \{0,1\} \qquad \forall i \in A, \forall m \in M_i \tag{11}$$

The binary decision variables $y_{i,m}$ are set to one if and only if the activity i is executed in mode m. The constraint (9) assigns exactly one mode to each activity. Term (10) ensures that the non-renewable resources are not exceeded. The objective function (8) is just subsidiary for finding a feasible mode assignment.

These two approaches can be combined a new lower bound procedure: the *Feasible Mode Capacity Bound*:

$$\min B_l = \sum_{i \in A} \sum_{m \in M_i} y_{i,m} \cdot d_{i,m} \cdot r^r_{i,m,k} \tag{12}$$

$$\text{s.t.} \sum_{m \in M_i} y_{i,m} = 1 \qquad \forall i \in A \tag{13}$$

$$\sum_{i \in A} \sum_{m \in M_i} y_{i,m} \cdot r^n_{i,m,k} \leq R^n_k \qquad \forall k \in \mathcal{R}^n \tag{14}$$

$$y_{i,m} \in \{0,1\} \qquad \forall i \in A, \forall m \in M_i \tag{15}$$

The aim is to find a feasible mode assignment with the minimal renewable resource usage B_l for each renewable resource $l \in \mathcal{R}^r$. The binary decision variables (15) are set to one if and only if the activity i is executed in mode m. The objective function (12) is similar to Term (7) with the addition of $y_{i,m}$ which ensures that only the chosen modes are considered. With (13) each activity is assigned to exactly one mode. Constraint (14) considers all non-renewable resources $k \in \mathcal{R}^n$.

This mathematical model ((12) – (15)) is solved for each renewable resource $l \in \mathcal{R}^r$. In the next step the *Feasible Mode Capacity Bound* can be solved:

$$\textit{Feasible Mode Capacity Bound} = \max\{\lceil B_l/R_l \rceil : l \in \mathcal{R}^r\} \ . \tag{16}$$

Each quotient is rounded up since only integer periods are considered. The maximal quotient determines the *Feasible Mode Capacity Bound.* The next section will display the computational experiments.

3 Computational experiments

The experiments were done on the MMLIB instances presented by Van Peteghem and Vanhoucke (2014) and carried out on a PC with an Intel Xeon X5650 CPU at 2.66 GHz. The algorithm is implemented in C# and CPLEX 12.6.3 was used as solver.

Although finding a feasible mode assignment is already $\mathcal{NP}$-complete if the instance has more than one non-renewable resource (Kolisch and Drexl (1997)) the procedure is quite fast. The computation of the lower bounds took 24.03 seconds on average, with a minimum of 0.01 seconds for 204 instances and a maximum of 109.91 seconds for an MMLIB+ instance with 100 activities and 9 modes. Table 1 shows the results for the computational experiments.

Table 1. Computational experiments for the *Feasible Mode Capacity Bound* on the MMLIB

	MMLIB50	MMLIB100	MMLIB+	Sum
total number of instances	540	540	3 240	4 320
Best known solution = CP-LB	229	264	473	966
Feasible Mode Capacity Bound > CP-LB	107	120	1 878	2 105

For 966 instances the best known solution is already equal to the *Critical Path Bound* and therefore optimal. The presented procedure was able to find a better *Feasible Mode Capacity Bound* compared to *Critical Path Bound* for 2 105 of the remaining 3 354 instances.

4 Conclusion

This work presented a new approach for computing lower bounds for the MRCPSP. The computational experiments showed that the procedure is able to improve the lower bound for 67.76 % of the MMLIB instances without a known optimum. Furthermore, the experiments showed that the computation time of the approach is reasonably low with a few seconds for most of the MMLIB instances.

References

Kelley, J. E. 1963, “The critical-path method: Resources planning and scheduling", *Industrial Scheduling*, Vol. 13, no. 1, pp. 347-365.

Klein, R. and Scholl, A. 1999, “Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling", *European Journal of Operational Research*, Vol. 112, no. 2, pp. 322-346.

Kolisch, R. and Drexl, A. 1997, “Local search for nonpreemptive multi-mode resource-constrained project scheduling", *IIE Transactions*, Vol. 29, no. 11, pp. 987-999.

Maniezzo, V. and Mingozzi, A. 1999, “A Heuristic Procedure For the Multi-mode Project Scheduling Problem Based on Benders Decomposition", In: *Project Scheduling: Recent Models, Algorithms and Applications* eds: Węglarzz, J., pp. 179-196, Kluwer(Boston).

Pesch, E. 1999, “Lower bounds in different problem classes of project schedules with resource constraints", In: *Project Scheduling: Recent Models, Algorithms and Applications* eds: Węglarzz, J., pp. 179-196, Kluwer(Boston).

Stürck, C. and Gerhards, P. 2018, “Providing Lower Bounds for the Multi-Mode Resource-Constrained Project Scheduling Problem", In: *Operations Research Proceedings 2016* eds: Fink, A., Fügenschuh, A. and Geiger, M. J., pp. 551-557, Springer (Cham).

Talbot, F. B. 1982, “Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case", *Management Science*, Vol. 28, no. 10, pp. 1197-1210.

Toffolo, T., Santos, H. G., Carvalho, M., and Soares, J. A., “An integer programming approach to the multimode resource-constrained multiproject scheduling problem", *Journal of Scheduling*, Vol. 19, no. 3, pp. 295-307.

Van Peteghem, V. and Vanhoucke, M. 2014, “An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances", *European Journal of Operational Research*, Vol. 235, no. 1, pp. 62-72.

Zhu, G., Bard, J. F. and Yu, G. 2006, “A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem", *INFORMS Journal on Computing*, Vol. 18, no. 3, pp. 377-390.

How to find Critical Mass of Task Threatening the Projects

Šubrt Tomáš[1], Brožová Helena[2]

Czech University of Life Sciences Prague, Fac. of Economics and Management,
Dept. of Systems Engineering, Kamýcká 129, 165 21 Praha 6 – Suchdol, CZ
[1]subrt@pef.czu.cz, [2]brozova@pef.czu.cz

Keywords: Project management, Critical Mass, task neglectedness, routine task

1 Introduction

Modern project management uses a variety of methods that identify tasks and tasks resources potentially threatening the successful completion of the project. From the 1950s to the 1980s, almost exclusively the critical path methods, such as CPM, MPM, PERT (Nicolas, Stein, 2012), and GERT (Pritsker, 1966), were used to identify tasks that delay a project's completion date due to delay of tasks. Based on the results that these methods provide the project managers can draw their attention to critical tasks that need to be monitored in terms of conditions for their implementation, resource allocation and sufficient budgeting. They can also define the potential threats, but the tasks highly uncritical with large slacks are omitted. Newer methods from the last quarter of the 20th century are more focused on the human factor in project tasks completing. Multi-resource tasks, or sequences of tasks with a single assigned resource, are seen as threatening tasks (InterPlan System, 2002). For instance, the Critical Chain method and its derivatives seek to implement project management practices that eliminate human factor influence and, with the use of post-limitation postulates (Goldratt, 1997) warn the project manager of potential project threats or failures. Other methods assessing the importance and criticality of tasks are based on heuristic or the criticalness potential of tasks (Brožová et al., 2016, 2019) measuring task characteristics in terms of cost, resources, duration, connectivity, distance from origin. The higher value of task criticalness potential the higher needed focus on the task.

The first thoughts that small and insignificant tasks that do not lie even not on the critical path, but are present in larger quantities, may also endanger the project due date or successful completion arose at the beginning of 21st century, when the Turnaround "project" Management theory emerges in the context of hurricane Katrina in 2005. The term Critical Mass is used to denote such tasks. In Turnaround context, these tasks concentrated into the critical mass can endanger the project but such tasks are not typical only for turnaround but also for general projects. The critical mass tasks typically do not exceed ten percent of the estimated man-hours, are small, short, inexpensive, low priority, practically without relationships to other tasks, with only few resource assignments. Even though the critical path tasks are on schedule, critical mass tasks generally have plenty of float (slack time) and can be executed just about any time in any order. Nevertheless, insufficient resources often force these tasks to be scheduled late (InterPlan Systems, 2002). The omission or neglect of one of these tasks may not jeopardize the whole project, but grouping them in the critical mass will endanger the project in its very essence.

Typical examples of such tasks are various control or verification tasks, but also performed routinely tasks, many times per project, which are normally performed only a few times instead of the scheduled number of repetitions. Therefore, we try to derive indicators that can reveal these tasks using the indicators values or similarities of these values.

2 Critical Mass Potential and Routine Tasks

The Critical Mass Member Candidate (CMMC) tasks may endanger the project itself or its crucial parts. As mentioned above, tasks of this type are often short and inexpensive, isolated, can take place at any time, and have no or few successors. The starting points of suggested method of identification of the critical mass task are the criteria and indicators following the specific features of critical mass tasks and the reversed concept of criticality. Next the task neglectedness potential of the project tasks is estimated using the multiple attributes decision-making method similarly the tasks criticalness potential (Brožová et al., 2016, 2019, Šubrt et al., 2019).

The first criterion for this assessment is the continuity criterion based on the degree of isolation of the task. The number of successors of the project task is crucial for its determination. The predecessors do not play an important role; there is no control mechanism for initiating the next task for predecessors. Suppose the project is formalized by an Activity On Node network graph. Then the continuity criterion can be derived from the output degree of the task node. Although it is significant for all relationship types, the major impact is in the case of FS relationship. If the output degree is greater than 2, sufficient successors can be assumed to control task completion, in the opposite site, there is lack of control of task completion. The 1 is added as each task has at least one successor – the dummy tasks "end of the project". The continuity criterion was set as

$$ci_i = \frac{deg_{out}i + 1}{2} \tag{1}$$

where ci_i is criterion of task continuity, $deg_{out}i$ is output degree of the node i. This criterion for the CMMC tasks is relevant if and only less or equal to 1.

The second criterion showing another view on potential task neglectedness combines the criteria of resource diversity and resource intensity. Again, it is considered that CMMC tasks are not resource intensive or resource variable. The analysis shows neglect threating, if no more than two resource types and/or no more than two assigned units appears. Criterion of resource variability is

$$crv_i = \frac{k_i}{2} \tag{2}$$

where crv_i is criterion of resource variability, k_i is number of resource types assigned to task i. Value of this criterion is CMMC relevant if and only if less or equal to 1.

Criterion of resource intensity evaluates the amount of resources units as

$$cri_i = \frac{\sum_{k=1}^{s} r_i^k}{2} \tag{3}$$

where cri_i is criterion of resource intensity, r_i^k is number of resource units assigned to task i, k is number of resource types assigned to task i, s is number of resource types assigned to task i. This criterion is relevant if and only if less or equal to 1.

These two criteria can be combined to the resource neglectedness criterion using their product

$$cr_i = crv_i . cri_i \tag{4}$$

This criterion is again CMMC relevant if and only if less or equal to 1.

Next criteria for evaluation of CMMC are directly derived from the factors based on the project schedule, and are compared with the aggregate indicators of the entire project (total duration, budget, total work effort) for normalization of its values. The reason for this is to eliminate the influence of dominant task and identify the tasks with minimal demands. Therefore, we can define four additional criteria:

The CMMC are tasks that are generally very short. Therefore, the duration neglectedness criterion of the Critical Mass task has to reach very low value

$$ct_i = \frac{t_i}{T} \tag{5}$$

where ct_i is task duration neglect criterion, t_i is duration of task i, T is total duration of the project.

Task cost neglectedness criterion also has to have very low value if the Critical Mass tasks are really inexpensive

$$cc_i = \frac{c_i}{C} \tag{6}$$

where cc_i is task cost neglect criterion, c_i is cost of task i, C is project budget at completion.

Task work neglectedness criterion describes the needed effort which is very low for the Critical Mass tasks

$$cw_i = \frac{w_i}{\sum_{i=1}^{N} w_i} \tag{7}$$

where cw_i is task work neglectedness criterion, w_i is amount of work of the task i, N is the number of tasks in the project

The task neglectedness criterion of the slack has a special position among the neglectedness criteria. It is based on the idea that CMMC are tasks that can be performed practically any time during project timespan, thus having a large total slack with respect to the duration of the project so the value $\frac{s_i}{T}$ is high. Therefore, the slack neglectedness criterion is

$$cs_i = 1 - \frac{s_i}{T} \tag{8}$$

where cs_i is task slack neglectedness criterion, s_i is total slack of task i, T is total duration of the project. This criterion is CMMC relevant if and only if less than 1.

The procedure of the analysis of the Critical Mass of the project has three steps:

In the first step the possible CMMC tasks are selected using the aspiration level method from all project tasks. These are all tasks with all values of continuity, resource variability, resource intensity and resource neglectedness criteria smaller than 1.

In the second step the CMMC potential is calculated using Simple weighted additive method so that the highest value means higher potential

$$CM_i = 1 - (u_1 ct_i + u_2 cc_i + u_3 cw_i + u_4 cs_i) \quad i = 1,2,..,N \tag{9}$$

where CM_i is global evaluation of the CMMC task potential, u_1, u_2, u_3, u_4 are the weights of partial criteria. The tasks with the higher Critical Mass potential are identified as possible Critical Mass tasks which would be the tasks whose duration, cost, work and slack neglectedness criteria have a lowest value (Šubrt et al., 2019).

The setting of individual criteria weights for this concept is not clear and easy. According to practice, the slack and duration neglectedness criteria are probably the most important, but this cannot be generalized.

In the third step the similarity of the tasks with higher Critical Mass potential is checked. The routine of tasks is situation when the tasks are repeated several times during a project progress in the same or analogous form. The routine is considered an essential feature for determining the Critical Mass and is quite difficult to capture, because in practice, these tasks does not have to be named identically (Substation Test 1, Substation Test 2, …), or assigned the same resources (revision technicians, …).

Each task is represented by vector of its evaluation according to the neglectedness criteria, so the task routine matrix is obtained, each task is in one row.

$$TRM = \begin{pmatrix} ci_1 & crv_1 & cri_1 & ct_1 & cc_1 & cw_1 & cs_1 \\ ci_2 & crv_2 & cri_2 & ct_2 & cc_2 & cw_2 & cs_2 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ ci_m & crv_m & cri_m & ct_m & cc_m & cs_m & cs_m \end{pmatrix} \tag{10}$$

where TRM is task routine matrix, m is a number of possible CMMC tasks.

We suggest to use the Saaty's compatibility index (Saaty, Peniwati, 2007) to investigate the similarity of potentially CM task. For two tasks T_i and T_i the compatibility index is calculated as

$$S_{ij} = (\frac{1}{n^2}) e^T A_i \otimes A_j^T e \leq 1.1 \tag{11}$$

where the elements of matrix A_i is calculated as pairwise comparisons matrix which consists of all ratios of every two elements from task i evaluation vector and $\otimes$ is the Hadamard product of matrices. If $S_{ij} \leq 1.1$ the tasks can be assumed as recurring (routine) tasks with neglectedness potential. The more lines in the TRM are compatible, the more routine tasks can be supposed in the project.

3 Example

The Critical Mass approach provides relevant results for larger and large projects, because the possibility of neglecting the timely execution of any task is small for small projects. However, let's assume the following small-scale project whose indicators, as well as the time schedule (Gantt Chart), are created using MS Project (Figure 1).

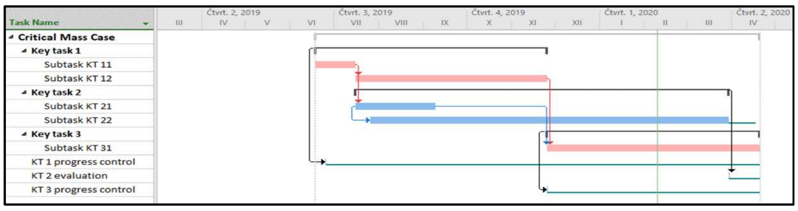


Figure 1 Example: Critical Mass Project Gantt Chart

All the tasks have assigned resources and work and all the resources are rated. Summary durations, costs and work parameters together with neglectedness criteria are presented in the following table (Table 1). For summary tasks the neglectedness criteria have no meaning.

Step 1: If continuity and resource criteria are bigger than one, the corresponding tasks are not supposed to be the critical mass tasks.

Step 2: Evaluation of the last column in Table 1 – Critical mass Potential – shows, that the highest risk of neglecting has “KT 1 progress control”, followed by “KT 3 progress control” and “KT 2 evaluation”. In case of bigger project, such type of tasks can potentially form a Critical Mass threating the project as a whole. The lower neglectedness potential has the task KT 11.

Task Name	Duration in dys	Total slack in dys	Cost in CZK	Work in hrs	Number of resources	Number of resource units	Number of successors	cii - izolovanost	crvi - počet zdrojů	crii - počet jednotek	cri	cti - délka	cci - náklady	cwi - práce	csi - rezerva	**TASK NEGLECT POTENCIAL**
Criteria weights								*0.125*	*0.125*	*0.125*	*0.125*	*0.125*	*0.125*	*0.125*	*0.125*	
												0.25	*0.25*	*0.25*	*0.25*	
Critical Mass Case	**218.88**	**0**	**13351080.00**	**8876**												
Key task 1	**113.88**	**0**	**3292160.00**	**1822**												
Subtask KT 11	20	0	288000.00	320	2	2	1	1	1	1	1	0.0914	0.0216	0.0361	1	0.7128
Subtask KT 12	93.88	0	3004160.00	1502	1	2	2	1.5	1	1	1	0.4289	0.2250	0.1692	1	
Key task 2	**183.88**	**14.95**	**6446480.00**	**4532**												
Subtask KT 21	38.88	14.95	466560.00	311	1	1	2	1.5	0.5	0.5	0.25	0.1776	0.0349	0.0350	0.9317	
Subtask KT 22	175.88	14.95	5979920.00	4221	2	3	0	0.5	1.5	1.5	2.25	0.8035	0.4479	0.4756	0.9317	
Key task 3	**105**	**0**	**3612000.00**	**2520**												
Subtask KT 31	105	0	3612000.00	2520	2	3	0	0.5	1.5	1.5	2.25	0.4797	0.2705	0.2839	1	
KT 1 progress control	0.1	213.78	2000.00	0.8	1	1	0	0.5	0.5	0.5	0.25	0.0005	0.0001	0.0001	0.0233	0.9940
KT 2 evaluation	0.05	14.95	1000.00	0.4	1	1	0	0.5	0.5	0.5	0.25	0.0002	0.0001	0.0000	0.9317	0.7670
KT 3 progress control	0.07	104.93	1400.00	0.56	1	1	0	0.5	0.5	0.5	0.25	0.0003	0.0001	0.0001	0.5206	0.8697

Table 1 Example: Task parameters and indicators

Step 3: Now the similarity among the selected tasks is checked using Saaty’s compatibility index. Only two task seems to be routine tasks - “KT 3 progress control” and “KT 2 evaluation”. Its compatibility index is 1.086.

4 Conclusion

Sometimes it happens that there are tasks in project management practice which seem to be not important. However, projects are sequences of tasks having an irreplaceable role and their failing or neglect may have fatal consequences, could endanger the whole project or significantly reduce its quality. The procedure proposed in this article allows to identify and analyse such tasks on the basis of the neglectedness criteria and neglectedness potential. The procedure allows to identify the critical mass task is also designed. Our approach should help project managers to have a wider and more diverse set of tools that make project management increasingly complex without distinguishing between complex and complicated.

5 Acknowledgements

The research is supported by the Operational Programme Prague – Growth Pole of the Czech Republic - Implementation proof-of-concept activities CULS to promote technology transfer and knowledge into practice, No: CZ.07.1.02/0.0/0.0/17_049/0000815 - KZ10.

References

Brožová, H., Bartoška, J., Šubrt, T. and Rydval, J., 2016, “Task Criticalness Potential: A Multiple Criteria Approach to Project Management”, Kybernetika, Vol. 52, pp. 558-574.

Brožová, H., Šubrt, T. Rydval, J. and Pavlíčková, P., 2019, “Fuzzy Threatness Matrices in Project Management”, In Proceedings of the 15th International Symposium on Operational Research in Slovenia, pp. 581-586. Bled, Slovenia.

Goldratt, E. M., 1997, “Critical Chain”, The North River Press, MA.

Nickolas, J. M. and Steyn, H., 2016, “Project Management for Business, Engineering and Technology”, Taylor and Francis.

InterPlan Systems, 2002, “Justification for Managing Turnarounds”, Available on: https://www.interplansystems.com/turnaround-project-management-primer/, [cit. 13.05.2019].

Saaty, T. L. and Peniwati, K., 2007, “Group decision-making: Drawing out and reconciling differences”, RWS Publications, Pittsburgh, PA.

Šubrt, T., Bartoška, J. and Kučera, P., 2019, “Critical Mass Task Identification in Projects”, In: Proceedings of the 37th International Conference on Mathematical Methods in Economics, České Budějovice, Czech Republic.

Pritsker, A. A.B., 1966, “GERT: Graphical Evaluation and Review Technique“, Memorandum RM-4973-NASA.

Solving the stochastic multimode resource-constrained project scheduling problem

Claudio Szwarcfiter[1], Avraham Shtub[2], and Yale T. Herer[3]

Faculty of Industrial Engineering and Management
Technion—Israel Institute of Technology, Haifa, Israel
[1]e-mail: claudioszw@campus.technion.ac.il

[2]e-mail: shtub@technion.ac.il

[3]e-mail: yale@technion.ac.il

Keywords: Stochastic resource-constrained project scheduling, multimode project management, stability and robustness in project management, reinforcement learning.

1. Introduction

The resource-constrained project scheduling problem (RCPSP) is a classic problem in project management, and its extensions, the multimode and the stochastic RCPSP (MRCPSP and SRCPSP), have received considerable attention. A standard procedure for solving these problems is the employment of heuristic methods, since the RCPSP is known to be NP-hard. However, less attention has been paid to the advances in artificial intelligence, particularly reinforcement learning (RL), and the opportunities they present for improving the search.

In this paper, we provide a novel RL-based approach for solving a version of the stochastic multimode RCPSP (SMRCPSP). This approach provides effective exploration of the search space, scanning a wide range of combinations of activity modes and start times, while simultaneously exploiting the learned knowledge. Our experiments currently being conducted suggest that the RL algorithm combines speed with performance close to the optimum.

2. Problem and solution approach

We model our SMRCPSP based on the flow-based formulation described in Artigues *et al.* (2015), expanding it for a multimode setting. Furthermore, we consider stochastic activity durations; therefore, the duration constraints cannot, in general, be guaranteed with certainty and thus we will model them as chance constraints. One way of solving the resulting stochastic program is by scenario optimization (SO), introduced in Calafiore and Campi (2005). The idea is to take *S* samples, or scenarios, of the realization of the random variables in the constraints—in our case, the activity durations—and substitute the deterministic scenario constraints for the stochastic chance constraints. The result is a mixed-integer linear program (MILP).

We consider a project with *J* activities. Each activity *j* can be executed in one of M_j modes and is preceded by a set of immediate predecessors $\mathrm{P}(j)$. Each activity *j* executed in mode *m* in scenario *s* has a duration d_{jms}. There are *K* different renewable resources. Activity *j* executed in mode *m* needs r_{jm}^k units of resource *k*, which has a total availability of R^k. ES_{js} and LS_{js} are the earliest and latest start for activity *j* in scenario *s*, respectively. Decision variable *D* is the project delivery date. We set parameter β as the desired probability of the project finishing within the delivery date, and θ as an upper bound for delivery date overrun. Binary decision variable δ_{jm} indicates if activity *j* is carried out in mode *m* and decision variable t_{js} denotes, for scenario *s*, the starting time of activity $j, j = 0,...,J+1$, where $j = 0$ and $J = J+1$ are dummy activities with a single mode, no duration and resources, and represent the start and end of the project, respectively. Binary decision variable z_{ij} indicates (value 1) if activity *j* starts after activity *i* finishes. The amount of resource *k* transferred from activity *i* to activity *j* is modeled by the flow variable ϕ_{ij}^k.

τ_s is a binary decision variable indicating whether, in scenario *s*, the project finishes within the delivery date. The model is as follows.

$$\text{Min}\, D, \tag{1}$$

Subject to:

$$t_{J+1,s} - \theta(1-\tau_s) \le D,\ \forall s = 1,...,S, \tag{2}$$

$$\sum_{s=1}^{S} \tau_s \ge \beta S, \tag{3}$$

$$z_{ij} + z_{ji} \le 1,\ \forall i = 0,...,J,\ \forall j = 1,...,J+1,\ \forall i < j, \tag{4}$$

$$z_{ij} + z_{jh} - z_{ih} \le 1,\ \forall i,j,h = 0,...,J+1, \forall i \ne j \ne h, \tag{5}$$

$$z_{ij} = 1,\ \forall i \in \mathrm{P}\left(j\right),\ \forall j = 1,...,J+1, \tag{6}$$

$$t_{js} - t_{is} - Mz_{ij} \ge \sum_{m=1}^{M_i} \delta_{im} d_{ims} - M,\ \forall i,j = 0,...,J+1,\ \forall i \ne j,\ \forall s = 1,...S, \tag{7}$$

$$ES_{js} \le t_{js} \le LS_{js},\ \forall j = 0,...,J+1,\ \forall s = 1,...,S, \tag{8}$$

$$\phi_{ij}^k - \min\left(\tilde{r}_{im}^k, \tilde{r}_{jm'}^k\right) z_{ij} - \left(1-\delta_{im}\right)\left(\tilde{r}_{ij}^{\max,k} - \min\left(\tilde{r}_{im}^k, \tilde{r}_{jm'}^k\right)\right) - \left(1-\delta_{jm'}\right)\left(\tilde{r}_{ij}^{\max,k} - \min\left(\tilde{r}_{im}^k, \tilde{r}_{jm'}^k\right)\right) \le 0,$$

$$\text{where } r_{ij}^{\max,k} = \max\left(\max_{m=1,...,M_i} \tilde{r}_{im}^k, \max_{m'=1,...,M_j} \tilde{r}_{jm'}^k\right) \text{ and } \tilde{r}_{j''m}^k = \begin{cases} r_{jm}^k & \text{if } 0 < j'' < n+1 \\ R^k & \text{if } j'' = 0 \text{ or } j'' = n+1, \end{cases} \tag{9}$$

$$\forall i = 0,...,J,\ \forall j = 1,...,J+1,\ \forall i \ne j,\ \forall k = 1,...,K,\ \forall m = 1,...,M_i,\ \forall m' = 1,...,M_j,$$

$$\sum_{m=1}^{M_j} \delta_{jm} = 1,\ \forall j = 0,...,J+1, \tag{10}$$

$$\sum_{j\in\{1,...,J+1\}\setminus\{i\}} \phi_{ij}^k = \sum_{m=1}^{M_i} \tilde{r}_{im}^k \delta_{im},\ \forall i = 0,...,J,\ \forall k = 1,...,K, \tag{11}$$

$$\sum_{i\in\{0,...,J\}\setminus\{j\}} \phi_{ij}^k = \sum_{m=1}^{M_j} \tilde{r}_{jm}^k \delta_{jm},\ \forall j = 1,...,J+1,\ \forall k = 1,...,K, \tag{12}$$

$$0 \le \phi_{ij}^k \le \min\left(\max_{m=1,...,M_i} \tilde{r}_{im}^k, \max_{m=1,...,M_j} \tilde{r}_{jm}^k\right),\ \forall i = 0,...,J,\ \forall j = 1,...,J+1,\ \forall i \ne j, \tag{13}$$
$$\forall k = 1,...,K.$$

The objective function (1) aims to minimize the project delivery time. Constraints (2) indicate whether a scenario finishes on time. Constraint (3) counts the fraction of scenarios that finish on time and forces it to remain above the predetermined threshold. Constraints (4) and (5) avoid cycles of 2 and 3 or greater, respectively. Constraints (6) enforce the precedence constraints. Constraints (7) link the continuous activity start time variables with the binary sequencing variables. Constraints (8) give upper and lower bounds for the activity start times. Constraints (9), from Balouka and Cohen (2019), connect the continuous resource flow variables with the binary sequencing variables and the binary mode variables. Constraints (10) force the selection of only one mode per activity. Outflow constraints (11) ensure that all activities, except for $J+1$, send their resources to other activities. Inflow constraints (12) ensure that all activities, except for activity 0, receive their resources from other activities. Constraints (13) bound the flow variables with the maximum resource consumption modes.

2.1. Reinforcement learning solution approach

Reinforcement Learning (RL) has been shown to be successful in diverse applications with uncertain environments. This success is the factor motivating the application of RL to our stochastic environment. To the best of our knowledge, multimode problems involving stochastic activity duration have not yet been tackled with RL.

Our RL model starts with an agent at project activity *j*. The agent undertakes an action by choosing a mode $\hat{m}_j$ and start time $\hat{t}_j$ for activity *j* and then moves on to the next activity. After selecting modes and start times for all activities $j = 1,...,J,$ she receives a reward $\mathrm{R}\left(j,m,t\right)$. The agent follows a policy $\pi(j,m,t)$ that tells her at each activity which action she should take. We further define an action-value function $q(j,m,t)$ as the estimated reward for taking an action at activity j and thereafter following policy $\pi(j,m,t)$. The RL problem's objective is to learn a policy that maximizes the agent's reward. We use Monte Carlo Control (MCC), based on Sutton and Barto (1998). Figure 1 presents our main MCC pseudocode.

```
Initialize action-values
**while not** stopping criterion:
    calculate policy
    choose mode and start time
    calculate reward
    update action-values $RL_1$
    or update action-values $RL_2$
```

Figure 1. MCC pseudocode.

Our algorithm starts with the initialization of the action-values table with artificially high values. The action-values table is then used to calculate the policy. To balance exploration and exploitation we adopt an ε-greedy policy, meaning that in the policy table we ascribe a probability ε of taking a random action and a probability $(1-\varepsilon)$ of taking a greedy action, i.e. the action with the highest action-value. Next, we take an action based on the policy, choosing for each activity the mode and start time according to the probabilities in the policy table. Then, we calculate the reward for the actions taken as $(1/D)$, where D is the delivery date for on-time probability β. The last step in the algorithm is to update the action-value table using the reward. We can choose from two update methods, RL_1 and RL_2: RL_1 learns an action-value by averaging all the rewards this action has received each time it was taken. RL_2 updates the action-values giving an exponentially large weight to the last action.

3. Experimental setting and partial results

To validate the RL procedure we propose a factorial experiment, summarized in Table 1, as follows. We will compare three project sizes, each with three modes per activity. For the 10-activity projects we will use the PSPLIB datasets (Kolisch and Sprecher, 1997), and for the 50 and 100-activity projects, the MMLIB datasets (Van Peteghem and Vanhoucke, 2014), generating additional data for the stochastic activity durations. We will run our RL algorithm using both methods for updating the action-values: RL_1 and RL_2, as described in Section 2.1. The delivery dates obtained with both variants will be compared to those from two benchmarks: the best combination of mode and activity priority rules (Peng, Huang and Yongping, 2015) and a solution for our MILP, using the Gurobi 8.1 solver. We will compare two types of constraints: solving the deterministic problem and then simulating realized durations to generate the delivery date, and solving directly the chance-constrained problem; in both cases, we will set the desired probability of the project finishing within the delivery date $\beta = 0.95$.

Table 1. Partial factorial design.

Project size	Algorithm	Constraints
10	RL_1	Chance constraints
50	RL_2	Deterministic
100	Solver	
	PR	

The algorithm is currently being executed and evaluated and we will be reporting the results in the conference. We present here partial results for 10-activity projects. Chance-constrained RL_1 (CRL_1) outperformed the other algorithms. Figure 2 provides a comparative view of project delivery for 10-activity projects; for clarity, we show only three curves: CRL_1, chance-constrained solver (CS) and deterministic-constrained priority rules (DPR). CRL_1, represented by the solid line, is consistently below the other curves. In fact, Wilcoxon signed rank tests for pairwise comparisons between CRL_1 and all the other methods, showed that CRL_1 generated shorter deliveries with p-value < 0.0001.

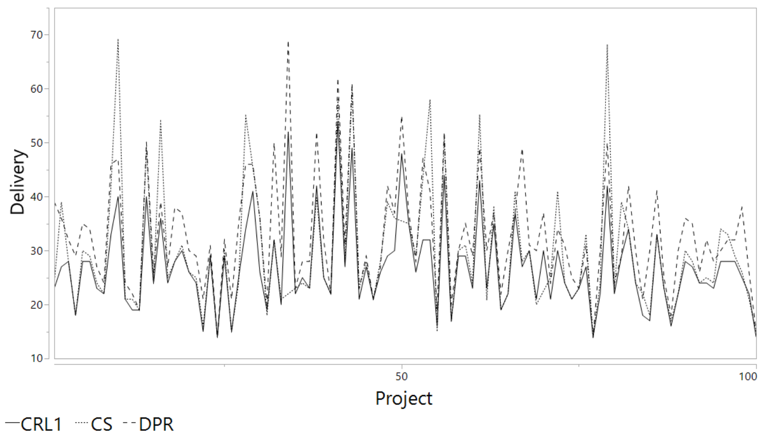


Figure 2. 10-activity projects: Overlay plot comparing CRL$_1$ with DPR and CS; for clarity, we show here a random subsample of 100 projects from the 535-project sample

4. Conclusions

In this paper, we presented a flow-based formulation of a variant of the SMRCPSP. The objective is to minimize the project delivery date and we introduce a constraint imposing a lower bound on the probability of finishing within this date. We described a novel RL-based approach for solving the problem and proposed a partial factorial design for the evaluation of our method. We have completed experiments for 10-activity projects and have concluded, with statistical significance, that for this project size, our RL approach renders shorter schedules than both the best priority rules, and the MILP solutions obtained with the solver using SO. We will be reporting the main results for the full experiment at the conference.

Acknowledgements

This study has received funding from EIT Food, the innovation community on Food of the European Institute of Innovation and Technology (EIT), a body of the EU under the Horizon 2020, the EU Framework Programme for Research and Innovation, project number 19147, and from the Bernard M. Gordon Center for Systems Engineering at the Technion.

References

Artigues, C. *et al.* (2015), 'Mixed-integer linear programming formulations', in *Handbook on Project Management and Scheduling Vol.1*. Cham: Springer International Publishing, pp. 17–41.

Balouka, N. and Cohen, I. (2019), 'A robust optimization approach for the multi-mode resource-constrained project scheduling problem', to be published in *European Journal of Operational Research* [Preprint].

Calafiore, G. and Campi, M. C. (2005), 'Uncertain convex programs: Randomized solutions and confidence levels', *Mathematical Programming*, 102(1), pp. 25–46.

Kolisch, R. and Sprecher, A. (1997), 'PSPLIB – A project scheduling problem library', *European Journal of Operational Research*, 96(1), pp. 205–216.

Peng, W., Huang, M. C. and Yongping, H. (2015), 'A multi-mode critical chain scheduling method based on priority rules', *Production Planning and Control*, 26(12), pp. 1011–1024.

Van Peteghem, V. and Vanhoucke, M. (2014), 'An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances', *European Journal of Operational Research*. North-Holland, 235(1), pp. 62–72.

Sutton, R. S. and Barto, A. G. (1998), *Reinforcement learning: An introduction*. MIT Press.

Maximizing value—Modeling and solving lean project management

Claudio Szwarcfiter[1], Avraham Shtub[2], and Yale T. Herer[3]

Faculty of Industrial Engineering and Management
Technion—Israel Institute of Technology, Haifa, Israel
[1]e-mail: claudioszw@campus.technion.ac.il

[2]e-mail: shtub@technion.ac.il

[3]e-mail: yale@technion.ac.il

Keywords: Lean project management, project scheduling, multimode project management, stability and robustness in project management.

1. Introduction

Lean Project Management (LPM) is a comprehensive framework with the goal of creating value and minimizing waste, in a minimum of time (Oehmen (Ed.), 2012). A key feature of LPM is the integration of two disciplines, which until now have been kept separate: program management and systems engineering. The former relates to project scope and the latter, to product scope.

The main contribution of our research is to provide a decision support tool for project managers by modeling and solving a novel LPM application. The model's objective is to maximize project value subject to due date and budget constraints. Moreover, our model tackles project risk, which often plays out in projects by causing unforeseen delays in activity durations, ultimately leading to due date and budget overruns. Therefore, our model considers stochastic activity durations, and the way we proactively manage risk is by generating a stable project plan in which the due-date and budget violation probabilities are kept within a desired threshold.

Project scope and product scope integration—which is key to LPM—is achieved in our model using a multimode approach, in which each project activity can be executed in one or more modes or alternatives. The integrative feature is that each activity mode, apart from containing information on the project scope, such as stochastic duration parameters, fixed and resource costs, also embodies data on the product scope, i.e., value parameters. Thus, when a mode is chosen, not only does the choice impact the project duration, cost, and associated risks, but also the project value. For example, for the activity "antenna design" in a radar project, there could be two modes: "re-engineer" and "new design"; associated with each mode we could have a radar range parameter, so that the mode choice would affect the overall radar range.

To the best of our knowledge, this is the first attempt to assist lean project managers and their teams with a decision support tool that models and solves the project planning problem focusing not only on the project scope (the work to be done) but also on the product scope (the features and functions of the product).

2. Problem and solution approach

We consider a project with *J* activities. Each activity *j* can be executed in one of M_j modes. The parameter V_{jmv} designates the value of attribute *v* for activity *j* executed in mode *m*. Decision variable V_{jv}' corresponds to the value of attribute *v* for activity *j* executed in its chosen mode. We define a function $F_v\left(V_{1v}',\ldots,V_{Jv}'\right)$ that determines the project value for each attribute *v* given the individual attributes V_{jv}', and a function $V''\left(F_1,\ldots,F_V\right)$ that calculates the project value given the values for each attribute. Binary decision variable δ_{jm} indicates if activity *j* is carried out in mode *m*. The objective of our model is to maximize the project value, which is a project-specific function of the chosen modes and can be a non-linear function:

$$\text{Maximize } V''\left(F_1\left(V_{11}',\ldots,V_{J1}'\right),\ldots,F_V\left(V_{1V}',\ldots,V_{JV}'\right)\right),$$

where

$$V_{jv}' = \sum_{m=1}^{M_j} \delta_{jm} V_{jmv}, \ \forall v = 1,...,V, \ \forall j = 1,...,J.$$

We formulate the deterministic version of our problem as a mixed integer program subject to duration and cost constraints. If we now consider stochastic activity durations, the duration constraints cannot, in general, be guaranteed with certainty and thus we model them as chance constraints. One way of solving the resulting stochastic program is by Scenario Optimization (SO), introduced in Calafiore and Campi (2005) and applied in recent project scheduling papers. The idea is to take *S* samples, or scenarios, of the realization of the random variables in the constraints—in our case, the activity durations—and substitute the deterministic scenario constraints for the stochastic chance constraints. Thus, if our objective function is linear, the new SO program is a mixed integer linear program (MILP), and can be solved with a commercial solver. We use this method as a benchmark in the computational experiments.

2.1. Reinforcement learning solution approach

Reinforcement Learning (RL) has been shown to be successful in diverse applications with uncertain environments. This success is the factor motivating our application of RL to learning the activity modes that maximize value in a stochastic environment. RL-based heuristics have also been applied to project scheduling, but to the best of our knowledge, problems involving stochastic activity duration or project value have not yet been tackled with RL. Hence, another contribution of our research is applying RL to this problem.

The RL model starts with an agent in a state Σ. The agent undertakes action A and moves to state S', receiving a reward R'. She then executes action A', moving to state S'' and receiving a reward R'', and so on. We can thus represent the agent's life trajectory as $\mathrm{S},\mathrm{A},\mathrm{R}',\mathrm{S}',\mathrm{A}',\mathrm{R}'',\mathrm{S}'',\mathrm{A}'',\mathrm{R}''',\mathrm{S}''',\mathrm{A}'''$, etc. The agent follows a policy $\pi(\mathrm{S},\mathrm{A})$ that tells her at each state which action she should take. The RL problem's objective is to learn a policy that maximizes the agent's reward. We further define an action-value function $q(\mathrm{S},\mathrm{A})$ as the estimated reward for taking action A on state Σ and thereafter following policy $\pi(\mathrm{S},\mathrm{A})$.

Applying the RL model to our problem, we define a state as project activity *j*. The agent undertakes an action by choosing a mode $\hat{m}_j$ for activity *j* and then moves on to the next activity. After selecting modes for all activities $(\hat{m}_j, \forall j = 1,...,J)$, she receives a reward, which we define as the project value V'' if, after a number of simulation runs with the chosen modes, the proportion of projects on time and on budget is more than or equal to the pre-defined on-time and on-budget probabilities; otherwise, the reward is zero. To balance exploration and exploitation, we employ *ε*-greedy policies, in which we set a probability *ε* of choosing a random mode for an activity; otherwise, a mode is chosen greedily, i.e., the one that has the highest action-value. We employ two alternative methods for updating the action-values. The first is Average Rewards (RL_1), in which the action-values are calculated by averaging the rewards accrued every time a certain mode is chosen for a certain activity. The second method is Constant Step (RL_2), which tries to leverage the learning by giving an exponentially larger weight to the last actions. We use a RL procedure known as Monte Carlo Control (MCC; based on Sutton and Barto, 1998), in which first we initialize the action-values; then, we calculate the policy, choose the activity modes based on the policy, calculate the reward accrued from this choice, and update the action-values using RL_1 or RL_2; we then once again calculate the policy using the updated action-values, and so on until the stopping criterion is met.

3. Experimental setting and main results

To validate the RL procedure we designed and conducted a factorial experiment, summarized in Table 1, as follows. We ran the algorithms with deterministic (zero risk) and stochastic activity durations and compared three project sizes, each with three modes per activity. For the 10-activity projects we used the PSPLIB datasets (Kolisch and Sprecher, 1997), and for the 50 and 100-activity projects, the MMLIB datasets (Van Peteghem and Vanhoucke, 2014). Both datasets are the standard in the multimode project management literature. We ran our RL algorithm using both methods for updating the action-values: RL_1 and RL_2, as described in Section 2.1. The project values obtained with both variants were compared to those from two benchmarks, a genetic algorithm (GA; Balouka, Cohen and Shtub, 2016) that seeks to maximize the project value for deterministic problems, with populations of 500, 1000 and 10,000, and a solution for our mixed integer program, in the case of linear objective functions (MILP), using the Gurobi 8.1 solver. For our stochastic settings, we developed a new fitness function for the GA:

$$f(I)=\begin{cases} V''(I), \text{ if } E(I)=0 \\ V''(I)-E(I)+Min\_V''(\text{feasible solutions})-Max\_V''(\text{all solutions}), \text{ otherwise,} \end{cases}$$

where $E(I)=\max\left(0,\hat{\beta}-\text{ proportion of on-budget runs}\right)$ for a solution I, i.e., the selected mode for each activity, and $\hat{\beta}$ is the desired probability of the project finishing within the budget.

Table 1. Partial factorial design: All the combinations were tested, except $GA_{10,000}$ for the stochastic trials and Solver for nonlinear objective functions.

Project risk	Project size	Algorithm	Objective function	Stopping criterion
Deterministic	10	GA_{500}	Linear	GA_S
Stochastic	50	GA_{1000}	Nonlinear	5 min
	100	$GA_{10,000}$		Max/near max
		RL_1		
		RL_2		
		Solver		

For each project, we generated two objective functions: a linear one, $0.6\sum_{j=1}^{J}V_{j1}'+0.4\sum_{j=1}^{J}V_{j2}'$ and a nonlinear one, $0.6\prod_{j=1}^{J}V_{j1}'+0.4\prod_{j=1}^{J}V_{j2}'$. For the linear objectives, we drew uniformly random value parameters V_{jmv} from the sets $\{0,1,2,...,10\}$, $\{0,0.2,0.4,...,2\}$, and $\{0,0.1,0.2,...,1\}$ for projects with $J=$ 10, 50 and 100 activities, respectively. For the nonlinear objectives, we generated uniformly random parameters from the sets $\left\{1+a\left(100^{1/J}-1\right)/5 \middle| a=0,1,...,5\right\}$ for experiments with $J=$ 10, 50 and 100 activities. Finally, we compared three stopping criteria: stopping RL_1 and RL_2 at the same execution time it took for the GA to converge (according to its published stopping criterion, two generations with the same best value; GA_S in Table 1);[1] five minutes, which is a reasonable running time for applications in industry; and the time it takes for the algorithms to give their best performance and produce a near-optimal solution ("max/near max" in Table 1) – by simulation, we estimated this time to be 20 seconds for the 10-activity projects and 2 hours for 50 and 100 activities.

Table 2 shows the results for the deterministic experiments with nonlinear objectives and GA_S stopping criterion (we show here only GA population 1000), and Table 3 shows the same results for the stochastic experiments. The columns show the number of projects tested, number of activities, GA population, average running time for the GA, and the average percent differences of the objective values; the alternative hypotheses H_1 are tested using one-tailed sign tests to evaluate whether one algorithm produces better solutions than the other and the p-values are presented. If no significant difference is found, a two-tailed sign test is conducted and the alternative hypothesis $a \neq b$ and p-value are recorded.

Table 2. Sign tests comparing RL to GA: Deterministic, nonlinear objective, stopping criterion GA_s.

Proj	Act	Pop_{GA}	T_{GA}(s)	RL_1-GA	H_1	P_V	RL_2-GA	H_1	P_V	RL_1-RL_2	H_1	P_V
535	10	1000	0.65	7.52	RL_1>GA	0.000	8.91	RL_2>GA	0.000	-1.23	RL_2>RL_1	0.000
540	50	1000	3.26	25.93	RL_1>GA	0.000	5.63	RL_2>GA	0.000	-0.94	RL_2>RL_1	0.003
540	100	1000	7.05	30.21	RL_1>GA	0.000	27.97	RL_2>GA	0.000	3.34	RL_1>RL_2	0.000

[1] This stopping criterion generated near-optimal solutions for 10- and 20-activity projects using populations of 500 and 1000 in Balouka, Cohen and Shtub, (2016).

Table 3. Sign tests comparing RL to GA: Stochastic, nonlinear objective, GA_S stopping criterion.

Proj	Act	Pop_{GA}	$T_{GA}(s)$	RL_1-GA	H_1	P_V	RL_2-GA	H_1	P_V	RL_1-RL_2	H_1	P_V
535	10	1000	119.18	7.70	$RL_1>GA$	0.000	7.44	$RL_2>GA$	0.000	1.80	$RL_1\neq RL_2$	0.166
120	50	1000	479.15	0.64	$RL_1\neq GA$	0.519	13.06	$RL_2>GA$	0.000	-9.99	$RL_2>RL_1$	0.000
71	100	1000	798.01	-3.43	$RL_1\neq GA$	1.0	4.50	$RL_2\neq GA$	1.0	-4.13	$RL_1\neq RL_2$	0.453

Considering a significance level of 0.05, we can conclude that for the GA_S stopping criterion in the deterministic experiments, RL_1 and RL_2 render better solutions than GA across the board, and the results are strongly significant. For the stochastic experiments, where the null hypothesis was rejected, similar results were obtained.

4. Conclusions

In this paper, we presented a new model for LPM, maximizing value and providing stability by complying with minimum on-time and on-budget probabilities set by the decision makers. We developed a stochastic programming model with a SO formulation. We introduced a new RL method to solve the problem, with two variants for action-value updates. We conducted a partial factorial experiment, both with small 10-activity and with larger 50- and 100-activity projects, comparing both RL variants with two benchmarks, a GA and, for linear objectives, a solution using a commercial solver, reaching the following conclusions, with statistical significance:

1. The RL methods are the best option when we want to find good solutions in less time, as they are much faster than the GA.
2. When given enough time to perform its best, the GA can outperform both RL variants, and for a fixed running time, the GA achieves better results with smaller populations (e.g., 500).
3. RL2 generally reaches good results faster than RL1, but when both variants are given enough time to perform their best, RL1 tends to give better results.
4. Although SO provides higher objective values for linear problems, it generates a higher proportion of infeasible solutions when these are simulated with test sets, apart from also resulting in long running times, typical of large MILP problems.

Our LPM modelling using RL opens avenues for new research. One research track could be the enhancement of the problem setting, introducing, for example, resource constraints and redefining the agent’s actions accordingly. Another track is the application of different RL methods, such as Q-Learning and function approximation.

Acknowledgements

This study has received funding from EIT Food, the innovation community on Food of the European Institute of Innovation and Technology (EIT), a body of the EU under the Horizon 2020, the EU Framework Programme for Research and Innovation, project number 19147, and from the Bernard M. Gordon Center for Systems Engineering at the Technion.

References

Balouka, N., Cohen, I. and Shtub, A. (2016) ‘Extending the multimode resource-constrained project scheduling problem by including value considerations’, *IEEE Transactions on Engineering Management*, 63(1), pp. 4–15.

Calafiore, G. and Campi, M. C. (2005) ‘Uncertain convex programs: Randomized solutions and confidence levels’, *Mathematical Programming*, 102(1), pp. 25–46.

Kolisch, R. and Sprecher, A. (1997) ‘PSPLIB – A project scheduling problem library’, *European Journal of Operational Research*, 96(1), pp. 205–216.

Oehmen (Ed.), J. (2012) *The guide to lean enablers for managing engineering programs*. Joint MIT-PMI-INCOSE Community of Practice on Lean in Program Management.

Van Peteghem, V. and Vanhoucke, M. (2014) ‘An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances’, *European Journal of Operational Research*. North-Holland, 235(1), pp. 62–72.

Sutton, R. S. and Barto, A. G. (1998) *Reinforcement learning: An introduction*. MIT Press.

Generating instances for the two-stage multi-machine assembly scheduling problem

Carla Talens[1], Victor Fernandez-Viagas[1] and Paz Perez-Gonzalez[1]

Industrial Management, School of Engineering, University of Seville
cartafa@us.es, vfernandezviagas@us.es, pazperez@us.es

Keywords: Benchmark, Assembly, Total Completion Time.

1 Introduction and description of the problem

The two-stage assembly scheduling problem consists of sequencing n jobs in a layout composed of two stages. Each job has $m_1 + 1$ operations. In the first stage, there are m_1 dedicated parallel machines, in which the first m_1 operations are conducted, while in the assembly stage there are m_2 identical parallel machines, being $m_2 \geq 1$. Only after all m_1 operations are completed, the assembly operation may start in an assembly machine. A job j has a processing time p_{ij} on machine i in the first stage and an assembly processing time at_j. The problem under study consists of scheduling the jobs in each machine so the total completion time is minimized.

In this contribution, we focus on the development of two new benchmarks for the two-stage assembly scheduling problem because two different variants of the problem are studied. The first one consists of several dedicated parallel machines in the first stage and one assembly machine in the second stage, and, following the notation by Framinan et al. (2019), it is denoted as $DP_{m_1} \rightarrow 1||\sum C_j$. In the second one, there are m_2 identical parallel machines in the last stage with $m_2 \geq 1$. It is denoted as $DP_{m_1} \rightarrow P_{m_2}||\sum C_j$.

2 New benchmark: parameters and generation procedure

In this section, we detail the characteristics, the parameters, and the generation procedure of the two new benchmarks. According to the works by Hall and Posner (2001) and Vallada et al. (2015), the next characteristics should be taken into account in the new proposed benchmarks: *adequacy, hardness, exhaustiveness* and *amenability for statistical analysis.*

2.1 Parameters

The first testbed, denoted as $\mathcal{B}_1$, is a set of 240 instances with the following combinations of number of jobs (n), number of machines in the first stage (m_1) and number of machines in the second stage (m_2): $n \in \{50, 100, 150, 200, 250, 300\}$, $m_1 \in \{2, 4, 6, 8\}$ and $m_2 = 1$. The second testbed, denoted as $\mathcal{B}_2$, is a set of 960 instances where $n \in \{50, 100, 150, 200, 250, 300\}$, $m_1 \in \{2, 4, 6, 8\}$ and $m_2 \in \{2, 4, 6, 8\}$. The values have been taken based on Al-Anzi and Allahverdi (2006) and Allahverdi and Al-Anzi (2012). Both testbeds have 10 associated instances for each combination. Note that, we have selected both a wide range of levels and equidistant values for the number of jobs and machines to fulfil the exhaustiveness and amenability requirements.

2.2 Adequacy

With respect to the adequacy characteristic, we should generate instances which exactly suit the problem under study and not related problems. In our case, the instances should be representative of the two-stage multi-machine assembly scheduling problem. Therefore, the relationship between this problem and the related scheduling problems, such as the Customer Order (CO) scheduling problem and the Parallel Machines (PM) scheduling problem has to be analysed. To perform this analysis, we carry out a preliminary experiment which lead us to generate the processing times in the first stage from a uniform distribution $U[1, 100]$ and in the second stage from a uniform distribution $U[1, \alpha 100]$. Parameter α is designed to balance the connection between both stages. The next values of α are tested: $\alpha = \{0.5, 1, 2, 3\}$.

We generate two preliminary small testbeds. The first one, testbed A, consists of 540 instances with $n \in \{8, 10, 12\}$, $m_1 \in \{2, 4\}$ and $m_2 = 1$, and, the testbed B consists of 1,080 instances with $n \in \{8, 10, 12\}$, $m_1 \in \{2, 4\}$ and $m_2 = \{2, 3\}$. Then, we solve them applying exact methods in order to identify representative instances of the problem under study and the related problems. To do so, firstly, we have adapted to our problem the mathematical model found in the paper by Navaei et al. (2013). Then, we have modified it to solve the CO scheduling problem. And, finally, we take the SPT rule plus the First Available Machine rule to solve the PM scheduling problem. Therefore, we hold three exact methods to solve the three different scheduling problems.

Solving each instance by the three methods we obtain the optimal sequences for each case denoted as MMA^*, CO^* and PM^*. The evaluation of the objective function of these schedules for the problem MMA are denoted as $MMA(MMA^*)$, $MMACO^*$ and $MMA(PM^*)$. These values allow us to analyse the relationship between the problem under study and the related problems (CO and PM).

To determine how is the relationship between the three problems we calculate the Relative Percentage Deviation (RPD) of the $MMACO^*$ and $MMA(PM^*)$, as follows:

$$RPD_{hs} = \frac{C_{hs} - C_s^*}{C_s^*} \cdot 100 \tag{1}$$

with C_{hs} the total completion time obtained by $h \in (MMA(CO^*), MMA(PM^*))$ in instance s ($s = 1, \ldots, S$) and C_s^* the minimum completion time known for instance s. Then, the ARPD is computed as the average of all the instances.

Low values of ARPD mean that the problems are highly connected, i.e. the fact that $ARPD_{MMA(CO^*)}$ is low means that the instances can be solved by methods of the CO scheduling problem. Contrarily, high values of ARPD denote that the problem under study is not related to the other problem.

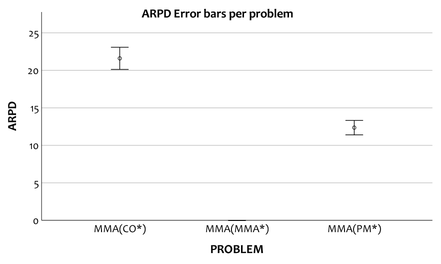


Fig. 1: ARPD of exact methods when $m_2 = 1$ and $\alpha = 2$

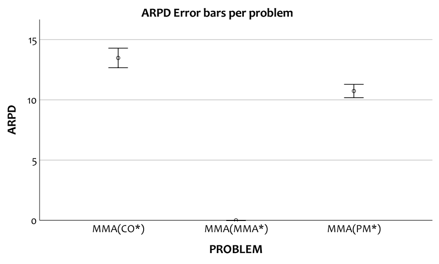


Fig. 2: ARPD of exact methods when $m_2 \geq 2$ and $\alpha = 2$

For the $DP_{m_1} \rightarrow 1||\sum C_j$ problem, Figures 1 and 2 show the relationship with CO and PM when $\alpha = 2$, since the ARPD in both cases are high (greater than 10), so the instances are representative of our problem. When $\alpha = 0.5$, the instances seem to be more representative of the CO scheduling problem and, when $\alpha = 3$, more representative of the PM scheduling problem. The same results are obtained for the $DP_{m_1} \rightarrow P_{m_2}||\sum C_j$ problem as it can be seen in Figure 2.

Next, we should verify that the behaviour of these instances are also fulfilled when the number of jobs and machines increases. As we can not solve bigger instances applying exact methods, we solve the preliminary small testbeds by an approximate method, and then, we compare these results with those obtained solving bigger instances with the same approximate method. In this case, we apply the heuristic by Framinan and Perez-Gonzalez (2017), adapted to $DP_{m_1} \rightarrow P_{m_2}||\sum C_j$ in the case with more than one machine in the second stage.. As it can be seen in Figures 3 and 4, when the number of jobs, n, increases, the instances have the same behaviour as when n is low.

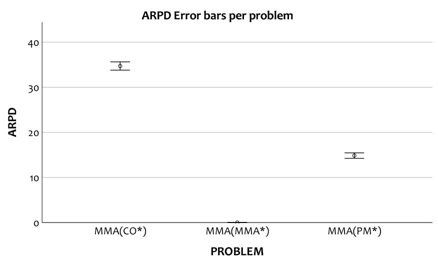


Fig. 3: ARPD of the heuristic when $m_2 = 1$ and $\alpha = 2$

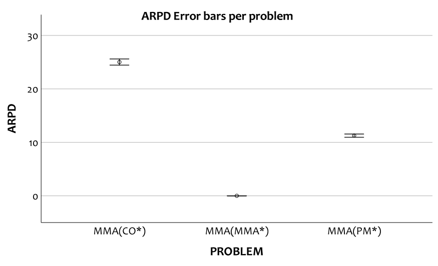


Fig. 4: ARPD of the heuristic when $m_2 \geq 2$ and $\alpha = 2$

2.3 Empirical hardness

In order to determine the empirical hardness, we use the following approach which is based on the difference between a well-performing metaheuristic and a bound of the problem (see Taillard, 1990 and Vallada et al., 2015). For each combination of n, m_1 and m_2, we generate 1,000 instances. So, in total 120,000 instances are generated: 24,000 for the combinations when $m_2 = 1$ and 96,000 for the combinations when $m_2 \geq 2$. 10 instances are selected according to the following procedure: To test the difficulty of each instance by, firstly, computing a lower bound and, secondly, solving it by the Iterated Greedy, IG, algorithm developed by Ruiz and Stützle (2007). We have adapted the lower bound developed by Blocher and Chhajed (2008), which is computed by Equation 2, to the problem under study, where $w_j = \sum_{\forall i} \frac{p_{ij}}{m_1}$. The IG works as follows: The NEH heuristic (Nawaz et al., 1983) gives the initial solution of the IG. The central procedures are the destruction and the construction phases. Then, a local search procedure is carried out by improving each solution generated in the construction phase. Finally, the new sequence is accepted or not as the incumbent solution for the next iteration by a Simulated Annealing.

$$LB = \sum_j \left(max\{\sum_{\forall k \leq j} \lceil w_j \rceil, max_{\forall i} \sum_{\forall k \leq j} p_{ij}\} \right) + \sum_{\forall j} p_{jA} \tag{2}$$

We set a stopping criterion depending on the size of the instance and the complexity of computing the objective function (the total completion time) of this problem: $n \cdot m/2 \cdot 90/1000$ milliseconds, where m is equal to $m_1 + 1$. Then, we obtain, for each instance, the

ARPD of the IG with respect to the lower bound. Following the idea by Vallada et al. (2015), the higher the $ARPD_{IG}$, the harder the instance is, i.e. if the solution founded by the IG is further from the theoretical lower bound, the instance is hard to solve.

So, the procedure to obtain the hardest instances per combination is as follows: the $ARPD_{IG}$ for the 1,000 instances are sorted in descending order. Then, the 10 first instances per combination are selected to be part of the new benchmark. As a result, a new benchmark with 240 instances ($DP_{m_1} \rightarrow 1||\sum C_j$) and another one with 960 instances ($DP_{m_1} \rightarrow P_{m_2}||\sum C_j$) are generated.

Regarding the exhaustiveness, the benchmark consists of a large number of instances, 240 and 960 respectively, and different size of the parameters and different combinations have been considered. Finally, the benchmark is amenable for statistical analysis since the levels of the parameters are equidistant and all the levels have been combined to generate the instances.

Acknowledgements

This research has been funded by the Spanish Ministry of Science and Innovation, under the project “PROMISE” with reference DPI2016-80750-P.

Bibliography

Al-Anzi, F. S. and Allahverdi, A. (2006). A Hybrid Tabu Search Heuristic for the Two-Stage Assembly Scheduling Problem. *International Journal of Operations Research*, 3(2):109–119.

Allahverdi, A. and Al-Anzi, F. (2012). A new heuristic for the queries scheduling problem on distributed database systems to minimize mean completion time. In *Proceedings of the 21st International Conference on Software Engineering and Data Engineering, SEDE 2012.*

Blocher, J. D. and Chhajed, D. (2008). Minimizing customer order lead-time in a two-stage assembly supply chain. *Annals of Operations Research*, 161(1):25–52.

Framinan, J. M. and Perez-Gonzalez, P. (2017). The 2-stage assembly flowshop scheduling problem with total completion time: Efficient constructive heuristic and metaheuristic. *Computers and Operations Research*, 88:237–246.

Framinan, J. M., Perez-Gonzalez, P., and Fernandez-Viagas, V. (2019). Deterministic assembly scheduling problems: A review and classification of concurrent-type scheduling models and solution procedures. *European Journal of Operational Research*, 273:401–417.

Hall, N. and Posner, M. (2001). Generating experimental data for computational testing with machine scheduling applications. *Operations Research*, 49(6):854–865.

Navaei, J., Fatemi Ghomi, S. M. T., Jolai, F., Shiraqai, M. E., and Hidaji, H. (2013). Two-stage flow-shop scheduling problem with non-identical second stage assembly machines. *International Journal of Advanced Manufacturing Technology*, 69(9-12):2215–2226.

Nawaz, M., Enscore Jr., E., and Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95.

Ruiz, R. and Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049.

Taillard, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 47(1):65–74.

Vallada, E., Ruiz, R., and Framinan, J. (2015). New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*, 240(3):666–677.

Metric Estimations for a Resource Leveling Problem With Variable Job Duration

Ilia Tarasov[1,3], Alain Haït[1], Olga Battaïa[2], and Alexander Lazarev[3]

[1] ISAE-SUPAERO, University of Toulouse, France
Ilia.TARASOV@isae-supaero.fr
[2] Kedge Business School (Talence), Talence Cedex, France
[3] V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Russia

Keywords: project scheduling, resource leveling.

1 Introduction

We consider a Resource Leveling Problem (RLP), which remains an object of intensive research in project scheduling. It is possible to apply this formulation for many real-life cases, such as transportation, production, education, and management. It is known to be NP-hard in the strong sense, see Neumann et al. (2002). A set of project jobs must be scheduled in a planning horizon without preemption, jobs require a different amount of resources, and resource capacity is given during the planning horizon. There are also precedence relations between jobs. For the RLP, three basic types of the objective function are known in the literature: the total squared resource utilization cost, total overload cost, and the total adjustment cost (see Rieck & Zimmermann (2015)).

Usually, these basic formulations are enriched with new features and constraints to make the model closer to a particular practical case. Hans (2001) presented a branch-and-price approach for a Resource Leveling Problem which allows jobs to be implemented non-uniformly, i.e. jobs have different intensities in different moments (discrete-time periods). Kis (2005) constructed the branch and cut algorithm for this model which outperformed branch and price approach. We focus on the total overload cost objective, a weighted sum of resource amount which is used in addition to given available resource amount to fit in the project deadline constraint. This case was studied by Bianco et al. (2016) and Baydoun et al. (2016). Bianco et al. (2016) presented a solution approach for the model with generalized precedence relations with time lags (GPR). Baydoun et al. (2016) review the case when the partial intersection of predecessor and successor job is allowed. To the best of our knowledge, these formulations are the closest to our model. In this paper we study a generalized version of the RLP. It was introduced earlier (see Tarasov et al. (2019)) with a comparison of solution quality to the concepts applied by Bianco et al. (2016) and Baydoun et al. (2016).

2 Problem formulation

Our problem planning horizon is divided into $T = \{1, ..., m\}$ periods of given length d. All jobs have their minimal duration superior to d. There is a fixed deadline for all jobs which is the end of the planning horizon. Available workload for each resource type $r \in R$ in each period $t \in T$ is limited by the free work volume L_{rt}. Additional workload for this resource type $r \in R$ costs e_r. The duration of the job is not given and can also vary in some limits, depending on the total workload required to implement this job by different resources. For each job $j \in J$ required workload W_{jr} for each resource $r \in R$ is given, with upper and lower bounds on assigned workload per period defined by multipliers, $p_{min,jr}$ and $p_{max,jr}$, $r \in R$. If some job $j \in J$ is implemented during time d_{jt} in period $t \in T$, assigned workload has lower and upper bounds $p_{min,jr}d_{jt}$ and $p_{max,jr}d_{jt}$, respectively. Precedence relations are represented in a directed graph which should be acyclic. We present all problem instance parameters and decision variables in Table 1, and the MILP model from Tarasov et al. (2019) below.

Table 1. Problem instance parameters and decision variables

	Parameters
T	planning horizon, $T = \{1, ..., m\}$
d	period length
R	resources set
L_{rt}	available workload of resource $r \in R$ in period $t \in T$
e_r	extra resource cost
J	jobs set
W_{jr}	job $j \in J$ workload on the resource $r \in R$
$p_{min,jr}$	job $j \in J$ minimum assigned workload of resource $r \in R$
$p_{max,jr}$	job $j \in J$ maximum assigned workload of resource $r \in R$
P	the set of arcs in precedence graph
	Decision variables
S_{jt}	binary step : equals 1 if job $j \in J$ starts in $\forall t_1 \in T$, $t_1 \leq t$, 0 otherwise
E_{jt}	binary step : equals 1 if job $j \in J$ ends in $\forall t_1 \in T$, $t_1 < t$, 0 otherwise
d_{jt}	duration of job $j \in J$ in period $t \in T$
c_{jrt}	work volume of the job $j \in J$ on the resource $r \in R$ in period $t \in T$
o_{rt}	extra cost of the resource $r \in R$ in period $t \in T$

$$S_{jt} \geq E_{jt},\ S_{jt} \leq S_{j,t+1},\ E_{jt} \leq E_{j,t+1},\ \forall j \in J,\ \forall t \in T; \tag{1}$$

$$d_{jt} \leq d\ (S_{jt} - E_{jt}),\ \forall j \in J,\ \forall t \in T. \tag{2}$$

$$d_{jt} \geq d\ (S_{jt} + S_{j,t-1} - 1 - E_{jt} - E_{j,t+1}),\ \forall j \in J,\ \forall t \in T. \tag{3}$$

$$S_{j_2 t} \leq E_{j_1,t+1},\ d_{j_1 t} + d_{j_2 t} \leq d,\ \forall t \in T,\ \forall (j_1, j_2) \in P; \tag{4}$$

$$d_{j_1 t} + d_{j_2 t} \leq d,\ \forall t \in T,\ \forall (j_1, j_2) \in P. \tag{5}$$

$$p_{min,jr} d_{jt} \leq c_{jrt} \leq p_{max,jr} d_{jt},\ \forall j \in J,\ \forall r \in R,\ \forall t \in T; \tag{6}$$

$$\sum_{t \in T} c_{jrt} = W_{jr},\ \forall j \in J,\ \forall r \in R; \tag{7}$$

$$o_{rt} \geq e_r(\sum_{j \in J} c_{jrt} - L_{rt}),\ \forall t \in T,\ \forall r \in R; \tag{8}$$

The objective function of our problem is represented in the following form: $\sum_{r \in R} \sum_{t \in T} o_{rt}$.

3 Metric approach in scheduling

In this paper, we study the scheduling problem solution approach, which was presented by Lazarev (2009). The latter approach has been shown to be effective in dealing with some NP-hard scheduling problems, including the total tardiness minimization problem Lazarev et al. (2017) and the single-machine scheduling problem Lazarev & Kvaratskheliya (2010).

The idea of this approach is to use polynomially solvable subcases of the problem, which is NP-hard in general. It allows producing the solution for an arbitrary instance with a guaranteed accuracy (objective function value difference) in polynomial time. Any scheduling problem input data instance represents the point in $f(n)$-dimensional space Ω, where n is the number of jobs in the instance. Firstly, if the same schedule π (the permutation of jobs) is used as the solution for two different arbitrary instances A and B, it is possible to make the estimation of the difference between objective function values $V_A(\pi)$ and $V_B(\pi)$ for these instances. This estimation is formed as the metric $\rho(A,B)$ defined on $\Omega \times \Omega$, such that

$$|V_A(\pi) - V_B(\pi)| \leq \rho(A,B).$$

Secondly, suppose there are two schedules π^A and π^B that are the optimal solutions for instances A and B, respectively. It also possible to construct the estimation for the expression

$$V_A(\pi^B) - V_A(\pi^A) \leq \Delta(\rho(A,B))$$

and prove that it depends on $\rho(A,B)$. Then $\Delta(\rho(A,B))$ is the absolute accuracy for the case when π^B is used as the solution for instance A instead of the real optimal solution. The original idea is to use the schedule π^B that is optimal for some polynomially solvable instance B as the solution for the original problem instance A, such that the difference between the objective function values is minimum, i.e., the value of $\rho(A,B)$ is minimum.

4 Application to a Resource Leveling Problem

Although in the case of a Resource Leveling Problem it is difficult to specify the solvable subclasses, it is possible to get the estimations. Our problem provides more possible options to construct the proper estimations $\rho(A,B)$. Therefore, it allows to make the following steps:

1. the construction of a metric function $\rho(A,B)$ for arbitrary instances A and B in case of particular model
   - present model implies some feasibility criteria on the solution π to be used for particular instance A;
   - we provide the objective function value difference estimations for instances A and B if the difference is only in some particular parameter (e.g. L_{rt}, W_{jr} etc.);

   Lemma 1. *An example for L_{rt}. Consider the instances A and B, which are different only in the parameters L_{rt}. If we apply the same solution σ to the both instances, the upper bound for objective function values difference is*

$$|V^A(\sigma) - V^B(\sigma)| \leq \rho_L(A,B) = \sum_{r\in R} e_r \sum_{t\in T} |L^A_{rt} - L^B_{rt}|. \qquad (9)$$

   Proof.

$$|V^A(\sigma) - V^B(\sigma)| = |\sum_{r\in R}\sum_{t\in T} o^A_{rt} - \sum_{r\in R}\sum_{t\in T} o^B_{rt}|, \qquad (10)$$

   here $o_{rt} = \max\{0, e_r(\sum\limits_{j\in J} c_{jrt} - L_{rt})\}$, and taking into account that costs are equal $e^A_r = e^B_r = e_r$ and $|\max\{a,b\} - \max\{c,d\}| \leq \max\{|a-c|, |b-d|\}$,

$$|V^A(\sigma) - V^B(\sigma)| \leq \sum_{r\in R}\sum_{t\in T} |e^A_r(\sum_{j\in J} c_{jrt} - L^A_{rt}) - e^B_r(\sum_{j\in J} c_{jrt} - L^B_{rt})| \leq$$

$$\leq \sum_{r\in R} e_r \sum_{t\in T} |L^A_{rt} - L^B_{rt}|. \qquad \square$$

  - we consider the objective function value difference estimations when several parameter types vary from instance A to B;
  - it is important to prove that general case estimations can be combined from the particular parameter changes and the impact is not multiplied;
2. the formulation of estimations $V_A(\pi^B) - V_A(\pi^A) \leq \Delta(\rho(A, B))$ of the difference between objective function values of optimal solutions for arbitrary instances A and B.

5 Conclusion

To sum up, we propose to apply the metrization approach to a Resource Leveling Problem. This approach allows studying the error upper bound when the given optimal solution of the instance is used as a suboptimal solution for another instance (which may differ in some parameters). The general idea is to provide theoretical estimations of the guaranteed absolute accuracy in this case and use these properties to deal with some real-life issues, for example, data uncertainty.

Future work includes the detailed study of possible improvements of these estimations and the allocation of useful subclasses of instances that are solved in a reasonable time. The second goal is to study useful applications to uncertainty cases and numerical experiments, for example:

- describe the conditions when there are some changes in the instance data, but the same solution (or the scheduling part, in particular) still remains optimal;
- provide the proven accuracy solutions for the cases with uncertain data in some parameters (e.g. presented by ranges).

References

Baydoun, G., Haït, A., Pellerin, R., Cément, B. & Bouvignies, G. (2016), ‘A rough-cut capacity planning model with overlapping’, *OR Spectrum* **38**(2), 335–364.

Bianco, L., Caramia, M. & Giordani, S. (2016), ‘Resource levelling in project scheduling with generalized precedence relationships and variable execution intensities’, *OR Spectrum* **38**(2), 405–425.

Hans, E. (2001), Resource Loading by Branch-and-Price Techniques, PhD thesis, Twente University Press (TUP), Netherlands.

Kis, T. (2005), ‘A branch-and-cut algorithm for scheduling of projects with variable-intensity activities’, *Mathematical Programming* **103**(3), 515–539.

Lazarev, A. A. (2009), ‘Estimates of the absolute error and a scheme for an approximate solution to scheduling problems’, *Computational Mathematics and Mathematical Physics* **49**(2), 373–386.

Lazarev, A. A., Korenev, P. S. & Sologub, A. A. (2017), ‘A metric for total tardiness minimization’, *Automation and Remote Control* **78**(4), 732–740.

Lazarev, A. A. & Kvaratskheliya, A. G. (2010), ‘Metrics in scheduling problems’, *Doklady Mathematics* **81**(3), 497–499.

Neumann, K., Schwindt, C. & Zimmermann, J. (2002), *Resource-Constrained Project Scheduling — Minimization of General Objective Functions*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 175–299.

Rieck, J. & Zimmermann, J. (2015), *Exact Methods for Resource Leveling Problems*, Springer International Publishing, Cham, pp. 361–387.

Tarasov, I., Haït, A. & Battaïa, O. (2019), ‘A generalized milp formulation for the period-aggregated resource leveling problem with variable job duration’, *Algorithms* **13**(1). **URL:** *https://www.mdpi.com/1999-4893/13/1/6*

Open shop problem with agreement graph: new results

Nour ElHouda TELLACHE[1], Mourad BOUDHAR[2] and Farouk YALAOUI[3]

[1] CERMICS Laboratory, ENPC, 6-8 avenue Blaise Pascal, 77455 Marne-la-Vallée Cedex 2, France
nour.tellache@gmail.com
[2] RECITS Laboratory, USTHB university, BP 32 El-Alia,Bab-Ezzouar, Algiers, Algeria
mboudhar@yahoo.fr
[3] LOSI Laboratory, UTT university, 12 rue Marie Curie BP 2060, 10010, Troyes Cedex, France
farouk.yalaoui@utt.fr

Abstract. This paper deals with the problem of scheduling on a two-machine open shop subject to constraints given by an agreement graph G, such that jobs can be processed simultaneously on different machines if and only if they are represented by adjacent vertices in G. The problem of minimizing the maximum completion time (makespan) is known to be NP-hard. In this work, we study the complexity of the problem when restricted to trees. Then, we present six Mixed Integer Linear Programming (MILP) models along with an experimental study to test their performance.

Keywords: open shop scheduling, agreement graph, complexity, MILP, makespan.

1 Introduction

The Open Shop problem with Agreement graph (OSA) which is discussed in this paper can be described as follows. The inputs consist of a finite set $\{J_j, j = 1, \ldots, n\}$ of n jobs that has to be processed on a set $\{M_i, i = 1, \ldots, m\}$ of m machines and a simple graph $G = (V, E)$ over the jobs, called the agreement graph. Each job J_j consists of m operations J_{ij} $(i = 1, \ldots, m)$, where J_{ij} has to be processed on machine M_i for $p_{ij} \geq 0$ time units. The order in which the jobs are processed on the machines is not fixed. On the other hand, each vertex in G represents a job and two jobs can be processed at the same time on different machines (are not in conflict) if and only if they are adjacent in G. The objective is to find a feasible schedule that minimizes the maximum completion time (makespan). According to the three field classification $\alpha/\beta/\gamma$ of Graham et al. [4], we denote our scheduling problem by $O2|AgreeG = (V, E)|C_{max}$, where $AgreeG = (V, E)$ indicates the presence of an agreement graph $G = (V, E)$ over the jobs. The proportionate processing times assumption implies that each job J_j has the same processing requirement p_j on each machine ($p_{ij} = p_j$ for all jobs J_j and all machines M_i); in that case, the resulting problem is called the proportionate OSA problem and it is denoted $Om|AgreeG = (V, E), prpt|C_{max}$.

In some practical applications, the jobs may require, besides the machines, some additional non-sharable resources with limited capacities for their processing [1]. In this case, two jobs can be processed simultaneously on different machines if the total requirement of at least one resource does not exceed its capacity. Therefore, this problem can be modeled as an OSA problem. For more details about the correspondence between the OSA problem and the open shop under resource constraints, the interested reader is referred to [6].

2 Literature review

Scheduling with agreement graph $G = (V, E)$ is equivalent to scheduling with conflict graph $\overline{G} = (V, \overline{E})$ (complement of the agreement graph). Tellache and Boudhar [6] studied the Open Shop problem with Conflict graph (OSC). They showed that the two-machine OSC problem with $p_{ij} \in \{1, 2, 3\}$ is NP-hard in the strong sense even when restricted to complements of bipartite graphs. The same result holds for the three-machine OSC problem with $p_{ij} = 1$ and an arbitrary conflict graph. After that, efficient algorithms were proposed for the two-machine OSC problem with $p_{ij} \in \{0, 1, 2\}$, and for the three-machine OSC problem with $p_{ij} = 1$ and $\overline{G}$ being a complement of triangle-free graph. They also proved that by allowing preemption, the two-machine OSC problem becomes easy to solve for arbitrary conflict graphs. On the other hand, they found that the OSC problem is polynomially equivalent to a special case of the open shop under resource constraints, from which new complexity results of the latter problem were established. They also presented a two-phase heuristic approach and lower bounds for the general m-machine OSC problem. In [5], the authors considered the same problem. They first proved the NP-hardness of the case of two values of processing times and more general agreement graphs which closes definitely the complexity status of the problem. Then, they presented some restricted cases that can be solved in polynomial time. They also derived new complexity results of the open shop under resource constraints and of the partition into triangles problem.

3 NP-hardness results

In this Section, we show that the OSA problem is NP-hard even when restricted to trees. The problem used in the reduction process is the 2-partition problem [3].

Theorem 1. *The problem $O2|AgreeG = (V, E)|C_{max}$ is NP-hard in the ordinary sense for G being a tree.*

In the following theorem, we consider the proportionate open shop problem.

Theorem 2. *The problem $O2|AgreeG = (V, E), prpt|C_{max}$ is NP-hard in the ordinary sense for G being a tree.*

4 Mathematical models

The following models are proposed for the problem $O|AgreeG = (V, E)|C_{max}$. The parameters used are:

- a_{jk}: 1 if J_j and J_k are in conflict, 0 otherwise.
- M: big constant.

The decision variables of the first model are:

- C_{max}: the maximum completion time determined by the completion time of the last operation.
- C_{ij}: completion time of job J_j on machine M_i.
- x_{ijk}: 1 if job J_j is scheduled any time before J_k on machine M_i, 0 otherwise.
- $y_{ii'j}$: 1 if job J_j is scheduled on M_i then on $M_{i'}$, 0 otherwise.
- $r_{ii'}^{jk}$: when $a_{jk} = 1$, this variable is equal to 1 if the operation J_{ij} is scheduled any time before $J_{i'k}$, 0 otherwise..

The MILP model is summarized in (P_1).

$$(P_1)\left|\begin{array}{llll}
\min C_{max} & & & \\
\text{S.C } C_{max} \geq C_{ij}; & i=1,\ldots,m; & j=1,\ldots,n, & (1)\\
\quad C_{ij}-M(1-x_{ikj}) \leq C_{ik}-p_{ik}; & i=1,\ldots,m, & 1\leq j<k\leq n, & (2)\\
\quad C_{ij}-C_{ik}-p_{ij} \geq -Mx_{ikj}; & i=1,\ldots,m, & 1\leq j<k\leq n, & (3)\\
\quad C_{ij}-M(1-y_{ii'j}) \leq C_{i'j}-p_{i'j}; & 1\leq i'<i\leq m, & j=1,\ldots,n, & (4)\\
\quad C_{ij}-C_{i'j}-p_{ij} \geq -My_{ii'j}; & 1\leq i'<i\leq m, & j=1,\ldots,n, & (5)\\
\quad C_{ij}-M(1-r_{iji'k}) \leq C_{i'k}-p_{i'k}; & \text{if } a_{jk}=1,\ (i\neq i'), i,i'=1,\ldots,m,, & j=1,\ldots,n, & (6)\\
\quad C_{ij}-C_{i'k}-p_{ij} \geq -Mr_{iji'k}; & \text{if } a_{jk}=1,\ (i\neq i'), i,i'=1,\ldots,m,, & j=1,\ldots,n, & (7)\\
\quad x_{ijk}, y_{ii'j} \in \{0,1\}; & 1\leq i'<i\leq m, & 1\leq j<k\leq n, & (8)\\
\quad r_{iji'k} \in \{0,1\}; & (i\neq i'), i,i'=1,\ldots,m, & 1\leq j<k\leq n, & (9)\\
\quad C_{ij} \geq p_{ij}; & i=1,\ldots,m, & j=1,\ldots,n. & (10)
\end{array}\right.$$

(1) equates the makespan to the maximum of the completion times of all operations.
(2) and (3) ensure that job J_k either precedes job J_j or follows it on M_i, but not both.
(4) and (5) ensure that the operations of the same job cannot be processed at the same time on different machines.
(6) and (7) ensure that two conflicting jobs cannot be processed simultaneously on different machines.

The conflict constraints between the jobs and the conflicts between the operations of the same job can be modelled without introducing the variables $y_{ii'j}$ and $r_{iji'k}$ as follows.

$$(P_2)\left|\begin{array}{llll}
\min C_{max} & & & \\
\text{S.C } C_{max} \geq C_{ij}; & i=1,\ldots,m; & j=1,\ldots,n, & (1)\\
\quad C_{ij}-M(1-x_{ikj}) \leq C_{ik}-p_{ik}; & i=1,\ldots,m, & 1\leq j<k\leq n, & (2)\\
\quad C_{ij}-C_{ik}-p_{ij} \geq -Mx_{ikj}; & i=1,\ldots,m, & 1\leq j<k\leq n, & (3)\\
\quad \frac{\max\{c_{ij}-c_{i'j};0\}}{p_{ij}}+\frac{\max\{c_{i'j}-c_{ij};0\}}{p_{i'j}} \geq 1; & 1\leq i'<i\leq m, & j=1,\ldots,n, & (11)\\
\quad \frac{\max\{c_{ij}-c_{i'k};0\}}{p_{ij}}+\frac{\max\{c_{i'k}-c_{ij};0\}}{p_{i'k}} \geq 1; & \text{if } a_{jk}=1,\ (i\neq i'), i,i'=1,\ldots,m, & 1\leq j<k\leq n, & (12)\\
\quad x_{ijk} \in \{0,1\}; & i=1,\ldots,m & 1\leq j<k\leq n, & (13)\\
\quad C_{ij} \geq p_{ij}; & i=1,\ldots,m, & j=1,\ldots,n. & (10)
\end{array}\right.$$

We replaced constraints (4) and (5) of (P_1) by constraint (11) of (P_2) and (6) and (7) of (P_1) by the constraint (12) of (P_2).

The disjoint constraints of (P_1) and (P_2) can be written in two different ways:

– Combine each pair of inequality dichotomous constraints into a single equality constraint that we set equal to a surplus variable as follows:

$$(2)+(3) \Rightarrow \begin{cases} C_{ij}-C_{ik}+Mx_{ikj}-p_{ij}=X_{ijk}; & i=1,\ldots,m, 1\leq j<k\leq n\\ M-p_{ik}-p_{ij}\geq X_{ijk}; & i=1,\ldots,m, 1\leq j<k\leq n\\ X_{ijk}\geq 0; & i=1,\ldots,m, 1\leq j<k\leq n \end{cases}$$

$$(4)+(5) \Rightarrow \begin{cases} C_{ij}-C_{i'j}+My_{ii'j}-p_{ij}=Y_{ii'k}; & 1\leq i'<i\leq m, j=1,\ldots,n\\ M-p_{i'j}-p_{ij}\geq Y_{ii'k}; & 1\leq i'<i\leq m, j=1,\ldots,n\\ Y_{ii'k}\geq 0; & 1\leq i'<i\leq m, j=1,\ldots,n \end{cases}$$

$$(6)+(7) \Rightarrow \begin{cases} C_{ij}-C_{i'k}+Mr_{iji'k}-p_{ij}=R_{iji'k}; & \text{if } a_{jk}=1, (i\neq i'), i,i'=1,\ldots,m, 1\leq j<k\leq n\\ M-p_{i'k}-p_{ij}\geq R_{iji'k}; & \text{if } a_{jk}=1, (i\neq i'), i,i'=1,\ldots,m, 1\leq j<k\leq n\\ R_{iji'k}\geq 0; & \text{if } a_{jk}=1, (i\neq i'), i,i'=1,\ldots,m, 1\leq j<k\leq n \end{cases}$$

By replacing these constraints in (P_1) and (P_2), we obtain the models (P_3) and (P_4) respectively.

– Keep the first inequality and add the fact that the sum of the two variables equals 1.

$$(2)+(3) \Rightarrow \begin{cases} C_{ij} - M(1-x_{ikj}) \leq C_{ik} - p_{ik}; & i=1,\ldots,m, 1 \leq j \neq k \leq n \\ x_{ijk} + x_{ikj} = 1; & i=1,\ldots,m, 1 \leq j \neq k \leq n \end{cases}$$

$$(4)+(5) \Rightarrow \begin{cases} C_{ij} - M(1-y_{ii'j}) \leq C_{i'j} - p_{i'j}; & 1 \leq i' \neq i \leq m, j=1,\ldots,n \\ y_{ii'j} + y_{i'ij} = 1; & 1 \leq i' \neq i \leq m, j=1,\ldots,n \end{cases}$$

$$(6)+(7) \Rightarrow \begin{cases} C_{ij} - M(1-r_{iji'k}) \leq C_{i'k} - p_{i'k}; & 1 \leq i' \neq i \leq m, 1 \leq j \neq k \leq n \\ r_{iji'k} + r_{i'kij} = 1; & 1 \leq i' \neq i \leq m, 1 \leq j \neq k \leq n \end{cases}$$

$$(8)+(9) \Rightarrow \left\{ x_{ijk}, y_{ii'j}, r_{iji'k} \in \{0,1\}; \; 1 \leq i' \neq i \leq m, 1 \leq j \neq k \leq n \right.$$

$$(13) \Rightarrow \left\{ x_{ijk} \in \{0,1\}; \; 1 \leq i' \neq i \leq m, 1 \leq j \neq k \leq n \right.$$

By replacing these constraints in (P_1) and (P_2), we obtain the models (P_5) and (P_6) respectively.

5 Computational experiments

The runs of the above mathematical models were made with Microsoft Visual Studio 2017 (using C++ language) and the models were solved using Cplex 12.8 solver. All experiments were carried out on randomly generated instances. The conflict graph of each instance is generated using the $G(n,p)$ Erdős Rényi method [2], where p is the probability that an edge exists between two vertices. The number of jobs we considered is from 3 to 11 and the number of machines $m \in \{2,3,5\}$. We also considered three values of p, $p \in \{0.2, 0.5, 0.8\}$. The processing times of the jobs were randomly generated from a uniform distribution in the interval $[0,100]$. Note that M in the experiments is set $M = \sum_{i=1}^{m} \sum_{j=i}^{n} p_{ij}$.

We observed from the implementation that the MILP models based on (P_1) ((P_1), (P_3) and (P_5)) require less CPU time than the models based on (P_2) ((P_2), (P_4) and (P_6)). Regarding the modeling of the disjoint constraints, we observed that the first and third types of constraints perform better than combining the inequalities into an equality constraint that we set equal to a surplus variable.

6 Perspectives

For future perspectives, more research is needed concerning the complexity of the OSA problem when restricted to other particular graphs. Also, more research and numerical simulations are needed to enhance the mathematical models, e.g. by adding valid cuts and by introducing procedures to improve the value of big M.

Bibliography

[1] J. Blazewicz, W. Cellary, R. Slowinski, and J. Weglarz. Scheduling under resource constraints-deterministic models. *Annals of Operations Research*, 7, 1986.

[2] P. Erdős and A. Rényi. On random graphs 1. *Publicationes Mathematicae*, 6:290–297, 1959.

[3] M. R. Garey and D. S. Johnson. *Computers and intractability: A Guide to the Theory of NP-Completeness.* New York: Freeman, 1979.

[4] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5:287–326, 1979.

[5] N. Tellache, M. Boudhar, and F. Yalaoui. Two-machine open shop problem with agreement graph. *Theoretical Computer Science*, 796:154–168, 2019.

[6] N. E. H. Tellache and M. Boudhar. Open shop scheduling problems with conflict graphs. *Discrete Applied Mathematics*, 227:103 – 120, 2017.

Scheduling loads injection during flows merging in a collector

B. Vacher[1,2], A. Jouglet[1], D. Nace[1], S. Pietrowicz[2] and M. Bouznif[2]

[1] Sorbonne Universités, Université de Technologie de Compiègne, CNRS, Heudiasyc UMR 7253, CS 60319, Compiègne cedex 60203, France
`blandine.vacher,antoine.jouglet,dritan.nace@hds.utc.fr`
[2] SAVOYE, Dijon 21000, France
`blandine.vacher,stephane.pietrowicz@savoye.com; marwane.bouznif@a-sis.com`

Keywords: scheduling, job shop, logistics.

1 Industrial context

These last years, the world of supply chain has been significantly impacted by the consumer society and by the extensive use of new technologies. The constant increase of orders to deal with is forcing logistics actors to be more reactive and competitive. SAVOYE is a company specialized in the automation of logistics warehouses, and is a manufacturer of equipments for order preparation. In this context, the company focus on optimizing its solutions. To complete an order, many operations are required (such as erecting a box, moving stored items on bins, weighting a box). Optimizing loads (boxes, bins, containers, etc.) travel time could significantly improve the warehouse performance.

In this paper, we consider the optimization of the injection of loads coming from different sources onto a single collector in aim to maximize the throughput and ensure the highest production rate possible. We define a flow as an ordered list of loads following the same path. We are interested in merging several flows into a single one. The system is made of several aisles materialized by conveyors arranged side by side which join the same collector. Each flow is carried by a conveyor and its loads are waiting in line to be injected onto the collector (see Figure 1).

The goal is to maximize the throughput of the final flow carried by the collector, conveying loads of the different flows coming from different aisles. To do that, we should be able to choose, for each aisle, the date when each load has to be injected onto the collector. In practice, a load can be injected at a given date t from a given conveyor i if there is no load in the junction between the conveyor i and the collector (see Figure 1) at time t. While the conveyor from the firt aisle can freely inject a load (since there is always no load in front of its exit), it is not the case at any time t for the conveyors downstream for which loads injected from conveyors upstream may occupy the place and do not allow these conveyors to inject loads at this precise time t.

Maximizing the throughput of the collector is equivalent to reducing the number of empty space on the collector while it runs at its highest mechanical speed capacity. Thus, we propose an approach computing the optimal dates of loads injection, based on which is built a final flow with as less as possible empty spaces. These injection dates are defined with respect to a given feasible final sequence in which the loads are waiting at the exit of the collector.

2 Problem definition

Let L be the set of the n loads to be injected onto the collector, each of them being identified by a unique identifier which corresponds to its position in the wished exit sequence

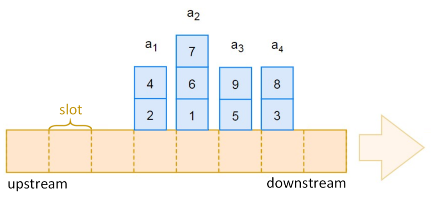


Fig. 1. Exemple of the studied systeme : instance A

σ. Let $A = \{a_1, \ldots, a_k\}$ be the set of the k aisles numbered from the most upstream to the most downstream. We assume that the system has full knowledge of the loads present in each aisles. Then, we denote by h_i the number of loads waiting in the aisle a_i. Next, for the purposes of notation, we define function $a_i : \mathbb{N} \to \mathbb{N}$ such that $a_i(j) = l \in L$ where l is the identifier of j^{th} load waiting in aisle a_i, i.e. its position in sequence σ. One can remark that we have $L = \{a_i(j), \forall i = 1 \ldots k, \forall j = 1 \ldots h_i\}$.

An instance is shown in Figure 1 with $k = 4$, $n = 9$, $A = \{a_1, a_2, a_3, a_4\}$, $L = \{1, 2, \ldots, 9\}$, $\sigma = (1, 2, 3, ..., 9)$. On this instance we notice that $h_1 = 2$ and $a_2(1) = 1$.

Finally, the collector can be divided into slots corresponding to the space occupied by a load (often larger than the physical space used, in order to take into account a safety space). To simplify, we will consider in this paper that aisles are equally distributed on consecutive slots along the collector, as shown in Figure 1. Moreover, the duration needed to run through the distance of one slot is taken as unit of time.

3 Optimally injecting loads onto the collector

We are looking to inject loads by respecting the given σ sequence so that the collector has the maximal throughput, fluid and continuous, compared to its mechanical capabilities. We recall that the maximal throughput is reached if loads can be injected onto the collector with no empty space between loads and without slowing down its speed. We will show below that this problem can be modeled as a job shop scheduling problem and can be solved using the algorithm describd below.

3.1 A job shop scheduling problem

Definition 1. *The sequence σ is said to be* feasible *if it verifies the precedence constraints induced by the mechanic configuration of the aisles: $\forall a_i \in A$, $\forall j, p \in \{1, \ldots, h_i\}$, $j < p$ we have $a_i(j) < a_i(p)$.*

For each feasible sequence σ, we are able to compute the optimal injection date for each load in such a way that there is no empty space between loads on the collector running at its maximal speed. For that, we model the problem as a *job shop* scheduling problem as follows.

There are n jobs $\{J_u, u \in L\}$ being associated with each load $\{u \in L\}$. Each job has an ordered list of operations to follow. An operation is a task to be processed on a special machine. There are k machines $\{M_1, ..., M_k\}$, each one being associated with an aisle. Reminder that aisles are numbered from upstream to downstream and are equally distributed on consecutive slots. Then it takes one unit of time for a load on the collector to pass from one aisle to the following one. Thus, a load injected onto the collector from aisle a_i will pass in front of each aisle a_p with $p \in \{i, i+1, \ldots, k\}$. This mechanism is represented by the fact that each job J_u associated with a load u from aisle a_i consists of $k - i + 1$ unitary operations $\{o_{u,i}, o_{u,i+1}, \ldots, o_{u,k}\}$. Operations on this ordered list have to be processed consecutively, without waiting, respectively by machines $\{M_i, M_{i+1}, \ldots, M_k\}$.

By construction, the injection date onto the collector of the load u from aisle a_i corresponds to the start-time of the first operation $o_{u,i}$ of the job J_u, while the start-times of the following operations (i.e. $\{o_{u,i+1}, \ldots, o_{u,k}\}$) represent the dates at which the same load is in front of each following aisles.

Note that each job is made of at least one operation on the last machine M_k (since each load passes in front of at least the last aisle a_k). The sequence of scheduled operations on machine M_k corresponds exactly to the order in which the associated loads will pass in front of the last aisle and their starting time to the time at which the loads pass in front of the last aisle. Therefore, if load u is before load v in the given final sequence, operation $o_{u,k}$ has to be scheduled before operation $o_{v,k}$. Moreover, enforcing the constraints that machine M_k has to process operations without idle time guarantees the fact that there is no empty space between loads on the collector, maximizing its throughput.

To conclude, maximizing the throughput of the collector is equivalent to schedule all operations on M_k without idle time in the order given by the associated loads on the final sequence.

3.2 A scheduling algorithm

Several methods to solve the Job Shop problem already exist, as in the review proposed by Jacek Blazevitcz *et. al.* (1995) and new approaches shown by J. F. Gonçalves *et. al.* (2005) or by P. Pongchairerks (2016). However, this scheduling problem can be solved by the following algorithm thanks to the specificities of our model.

To begin with, supposing the first load of the sequence comes from aisle a_i, the operations $\{o_{1,i}, o_{1,i+1}, \ldots, o_{1,k}\}$ associated with the job J_1 start at date t', $t' + 1$, …, $t' + k - i$.

Then, we schedule the operations of the job J_u corresponding to the next load u in the final sequence. This job will be scheduled first by dealing with the operation $o_{u,k}$ to be processed on machine M_k, just after (without idle time) the previous operation $o_{u-1,k}$ scheduled on it. Supposing that the operation $o_{u,k}$ starts at date t, the operation $o_{u,k-1}$ (if it exists) is scheduled on machine M_{k-1} at date $t-1$, and so on, until having all operations scheduled. This procedure is iteratively applied on the $n - 1$ jobs.

Real injecting dates on the collector are deduced directly from the start time of the first operation of each associated job according to the real duration of a time slot (which depends on the speed of the collector).

Table 1. Scheduling of the instance A

Time	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
M_1	**2**		**4**								
M_2	**1**	2		4		**6**	**7**				
M_3		1	2		4	**5**	6	7		**9**	
M_4			1	2	**3**	4	5	6	7	**8**	9

The Table 1 shows the application of this algorithm on the instance given in Figure 1. The load identifier written in a box means that this load is passing in front of the aisle pointed out by the line index, on the time defined by the column index. The wished final sequence is actually scheduled without empty space on the final machine M_4 and the injecting dates are deduced from their first occurrence (bolded in the table).

3.3 A formula for the injecting dates

Let i_1 be the aisle index containing the load 1, first load of σ. Moreover, we suppose that we can inject loads from date 0.

Proposition 1. *The first operation on M_k will start at the earliest date $t_0 = \max_{a_i(1), i=1...i_1} \{k - i - a_i(1) + 1\}$. Then, the injecting date of load $u \in L$ is given by $T(u)$ such that: $\forall i = 1...k, \forall j = 1...h_i,\ T(a_i(j)) = t_0 + a_i(j) - 1 - (k - i)$.*

Proof. Thanks to the previous job shop scheduling problem, we can calculate t_0 the earliest date we can schedule the first operation on machine M_k (proof not given in this paper). given the identifier (so the wished place on σ) of the j^{th} load waiting on the aisle a_i

Reminder that $a_i(j)$ gives the position of j^{th} load on aisle a_i on the final sequence σ. We deduce that the last operation of the job associated with the load $a_i(j)$ starts at $t_0 + a_i(j) - 1$. Thus, we deduce that this load injecting date is $(k - i)$ slots of time before according to the aisles layout hypothesis.

4 Results and perspectives

We succeed in maximizing the collector throughput while running at its maximal mechanical speed capacity, totally occuping and sorting loads according to a wished final sequence. To do that, we found an algorithm which always give an uninterrupted flow whatever feasible sequence for one instant of the system. Thanks to this method, we were able to extract a formula to directly calculate the loads injection dates. Iterating the process at strategic dates, linked correctly, allows for an uninterrupted dynamic flow on the collector.

This method has been adapted for aisles dispatched randomly across the collector and we also have thought about the construction of a good final sequence (if not given). Those solutions led to a patent deposit.

References

J. Blazevitcz, W. Domschke, E. Pesch, 1995, “The job shop scheduling problem: Conventional and new solution techniques", *European Journal of Operational Research* 93 (1996) 1-33

J. F. Gonçalves, J. J. Mendes, and M. G. C. Resende, 2005, “A hybrid genetic algorithm for the job shop scheduling problem", *European Journal of Operational Research*, vol. 167, no. 1, pp. 77–95.

P. Pongchairerks, 2016, “Efficient local search algorithms for job-shop scheduling problems", *International Journal of Mathematics in Operational Research*, vol. 9, no. 2, pp. 258–277.

New benchmark datasets for the RCMPSP

Rob Van Eynde[1] and Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
rob.vaneynde@ugent.be
mario.vanhoucke@ugent.be
[2] Operations and Technology Management Centre, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, United Kingdom

Keywords: Multiproject scheduling, benchmark datasets, decoupled scheduling.

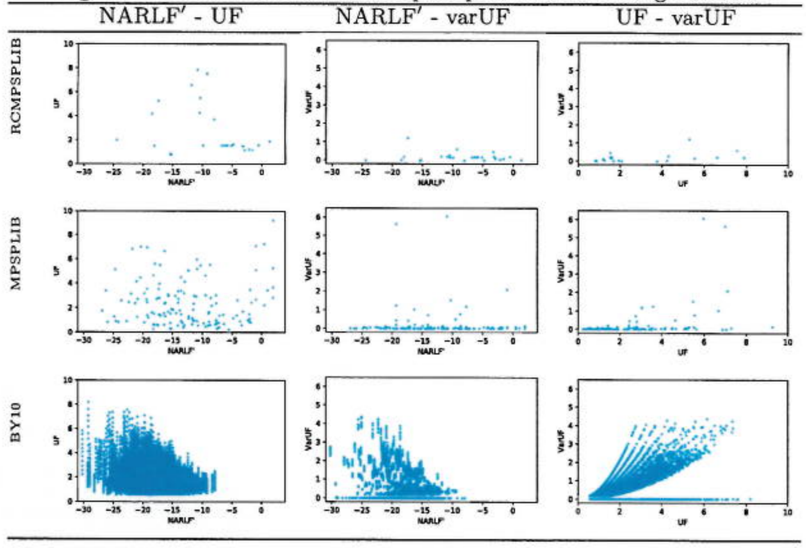
1 Introduction

In the resource-constrained project scheduling problem (RCPSP), a set of activities subject to precedence and resource constraints needs to be scheduled. Many extensions to the base problem have been developed and analysed (Hartmann and Briskorn 2010). One of the extensions is the resource-constrained multi-project scheduling problem (RCMPSP) in which a portfolio of different projects needs to be scheduled. All projects require (a subset of) the available renewable resource types, but there are no interproject precedence relations. The problem is to construct a resource feasible schedule in which a (time-related) objective is minimised. In recent years, the multi-project scheduling problem and its extensions have received more attention from researchers. However, the developed solution procedures are not always tested on the same datasets, inhibiting an unbiased comparison between the algorithms. The objectives of our research are threefold. First, we evaluate the existing datasets based on the feature space that they occupy. Second, we augment the generation procedure of Browning and Yassine (2010b) and use it to generate new datasets that cover a wider range of parameter values. Third, we set up a computational experiment with two scheduling procedures to analyse the differences between the new and existing datasets.

2 Existing datasets

In literature there are three publicly available datasets for the RCMPSP: MPSPLIB (Homberger 2007, Homberger 2012), BY10 (Browning and Yassine 2010b) and RCMPSPLIB (Vázquez *et. al.* 2015). Furthermore, there is the dataset of Wauters *et. al.* (2016) for the multi-mode extension of the RCMPSP. Because we address the basic variant, we will omit the last one from our analysis. To evaluate the datasets, we base ourselves on three resource-related measures that were designed for the multi-project context (Browning and Yassine 2010a). The first measure is the Normalised Average Resource Loading Factor (NARLF), which indicates whether the main part of the total resource demand occurs in the first or second half of the portfolio duration. We propose a small adaptation to the way the measure is calculated and name this alternative NARLF′. Second, the Modified Average Utilisation Factor (MAUF) compares the total resource demand to the total resource supply in the critical path schedule and takes the maximum ratio over all resource types. A higher MAUF means that a resource type is more constrained. The third measure (σ^2_{MAUF}) calculates the variance of the MAUF values of the different resource types from the maximum value and exhibits how equally constrained the different resource types are. Figure 1 shows the two-dimensional feature space plots for all instances of the three existing datasets.

Fig. 1. The 2-dimensional feature space plots of the existing datasets

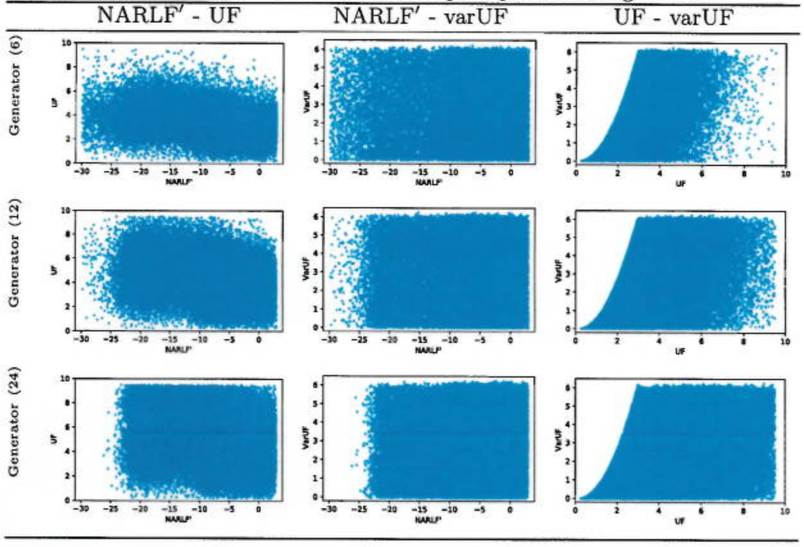


These plots show us that most of the instances in RCMPSPLIB and MPSPLIB have a low variance of resource utilisation. Furthermore, we see that BY10 contains a lot of instances but most of them have a NARLF′ lower than -10. Furthermore, the analysis of the network topology shows that most of the networks of the constituent projects are rather parallel. Based on these insights we adapt the generation procedure of Browning and Yassine (2010b) such that it is able to generate instances over a wider range of parameter combinations. The major adaptation is that RanGen2 (Vanhoucke *et. al.* 2008) is used to generate single-project networks, as it is capable of generating instances over the whole spectrum from parallel to serial networks. In order to compare the procedure with the existing sets, we generated instances of 6, 12 and 24 projects over the whole range of parameter combinations. The plots in Table 2 show that the generator is able to obtain instances over a broader feature space than the existing datasets. Although the resulting datasets cover a wider range, we observe dependencies between NARLF′ and the network structure of the single-projects. As a consequence, the feasible range of NARLF′ values for an instance depends on the network structure of its constituent projects. We propose the new datasets 6_60, 12_60 and 24_60, where each instance consists of 6, 12 and 24 projects respectively and each project contains 60 activities.

3 Evaluation using heuristics

In a second study we analyse the differences between the datasets in a computational experiment using decoupled schedule generation schemes (SGS) and a genetic algorithm (GA). Decoupled SGS's are similar to the priority rule based scheduling schemes from single-project literature. The difference lies in the fact that a single-project SGS selects one activity from the eligible set at each decision moment, while a decoupled SGS first selects a project and then selects an eligible activity from that project. In our computational experiment we tested all combinations of 16 project priority rules and 16 activity priority rules. Based on the results we come to the following conclusions. First, decoupled SGS's outperform single-project SGS's on all datasets when the objective is the minimisation of

Fig. 2. The 2-dimensional feature space plots of the generator



average project delay. Second, the choice of project priority rule has a larger impact on the performance than the choice of activity priority rule. Third, project rules related to the total resource demand perform better on the existing datasets, while critical path based rules perform better on the datasets that we propose. This indicates that the new datasets cover a part of the feature space that is not present in the existing datasets.

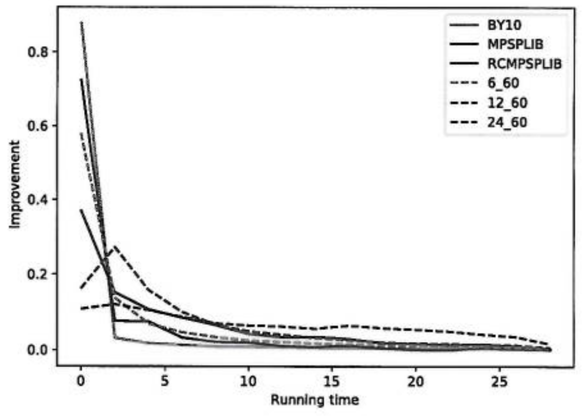
As solution procedures for the RCMPSP have advanced beyond priority rules, we execute a second computational experiment with a genetic algorithm. We implement four crossover operators and several mutation operators of Asta *et. al.* (2016). We let the GA solve each instance with a time limit of 30 seconds. Figure 3 shows the improvement found over time, relative to the total improvement after 30 seconds. The graph shows that on average, the GA quickly finds the largest improvement in the first few seconds for the datasets BY10, RCMPSPLIB and 6_60. The other datasets prove to be more challenging for the metaheuristic, as it keeps finding sizeable improvements for a longer time. This effect is the strongest for the dataset 24_60, where the figure suggests that the algorithm will still find relatively large improvements after 30 seconds. This experiment shows that MPSPLIB, 12_60 and 24_60 are more challenging for more advanced heuristics.

4 Conclusions

We propose new datasets for the RCMPSP that complement the existing sets in two ways. First, the new datasets contain a wider variety of parameter combinations, leading to different conclusions about which priority rules perform best. Second, the sets with 12 and 24 projects per instance (in addition to the set MPSPLIB) proved to be more challenging than the others for the genetic algorithm.

Furthermore, the experiments show that for the RCMPSP, decoupled schedule generations schemes outperform the traditional single-project scheduling schemes and that project prioritisation has the largest impact on performnce.

Fig. 3. Improvement profile of GA



An opportunity for future research is the design of new summary measures for the RCMPSP that are less dependent upon each other, which would facilitate generation of instances over the complete feature space.

Acknowledgements

We acknowledge the support provided by the Special Research Fund (BOF grant no. DOC014-18 Van Eynde) and the National Bank of Belgium for providing the first author with a pre-doctoral fellowship. The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, FWO and the Flemish Government department EWI.

References

Asta, S., Karapetyan, D., Kheiri, A., Özcan, E. and Parkes, A. J., 2016, "Combining monte-carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling problem", *Information Sciences*, Vol. 373, pp. 476-498.

Browning, T. R., Yassine, A. A., 2010a, "Resource-constrained multi-project scheduling: Priority rule performance revisited", *International Journal of Production Economics*, Vol. 126(2), pp. 212-228.

Browning, T. R., Yassine, A. A., 2010b, "A random generator of resource-constrained multi-project network problems", *Journal of Scheduling*, Vol. 13(2), pp. 143-161.

Hartmann, S., Dirk, B., 2010, "A survey of variants and extensions of the resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 207(1), pp. 1-14.

Homberger, J., 2007, "A multi-agent system for the decentralized resource-constrained multi-project scheduling problem", *International Transactions in Operational Research*, Vol. 14(6), pp. 565-589.

Homberger, J., 2012, "A (μ, λ)-coordination mechanism for agent-based multi-project scheduling", *OR spectrum*, Vol. 34(1), pp. 107-132.

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B. and Tavares, L. V., 2008, "An evaluation of the adequacy of project network generators with systematically sampled networks", *European Journal of Operational Research*, Vol. 187(2), pp. 511-524.

Vázquez, E. P., Calvo, M. P. and Ordóñez, P. M., 2015, "Learning process on priority rules to solve the RCMPSP", *Journal of Intelligent Manufacturing*, Vol. 26(1), pp. 123-138.

Wauters, T., Kinable, J., Smet, P., Vancroonenburg, W., Berghe, G. V. and Verstichel, J., 2016, "The multi-mode resource-constrained multi-project scheduling problem", *Journal of Scheduling*, Vol. 19(3), pp. 271-283.

Minimizing Delays in Aircraft-Landing Scheduling

Marie-Sklaerder Vié[1] Nicolas Zufferey[1] Roel Leus[2]

[1] GSEM - University of Geneva, Switzerland
marie-sklaerder.vie@unige.ch, n.zufferey@unige.ch
[2] Faculty of Economics and Business, KU Leuven, Belgium, roel.leus@kuleuven.be

Keywords: aircraft scheduling, heuristic, delay minimization.

1 Introduction

In collaboration with EUROCONTROL (*European Organization for the Safety of Air Navigation*), the considered *Aircraft Landing Planning* (ALP) problem aims at minimizing delays (with respect to the published airline schedules) while satisfying the *separation constraint* (which imposes minimum threshold times between planes, ranging from 90 to 240 seconds). In this study, the landing sequence of the planes has to be determined first, and subsequently their associated landing times and *Holding-Stack Patterns* (HSPs) needed to meet such landing times. HSPs consist of making a plane wait for its planned landing time by making circular patterns close to the airport. The uncertainty due to winds is taken into account in the simulation procedure (it has an impact on the arrival times).

Different pointers on ALP can be found in the literature (Avella *et al.* 2017, Bennell *et al.* 2017, Furini *et al.* 2015, Vié *et al.* 2018). The proposed solution method is a descent local search with restarts. It is quick enough with respect to implementation in real situations as it can be applied within seconds. Furthermore, the obtained results show that the delays can be reduced by approximately 50% on average when compared to a common practice rule. The paper is organized as follows. The problem is formally introduced in Section 2. Next, the proposed optimization method is designed in Section 3. Results are given in Section 4, followed by conclusions in Section 5.

2 Problem Formulation

Each instance covers a 3-hour planning horizon, which allows capturing the peak period of most airports. A rolling planning window $H^t = [t, t+w[$ (with $w = 45$ minutes) is associated with the current time t, with time steps of $\Delta t = 30$ seconds. At each time t, we only consider the flights that are in cruise and have their landing planned in H^t. The flights that have their landing in H^t but are not yet in cruise are only considered when they take-off. They are called the *pop-up* flights.

Each flight has different stages: (1) take-off, (2) cruise, (3) approach, (4) landing (the last $L = 15$ minutes, during which no modification is performed). In this paper, we only consider stages (2) and (3). From a practical standpoint, an initial schedule is first built when each flight enters the planning window (i.e., when it has taken off in the case of a pop-up flight, or when its expected landing time is within the next 45 minutes). Next, we can reschedule it (within the landing sequence) or make it wait to meet its planned arrival time (through HSPs). The popular *First-Come-First-Served* (FCFS) rule is employed to build the initial schedule. FCFS ranks the flights according to their entry times in H^t (i.e., with respect to increasing published arrival times). FCFS used to be the most employed current-practice approach (Erzberger 1995), and it is an optimal rule for the single-machine job-scheduling problems when the maximum tardiness has to be minimized (Pinedo 2016) (in our case we have to minimize the average tardiness).

We propose the following mathematical model (P^t) for each time t. Among the flights that have already taken off, we only consider the flights with planned landing times up to time $t+w$. Let J^t be the set of (say n) flights considered in H^t. For each flight $j \in J^t$, the following data is given:

- r_j: release date (i.e., take-off time).
- d_j: due date (i.e., published landing time).
- p_j^t: processing time (i.e., remaining time – in seconds – during the cruise phase).
- $s_{j,j'}$: set up time between flights j and j'. More precisely, for each pair (j, j') of flights such that j has to land before j', their landing times must be separated by $s_{j,j'} \in \{90, 120, 150, 180, 210, 240\}$ seconds, depending on the involved plane types.

We have two types of decision variables: (1) determine the vector Π^t of the positions of the flights involved at time t (i.e., improve the current landing sequence by performing an optimization method); (2) for each flight j, determine a feasible landing time C_j^t (with respect to the separation constraint) and assign a HSP of duration W_j^t in order to meet C_j^t. The objective function f to minimize is the sum of all positive delays (i.e., the total tardiness): $f = \sum_{j \in J^t} \max\{C_j^t - d_j, 0\}$. Constraints (1) impose that two flights are not scheduled in the same position. Constraints (2) capture the separation constraints. Constraints (3) determine the expected landing times. Constraints (4) are the domain constraints.

$$\Pi_j^t \neq \Pi_{j'}^t \qquad \forall j, j' \in J^t \tag{1}$$

$$C_{j'}^t \geq C_j^t + s_{j,j'} \qquad \forall j, j' \in J^t \text{ such that } \Pi_j^t + 1 = \Pi_{j'}^t \tag{2}$$

$$C_j^t = t + p_j^t + W_j^t + L \qquad \forall j \in J^t \tag{3}$$

$$\Pi_j^t \in \{1, \ldots, n\}, C_j^t \geq 0, W_j^t \geq 0 \qquad \forall j \in J^t \tag{4}$$

This problem can be seen as a variant of a single-machine total-tardiness problem with setup times, which is NP-hard even without setup times (Du and Leung 1990).

3 Optimization Method

Algorithm 1 presents how to roll the planning window H^t over the full 3-hour planning horizon. In Step 2, the landing positions Π^t of the new flights are computed with the following insertion rules used in practice: (1) each pop-up flight j that just entered H^t (i.e., $t \geq r_j$ but $t - \Delta t < r_j$, and $t \geq d_j - w$) is added to the landing sequence at a position Π^t such that its due date is respected (i.e., j is placed before all flights j' such that $C_{j'}^t \geq d_j$ but after all the other flights); (2) each flight j that took off a while ago but just entered H^t (i.e., $t \geq r_j$ and $t \geq d_j - w$, but $t - \Delta t < d_j - w$) is put at the end of the landing sequence (FCFS rule). In Step 3, each remaining processing times p_j^t is updated while considering an uncertainty parameter u_t randomly generated following the EUROCONTROL specifications. u_t generates a deviation (e.g., due to wind) of the cruise speed of around 7% (with an average of 0%, as positive deviations are compensated by negative ones). In Step 4, and after each modification of Π^t, C^t and W^t) are updated with the following current-practice rules. First, we re-number all flights of J^t as $j_1, j_2, \ldots, j_n$ such that $\Pi_{j_1}^t < \Pi_{j_2}^t < \ldots < \Pi_{j_n}^t$. Next, for $k = 1$ to n, we perform steps (S1) and (S2).

(S1) $C_{j_k}^t = \max\{C_{j_{k-1}}^t + s_{j_{k-1}, j_k}, t + p_{j_k}^t + L\}$ (i.e., the arrival time of j_k is as close as possible to the arrival time of the previous flight j_{k-1}, or as soon as j_k can land).

(S2) $W_{j_k}^t = C_{j_k}^t - (t + p_{j_k}^t + L)$ (i.e., the flight turns over the airport if it is too early with respect to the planned landing time).

Algorithm 1 Optimization for each time step t

```
Initialization: set $t = 0$, $J^t = J^{t-\Delta t} = \emptyset$ and $\Pi^t = \Pi^{t-\Delta t} = ()$.
While (not all flights have landed), do:
    1. Update $J^t$: remove the flights that have started landing (i.e., each flight $j$ for which $t \geq C_j^t - l$),
       and add the flights that have just entered the updated planning window $H^t$ (i.e., each flight
       $j$ for which $t \geq r_j$ and $t \geq d_j - w$).
    2. Compute the positions of the new flights (i.e., the flights that are in $J^t$ but not in $J^{t-\Delta t}$) to
       obtain the vector $\Pi^t$, based on $\Pi^{t-\Delta t}$ and the insertion rules.
    3. Update the remaining cruise time for each flight $j$: set $p_j^t = p_j^{t-\Delta t} - \Delta t \cdot (1 + u_j^t)$.
    4. Update $C^t$ and $W^t$ according to the new flight positions $\Pi^t$ and the processing times $p^t$.
    5. Improve solution $(\Pi^t, C^t, W^t)$ with a solution method.
    6. Move to the next time step: set $t = t + \Delta t$ and $H^t = [t, t + w[$.
```

As (1) the considered problem is NP-hard, (2) up to 24 flights are involved in H^t, and (3) the allowed computing-time limit T is very short ($T = \Delta t = 30$ seconds), quite a number of potential solution methods are not suitable for Step 5. Indeed, exact methods, cumbersome population-based metaheuristics (e.g., genetic algorithms, ant algorithms) or metaheuristics using a somewhat long learning process (e.g., simulated annealing) are too slow. In contrast, a descent local search (DLS) appears as a promising candidate.

DLS takes as input the solution from Step 4. At each iteration, a neighbor solution S' is generated from the current solution $S = (\Pi^t, C^t, W^t)$ by performing the best *Reinsert* move on S. A move *Reinsert* consists of changing the position Π_j^t of a flight $j \in J^t$ within the landing sequence. After each modification of Π^t, the associated variables (C^t, W^t) must be updated to have a feasible solution S' (separation constraint) and to know $f(S')$. The search process stops when no improvement of S is achieved during an iteration. In order to use the full time budget T, DLS is restarted when it encounters a local minimum (it occurs almost every second). The best visited solution is returned at the end.

At each iteration, two mechanisms are used for reducing the computational effort. First, the new position for the investigated flight j must be in $[\Pi_j^t - 5; \Pi_j^t + 5]$. This kind of *Constrained Position Shifting* is standard (Balakrishnan and Chandran 2010). Indeed, from a practical standpoint, it seems straightforward to reschedule a flight not too far away from its initial position. Second, only a random proportion ρ (tuned to 50%) of the possible neighbor solutions is generated. These mechanisms allows to perform more iterations during T seconds, which increases the exploration capability of DLS.

4 Results

The algorithms were coded in C++ (under Linux, 3.4 GHz Intel Quad-core i7 processor, 8 GB of DDR3 RAM). Table 1 compares the proposed DLS approach with FCFS (i.e., a common practice rule, see Algorithm 1 without Step 5). For each instance (provided by EUROCONTROL), the following information is provided: the number N of flights, the largest number n_{max} of flights encountered in a planning window, the average delay and the maximum delay (for both DLS and FCFS). The two latter quantities are computed with respect to all flights (in seconds), and averaged over 5 runs (with different uncertainty scenarios). The percentage gains of DLS (compared to FCFS) are given in the two last columns (a negative value indicates a better performance for FCFS). We can see that DLS can significantly reduce the average delays (almost 50%). Interestingly, the improvement is somewhat increasing with the difficulty of the instance (i.e., with N and n_{max}), but further investigations are required to understand the benefit of DLS with respect to the instance characteristics. FCFS is often better regarding the maximum delay. It makes sense as FCFS guarantees optimality for minimizing the maximum delay (but not the average delay) for single-machine job-scheduling contexts. However, DLS can sometimes do better even for the maximum delay, as it reacts to uncertainties whereas FCFS does not.

Table 1. Comparison of FCFS with DLS for 15 instances provided by EUROCONTROL.

			FCFS		**DLS**		**% Gain**	
Instance	N	n_{max}	**avg. delay**	**max delay**	**avg. delay**	**max delay**	**avg. delay**	**max delay**
1	59	16	91.36	305.00	59.96	450.20	34%	-48%
2	35	10	156.08	528.20	96.98	490.40	38%	7%
3	64	20	154.41	447.60	93.05	456.00	40%	-2%
4	79	24	388.30	782.60	228.31	1672.60	41%	-114%
5	53	14	189.53	545.00	110.36	538.40	42%	1%
6	79	21	328.86	709.20	181.40	1629.20	45%	-130%
7	75	18	208.88	558.80	111.38	475.40	47%	15%
8	75	24	288.57	651.40	143.88	1480.60	50%	-127%
9	62	16	240.26	539.20	118.33	505.20	51%	6%
10	70	22	207.41	503.20	99.57	563.40	52%	-12%
11	72	22	280.79	631.20	131.57	800.20	53%	-27%
12	71	18	170.19	514.80	77.72	475.20	54%	8%
13	97	23	386.69	903.00	174.56	1247.60	55%	-38%
14	61	15	195.54	644.60	86.58	612.80	56%	5%
15	97	20	234.52	569.00	97.04	662.20	59%	-16%
Average results			**234.76**	**588.85**	**120.71**	**803.96**	**48%**	**-31%**

5 Conclusion

The *Aircraft Landing Planning* is a challenging problem as the runway capacity is the bottleneck of many airports. In collaboration with EUROCONTROL, this study proposes a quick and efficient descent-based solution method for minimizing delays. Indeed, solutions can be obtained within seconds (which is appropriate for real-world implementation) and the average delay is reduced by almost 50%. Possible future works include the development of refined algorithms and other techniques (e.g., speed adjustments, detours) to make the flights meet their landing times, in order to reduce the over-the-airport traffic.

Acknowledgements. Study partially financed by EUROCONTROL (SESAR2020 program). Many thanks to Mr. Raphaël Christien for his availability/advices.

References

Avella P., Boccia M., Mannino C., Vasilyev I., 2017, "Time-Indexed Formulations for the Runway Scheduling Problem", *Transportation Science*, Vol. 51 (4), pp. 1031-1386.

Balakrishnan H., Chandran B.G., 2010, "Algorithms for scheduling runway operations under constrained position shifting", *Operations Research*, Vol. 58 (6), pp. 1650-1665.

Bennell J.A., Mesgarpour M., Potts C.N., 2017, "Dynamic scheduling of aircraft landings", *European Journal of Operational Research*, Vol. 258 (1), pp. 315-327.

Du J., Leung J.Y.T., 1990, "Minimizing total tardiness on one machine is NP-hard", *Mathematics of Operations Research*, Vol. 15 (3), pp. 483-495.

Erzberger H., 1995, "Design Principles and Algorithms for Automated Air Traffic Management", *AGARD Lecture Series No. 200: Knowledge-Based Functions in Aerospace Systems, Madrid, Paris, and San Francisco*, Vol. 7 (2).

Furini F., Kidd M.P., Persiani C.A., Toth P., 2015, "Improved rolling horizon approaches to the aircraft sequencing problem", *Journal of Scheduling*, Vol. 18, pp. 435-447.

Pinedo M., 2016, "Scheduling: Theory, Algorithms, and Systems", Springer.

Vié M.-S., Zufferey N., Leus R., 2018, "Aircraft landing planning: past, present and future", *Proceedings of the 19th ROADEF Conference*, Lorient, France.

Evaluation of Scheduling Policies for the SRCPSP in a Dynamic Multi-Project Environment

Hendrik Weber and Rainer Kolisch

Technical University of Munich, Germany
`hendrik.weber, rainer.kolisch@tum.de`

Keywords: Stochastic scheduling, dynamic multi-project scheduling, genetic algorithm, generalized preprocessor policies.

1 Introduction

We study the dynamic stochastic resource-constrained multi-project scheduling problem where projects arrive stochastically over time. Each project has a deterministic network and deterministic resource demand of activities, however, activity durations are stochastic. Resource availabilities provided for processing all projects are deterministic. The objective is to minimize average weighted flow times. We propose a new solution approach which, for each interarrival time of projects, calculates a scheduling policy and executes the latter until a new project arrives. The generation of scheduling policies is based on the stochastic resource-constrained (single) project scheduling problem (SRCPSP). In a computational study we assess several policy approaches which have been proposed for SRCPSP. The remainder of the paper is organized as follows: In Section 2 we review the relevant literature. In Section 3 we detail our solution approach and in Section 4 we provide information about our computational study.

2 Literature Review

The relevant literature for our study stems from two main streams of research, namely the stochastic resource-constrained project scheduling problem and the dynamic stochastic multi-project scheduling problem.

A number of contributions have been presented for the stochastic resource-constrained project scheduling problem (SRCPSP). Stork (2001) develops exact solution procedures to solve the SRCPSP by employing preselective, linear preselective, activity-based and earliest start policies in a branch-and-bound framework. Another exact model is presented by Creemers (2015): Under the assumption of PH-distributed activity durations, Creemers develops a model that employs a backward stochastic dynamic-programming recursion solution procedure.

Golenko-Ginzburg and Gonik (1997) present a heuristic solution approach that solves a 0-1 integer programming model at each decision point in order to determine which activity to process next. Tsai and Gemmill (1998) employ Tabu Search as well as Simulated Annealing to schedule activities in an SRCPSP-setting. Ballestín (2007) employs activity-based priority policies in combination with sampling procedures and a genetic algorithm to generate precedence-feasible activity lists. A similar approach based on activity-based priority policies is adapted by Ballestín and Leus (2009) who present a greedy randomized adaptive search procedure (GRASP). A novel solution procedure is presented by Ashtiani *et al.* (2011) who propose a new class of preprocessor policies encompassing resource-based and earliest start policies by heuristically inserting new precedence constraints of the type finish-to-start into the project network. This concept of preprocessor policies is further developed by Rostami *et al.* (2018). By introducing additional start-to-start constraints, the authors propose a new class of so-called generalized preprocessor policies.

Only limited work is available for the dynamic stochastic multi-project scheduling problem. Adler *et al.* (1995) were the first to extend the single stochastic project scheduling problems to a dynamic multi-project setting. Employing the real-life example of a product development organization, they develop an empirically-based framework for analysing development time in such a context. Choi *et al.* (2007) model this dynamic stochastic multi-project scheduling problem as a Markov decision process and employ a Q-learning based approach to heuristically determine policies for starting activities. Melchiors and Kolisch (2009) proceed likewise, however solve the Markov decision process by value iteration. Extensive computational studies are provided by Melchiors (2015), in which the author evaluates different priority rules in the dynamic multi-project setting. Fliedner (2015) evaluates sampling procedures as well as the genetic algorithm proposed by Hartmann (1998) in an experimental setup.

Our work extends the current body of literature by examining the applicability of the most recently proposed solution procedures to the SRCPSP in the context of stochastic and dynamic multi-project scheduling.

3 Proposed Solution Procedure

In comparison to proactive and reactive scheduling, which both rely on the determination of a baseline schedule to address uncertainty, we focus our research on stochastic scheduling where no initial schedule is determined. Following this approach, scheduling decisions are made "online" and solutions take on the form of policies that only utilize a-priori knowledge of activity duration distributions as well as the information that is available at the corresponding decision point in time when an activity is completed. Regarding this non-anticipativity constraint, these scheduling policies gradually build a schedule during the project's execution as actual realizations of uncertain activity durations unfold. This is achieved via the combination of a heuristically predetermined ordered activity list and a schedule generation scheme that is applied in the dynamic multi-project stochastic setting.

The proposed solution procedure employs an activity-on-node representation of the project network: Whenever a new project enters the system, the network of the arriving project as well as the networks of the projects still in the system are combined to a super-network which consists of all project activities, which still have to be processed or are being processed. For activities currently being in process, distributions of activity durations will be updated based on the so far observed duration. Until the arrival of the next project, the super-network can be viewed as multi-project SRCPSP instances to which we can apply SRCPSP solution methods. However, as even the SRCPSP is known to be NP-hard, we focus our efforts solely on heuristic procedures and resort to a discrete-event simulation approach. With the objective of minimizing average weighted flow times of projects, we evaluate the policies against the lower bound derived by the critical path length of the deterministic equivalent of the project.

3.1 Experimental Study

We evaluate four of the SRCPSP-solution procedures outlined above, namely the regret-based biased random sampling method and the genetic algorithm of Fliedner (2015) as well as preprocessor policies proposed by Ashtiani *et al.* (2011) and generalized preprocessor policies suggested by Rostami *et al.* (2018). From a theoretic standpoint, preprocessor policies should, as a superset of the earliest start and resource-based policy classes, strictly dominate the class of resource-based policies. Analogously, the class of generalized preprocessor policies should strictly dominate all other policy classes as it entails all resource-based, earliest start and activity-based policies. In order to account for this, we restrict the allotted computation time for each policy generation by limiting the number of generated schedules. Also, in contrast to the proposed class of (generalized) preprocessor policies, we do not evaluate every possible additional predecessor constraint. To limit the computational effort, we only consider additional predecessor constraints that concern activities whose expected start or finish times lie within a small time window starting from the current decision point: As new projects arrive and the solution procedure is repeated, more informed decisions regarding the inclusion of additional predecessor constraints can then be made at a later point in the simulation.

The setup for the proposed experimental study is based on Fliedner (2015). The selection of two uniform distribution (U1, U2), two beta distributions (B1, B2) and an exponential distribution (EXP) for activity durations is in line with Ballestín and Leus (2009). We further assume known true means of the activity duration distributions.

Using the ProGen/max generator by Schwindt (1995) we generate 120 different RCPSP problem instances where each project consists of |N|= 15 non-dummy activities and |K|= 4 different renewable resources. The average number of required resources is controlled by the resource factor and set to either 0.25, 0.5, 0.75 or 1. The arrival of new projects follows a Poisson process with arrival rate λ. In order to evaluate several levels of resource scarcity, we adjust λ which then results in different average utilization levels $u \in 0.5, 0.7, 0.9$. Newly arriving projects are all of the same type.

After a steady number of projects in the system is reached, the flow times of 200 successive projects are averaged and compared to the critical path of the deterministic equivalent. We conduct a full-factorial experimental study with the parameters summarised in Table 1. Simulation results will be presented.

Table 1. Level of problem parameters

Paramter	**Value**
|K|	4
RF	0.25, 0.5, 0.75, 1
u	0.5, 0.7, 0.9
Distribution	U1, U2, B1, B2, EXP

References

Adler P., Mandelbaum A., Nguyen V. and Schwerer E., 1995, “From project to process management: An empirically-based framework for analyzing product development time", *Management Science*, Vol. 41, pp. 458-484.

Ashtiani B., Leus R. and Aryanezhad M., 2011, “New competitive results for the stochastic resource-constrained project scheduling problem: exploring the benefits of pre-processing", *Journal of Scheduling*, Vol. 14, pp. 157-171.

Ballestín F., 2007, “When it is worthwhile to work with the stochastic RCPSP?", *Journal of Scheduling*, Vol. 10, pp. 153-166.

Ballestín F., Leus R., 2009, “Resource–constrained project scheduling for timely project completion with stochastic activity durations", *Production and Operations Management*, Vol. 18, pp. 459-474.

Choi J., Realff M. and Lee J., 2007, "A Q–Learning–based method applied to stochastic resource constrained project scheduling with new project arrivals", *International Journal of Robust and Nonlinear Control: IFAC–Affiliated Journal*, Vol. 17, pp. 1214-1231.

Creemers S., 2015, "Minimizing the expected makespan of a project with stochastic activity durations under resource constraints", *Journal of Scheduling*, Vol. 18, pp. 263-273.

Fliedner T., 2015, "Considering Uncertainty in Project Management and Scheduling", PhD-Thesis. Munich.

Golenko-Ginzburg D., Gonik A., 1997, "Stochastic network project scheduling with non-consumable limited resources", *International Journal of Production Economics*, Vol. 48, pp. 29-37.

Hartmann S., 1998, "A competitive genetic algorithm for resource–constrained project scheduling", *Naval Research Logistics (NRL)*, Vol. 45, pp. 733-750.

Melchiors P., Kolisch R., 2009, "Scheduling of Multiple R&D Projects in a Dynamic and Stochastic Environment", *Operations Research Proceedings 2008* Springer. Heidelberg.

Melchiors P., 2015, "Dynamic and stochastic multi-project planning", PhD-Thesis. Springer. Heidelberg.

Möhring R., Stork F., 2000, "Linear preselective policies for stochastic project scheduling", *Mathematical Methods of Operations Research*, Vol. 52, pp. 501-515.

Rostami S., Creemers S. and Leus R., 2018, "New strategies for stochastic resource-constrained project scheduling", *Journal of Scheduling*, Vol. 21, pp. 394-365.

Schwindt C., 1995, "ProGen/max: A new problem generator for different resource-constrained project scheduling problems with minimal and maximal time lags", *Technical report. Institut für Wirtschaftstheorie und Operations Research, Universität Karlsruhe.*, Vol. 449, pp. 1-57.

Stork F., 2001, "Stochastic resource-constrained project scheduling", PhD-Thesis. Berlin.

Tsai Y., Gemmill D., 1998, "Using tabu search to schedule activities of stochastic resource-constrained projects", *European Journal of Operational Research*, Vol. 111, pp. 129-141.

Modular equipment optimization in the design of multi-product reconfigurable manufacturing systems

Abdelkrim R. Yelles-Chaouche[1,2], Evgeny Gurevsky[3],
Nadjib Brahimi[4] and Alexandre Dolgui[2]

[1] IRT Jules Verne, Bouguenais, France
[2] LS2N, IMT Atlantique, Nantes, France
[3] LS2N, Université de Nantes, France
[4] Rennes School of Business, France

Keywords: Reconfigurable manufacturing systems, modularity, ILP, multi-product.

1 Introduction

This paper deals with reconfigurable manufacturing systems (RMS), which are designed for handling multiple products. These latter are manufactured by a fixed number of machines, each has a limited number of emplacements, where modules can be plugged. A module is a physical unit able to perform sequentially a set of tasks. One of the main characteristics of RMS is achieved through the use of modules, thanks to their ability to be easily moved and removed from one machine to another (see Koren et al. (1999)). The modules are activated one by one within a machine. As a consequence, the load of a machine (which can not exceed a predefined cycle time) is calculated as the sum of all the processing times of the tasks assigned to its modules.

As concerns the products, they share the same set of tasks to be executed. This is due to the fact that they belong to the same family. Each product is associated to its precedence constraints. However, the latter and the processing time of the tasks may be different from one product to another.

In this study, an admissible *configuration* refers to the set of tasks of a particular product assigned to a number of modules, which are allocated to machine emplacements while meeting all the aforementioned constraints. In the case where several products are to be manufactured, admissible configurations need to be designed for each of them. Since the studied line is reconfigurable, it is therefore possible to switch from one product configuration to another one by adding, moving or removing the modules.

This context arises an important optimization problem, which consists in designing an admissible configuration for each product, such as the total number of different modules between all these configurations is minimized. To tackle this new problem, an integer linear programming (ILP) model is developed, which is presented in Section 2. The preliminary results are shown and analyzed in Section 3. Finally, conclusion and perspectives are addressed in Section 4.

2 Problem formulation

In this section, the ILP formulation of the studied optimization problem is given. The used notations and variables are introduced below.

Notations:

- V is the set of all tasks;
- W is the set of available machines;
- $m_{\max}$ is the maximum number of modules per machine;

- $r_{\max}$ is the maximum number of tasks per module;
- M is the set of all the modules that could be generated;
- $E = \{1, \ldots, |W| \cdot m_{\max}\}$ is the set of all module emplacements within a configuration;
- $E(k) = \{(k-1)m_{\max}+1, \ldots, km_{\max}\}$ is the set of module emplacements corresponding to the machine $k \in W$
- P is the set of products;
- C is the cycle time;
- $t_i^{(p)}$ is a processing time of the task $i \in V$ for the product $p \in P$;
- $G^{(p)} = (V, A^{(p)})$ is a directed acyclic graph representing the precedence constraints of the product $p \in P$. Here, $A^{(p)}$ is the set of arcs for $G^{(p)}$, where an arc $(i, j) \in A^{(p)}$ means that the task j has to be assigned either to the same module as the task i, or to succeeding ones.

Variables:

- x_{im} is equal to 1 if the task $i \in V$ is assigned to the module $m \in M$, 0 otherwise.
- $y_{me}^{(p)}$ is equal to 1 if the module $m \in M$ is allocated in the emplacement $e \in E$ of the configuration corresponding to the product $p \in P$, 0 otherwise.
- $z_{ime}^{(p)}$ is equal to 1 if the task $i \in V$ is assigned to the module $m \in M$, which is allocated in the emplacement $e \in E$ of the configuration corresponding to the product $p \in P$, 0 otherwise.
- s_m is equal to 1 if the module $m \in M$ is not empty, 0 otherwise.

$$\min \sum_{m \in M} s_m \tag{1}$$

$$1 \leq \sum_{m \in M} x_{im} \leq |P|, \quad \forall i \in V \tag{2}$$

$$\sum_{e \in E} y_{me}^{(p)} \leq 1, \quad \forall m \in M, \quad \forall p \in P \tag{3}$$

$$\sum_{m \in M} y_{me}^{(p)} \leq 1, \quad \forall e \in E, \quad \forall p \in P \tag{4}$$

$$\sum_{m \in M} \sum_{e \in E} z_{ime}^{(p)} = 1, \quad \forall i \in V, \quad \forall p \in P \tag{5}$$

$$x_{im} \leq s_m, \quad \forall i \in V, \quad \forall m \in M \tag{6}$$

$$x_{im} + y_{me}^{(p)} \leq z_{ime}^{(p)} + 1, \quad \forall i \in V, \quad \forall m \in M, \quad \forall e \in E, \quad \forall p \in P \tag{7}$$

$$z_{ime}^{(p)} \leq x_{im}, \quad \forall i \in V, \quad \forall m \in M, \quad \forall e \in E, \quad \forall p \in P \tag{8}$$

$$z_{ime}^{(p)} \leq y_{me}^{(p)}, \quad \forall i \in V, \quad \forall m \in M, \quad \forall e \in E, \quad \forall p \in P \tag{9}$$

$$\sum_{m \in M} \sum_{e \in E} e \cdot z_{ime}^{(p)} \leq \sum_{m \in M} \sum_{e \in E} e \cdot z_{jme}^{(p)}, \quad \forall (i, j) \in A^{(p)}, \quad \forall p \in P \tag{10}$$

$$\sum_{e \in E(k)} \sum_{m \in M} \sum_{i \in V} t_i^{(p)} \cdot z_{ime}^{(p)} \leq C, \quad \forall k \in W, \quad \forall p \in P \tag{11}$$

$$\sum_{i \in V} x_{im} \leq r_{\max}, \quad \forall m \in M \tag{12}$$

$$\sum_{i \in V} t_i^{(p)} \cdot x_{im} \leq C, \quad \forall m \in M, \quad \forall p \in P \tag{13}$$

$$z_{ime}^{(p)} = 0, \quad \forall i \in V, \quad \forall m \in M, \quad \forall e \notin \bigcup_{k \in Q_i^{(p)}} E(k), \quad \forall p \in P \tag{14}$$

$$\left\lceil \frac{|V|}{r_{\max}} \right\rceil \leq \sum_{m \in M} s_m \leq |V| \tag{15}$$

$$s_{m+1} \leq s_m, \quad \forall m \in M \setminus \{|M|\} \tag{16}$$

$$x_{im} \in \{0,1\}, \quad \forall i \in V, \quad \forall e \in E$$

$$y_{me}^{(p)} \in \{0,1\}, \quad \forall m \in M, \quad \forall e \in E, \quad \forall p \in P$$

$$z_{ime}^{(p)} \in \{0,1\}, \quad \forall i \in V, \quad \forall m \in M, \quad \forall e \in E, \quad \forall p \in P$$

$$s_m \in \{0,1\}, \quad \forall m \in M$$

Objective function (1) minimizes the total number of non-empty modules. Constraints (2) state that any task should be assigned to at least one, but at most $|P|$ modules. Constraints (3) express that any module can be assigned to at most one emplacement within a configuration. Whereas (4) state that one emplacement could be occupied by no many than one module. Constraints (5) is used so that all the required tasks are performed in each configuration. Constraints (6) state that a module is not empty if at least one task is assigned to it. Constraints (7), (8) and (9) ensure that the assignment of the task i to the module m forces the allocation of this latter to an emplacement within at least one configuration. This helps the model to consider and allocate only the modules that are not empty. The precedence constraints in each configuration are expressed by inequalities (10). Constraints (11) provide that the cycle time for each machine in any configuration is not exceeded. Similarly, constraints (13) ensure that the sum of the processing time of the tasks assigned to the module do not exceed the cycle time. The maximum number of tasks per module is checked by constraints (12). Constraints (14) induce that the task i can only be allocated to a restricted set of workstations, denoted by the interval $Q_i^{(p)}$, where

$$Q_i^{(p)} = \left[\left\lceil \frac{t_i^{(p)} + \sum_{j \in \mathcal{P}_i^{(p)}} t_j^{(p)}}{C} \right\rceil, |W| + 1 - \left\lceil \frac{t_i^{(p)} + \sum_{j \in \mathcal{S}_i^{(p)}} t_j^{(p)}}{C} \right\rceil \right].$$

Here, $\mathcal{P}_i^{(p)}$ (resp. $\mathcal{S}_i^{(p)}$) represents the set of all predecessors (resp. all successors) of the task i with respect to the precedence graph $G^{(p)}$ corresponding to the product p. Since the objective function consists at minimizing the number of modules, one can notice that the latter can not be greater than the number of tasks (in the case where each module has only one task assigned to it). This is used to improve the upper bound on the number of modules. The lower bound could also be calculated as the ratio of the number of tasks and the maximum number of tasks per module. Thus, upper and lower bounds are expressed in constraints (15). Finally, constraints (16) is used to avoid symmetric solutions, meaning that the module $m+1$ can be filled, only if the module m is already not empty.

3 Computational results

The ILP model is tested on the basis of 224 instances of $|V| = 20$ provided by Otto et al. (2013). Two products ($|P| = 2$) are considered with $r_{\max} = 2$ and $r_{\max} = 3$. Additionally, for each instance, the resolution `CPU` time is limited to 600 seconds, $C = 1000$,

$$|W| = \max_{p \in P} \left\{ \left\lceil 1.4 \cdot \frac{\sum_{i \in V} t_i^{(p)}}{C} \right\rceil \right\},$$

and

$$m_{\max} = \max_{p \in P} \max \left\{ k \mid \sum_{i=1}^{k} t_{\pi_i}^{(p)} \leq C \right\},$$

where $(\pi_1, \pi_2, \ldots, \pi_{|V|})$ is a permutation of V with respect to the non-decreasing order of their processing times corresponding to the product $p \in P$.

The ILP model is solved using CPLEX 12.9, installed on an 1.90GHz Intel(R) Core(TM) i7-8650U computer with 32 GB RAM. The results are expressed in Table 1, where the first column represents the number of products. The second column displays the value of $r_{\max}$. The third column presents the total number of instances. The number of instances solved to optimality as well as their average CPU time are shown in the fourth and last columns, respectively. While the instances for which no optimal solution was found are expressed by their average GAP in the fifth column.

Table 1. Summary of computational results for $|P| = 2$.

$\|P\|$	$r_{\max}$	#INST	#OPT	Avg. GAP, (%)	Avg. CPU, (s.)
2	2	224	135	17.76	226.29
	3	224	129	21.88	236.80

We can clearly analyze from Table 1 that, for $|P| = 2$, 60% of the instances were optimally solved regarding $r_{\max} = 2$, versus 58% concerning the case where $r_{\max} = 3$. The instances, which were not optimally solved (89 and 95 instances for $r_{\max} = 2$ and $r_{\max} = 3$, respectively) within the maximum CPU time, provide a relatively low average GAP. This is due to constraints (15), which significantly reduce the searching space. More detailed results for $|V| = 50$ as well as $|P| = 3$ will be provided and analyzed during the presentation on the conference.

4 Conclusion

The proposed ILP model is a first attempt to address the studied problem. The obtained results are promising, but not satisfactory. Hence, for our future research, we are looking forward to develop specific reduction rules, valid inequalities and decomposition techniques for improving the computational results.

Acknowledgements

This work was financially supported by the IRT PERFORM program, managed by IRT Jules Verne (French Institute in Research and Technology in Advanced Manufacturing).

References

Koren, Y. and Heisel, U. and Jovane, F. and Moriwaki, T. and Pritschow, G. and Ulsoy, G. and Van Brussel, H., 2013, «Reconfigurable manufacturing systems». *In: Dashchenko A.I. (Ed.) Manufacturing Technologies for Machines of the Future.* Springer, Berlin, Heidelberg, pp. 627-665.

Otto, A. and Otto, C. and Scholl, A., 2013, «Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing». *European Journal of Operational Research, 228(1): 33-45.*

Decomposition approach for fixed jobs multi-agent scheduling problem on parallel machines with renewable resources

B. Zahout, A. Soukhal and P. Martineau

Université de Tours, France
LIFAT EA 6300, CNRS, ROOT ERL CNRS 7002
boukhalfa.zahout,ameur.soukhal,patrick.martineau@univ-tours.fr

Keywords: Competing multi-agent scheduling, fixed job scheduling, resource allocation, parallel machine, MILP, decomposition approach, ε-constraint.

1 Introduction

A scheduling problem involving several actors, where each has its own decision-making autonomy, in charge of executing its subset of jobs on the same resources (the jobs are competing for the use of the same machines), can be assimilated to a multi-agent scheduling problem, where a new type of compromise must be achieved. We define the term "agent" as an entity associated with a subset of jobs. Each agent aims at minimizing his own criterion that depends only on his own jobs. These agents are in competition since they share the same resources (Agnetis, Billaut, Gawiejnowicz, Pacciarelli and Soukhal 2014) We are therefore looking for the best compromised solutions. These problems are close to the multi-objective optimization problems and cooperative game theory (Agnetis, Pascale and Pranzo 2009).

To illustrate our approaches, we focus on the case of two agents A and B. In this paper, all the developed results can be generalized to L agents. Agent A (resp. B) is associated with the set of n_A (resp. n_B) jobs, denoted by $\mathcal{N}^A = \{J_1, J_2, ..., J_{n_A}\}$ (resp. $\mathcal{N}^B = \{J_{n_A+1}, J_{n_A+2}, ..., J_n\}$), where $n = n_A + n_B$.

The n independent jobs should be scheduled without preemption on m identical parallel machines. Additional renewable resources are however necessary to process each job. Several types of such resources, denoted $R_k, k = 1 \dots K$, are needed. Hence, at execution time of job j, r_{jk} units of resource R_k are required. For each job j, the start date s_j and its finished date f_j $(j = 1, \dots, n)$ are fixed where its processing time $p_j = f_j - s_j$. w_j is the weight of job j. Dealing with each type of resources k, the machine can process more than one job at a time provided the resource consumption does not exceed a given value R_k $(k = 1 \dots K)$. We assume that the machines are continuously available during the time interval $[0, \infty)$. All data are assumed positive integers. Without lost of generality, we assume that: $s_j < f_j$ and $r_{jk} \leq R_k$ for all $j = 1, \dots, n$ and $k = 1, \dots, K$. The objective of each agent is to find a feasible solution with maximum total weighted number of scheduled jobs. Let x_{ij} be the binary decision variable where $x_{ij} = 1$ if machine i processes job J_j; 0 otherwise. We denote the maximum total weighted number of scheduled jobs of Agent A and of agent B by: $Z^A = \sum_{i=1}^{m} \sum_{j=1}^{n_A} w_j \ x_{ij}$ and $Z^B = \sum_{i=1}^{m} \sum_{j=n_A+1}^{n} w_j \ x_{ij}$, respectively.

In this study, ε-constraint approach is used to determine one Pareto optimal solution (one objective function is minimized and the other one is bounded by Q). By modifying the value Q iteratively, it is possible to obtain the whole set of strict Pareto optimal solutions.

According to the three-field notation of multiagent scheduling problems introduced in (Agnetis, Billaut, Gawiejnowicz, Pacciarelli and Soukhal 2014), the addressed problems are denoted by $Pm|CO, s_j, f_j, r_{jk}, Q_B|\varepsilon(Z^A/Z^B)$ (computation of one Pareto optimal solution) and $Pm|CO, s_j, f_j, r_{jk}|\mathcal{P}(Z^A, Z^B)$ (computation of the optimal Pareto front).

This problem is $\mathcal{NP}$-hard even if only one agent is considered (mono-criterion case) (Zahout, Soukhal and Martineau 2017).

The studied problem can be met in a Data center for example, where the goal is to optimize the objective function of each user (agent). Jobs (applications) submitted by the users should be executed on the cluster defined by m identical parallel machines. Each application is executed in one container virtualized by Docker software, for example. The machines own certain limited types of renewable resources CPU, MEMORY and STORAGE, with capacities Qu_1 of CPU, a certain quantity of memory Qu_2 and a certain storage capacity Qu_3. In this case, to execute $Application_j$, a number of virtual CPUs r_{j1}, virtual memory r_{j2} and hard drives r_{j3} are needed.

The mono-criterion case has been addressed in (Angelelli, Bianchessi and Filippi 2014) where the authors consider only one additional resource (memory) and develop exact and heuristics methods to determine one feasible solution. In the context of grid computing, (Cordeiro, Dutot, Mounié and Trystram 2011) consider the multi-agent scheduling problem with global objective function. Each agent (including the global agent who is dealing with whole set of jobs) aims to minimize his makespan. They study the organizations that share clusters to distribute peak workloads among all the participants. The authors propose a 2-approximation algorithm for finding collaborative solutions.

2 Exact methods

To compute an optimal Pareto solution, we propose an integer programming formulation. Unfortunately, the linear relaxation of such a model is rather poor. For this reason, we develop a `Dantzig-Wolfe` decomposition scheme leading to a `Branch and Price` based on `column generation` scheme. Firstly, let us introduce the definition of the Maximal Subsets.
Maximal Subsets: Let L be the subset of overlapping jobs where $\bar{L} = \{j : \bigcap_{j\in L}[s_j, f_j) \neq \emptyset\}$. L is maximal if it is not included in any other subset of overlapping jobs. Let $\mathcal{L}$ be the set of all maximal subsets, $\mathcal{L} = \{L_1, \ldots, L_h, \ldots, L_H\}$. Dealing with L_h we can choose arbitrarily a sample time t_h that belongs to the processing interval of every job in L_h, i.e $t_h \in \bigcap_{j\in L_h}[s_j, f_j)$. There is a total ordering of the maximal subsets with respect to sample times. According to this ordering, when we pass from a maximal subset to the next one, at least one job finishes its processing and at least one new job starts its processing. As a consequence, there are at most $(n = n_A + n_B)$ maximal subsets. Hence, we have: $1 \leq H \leq n$ and the maximal subsets can be efficiently detected in $O(n^2)$ by using interval graph recognition algorithm introduced in (Habib, McConnell, Paul and Viennot 2000).

2.1 Integer programming formulation

Based on maximal job subsets, we propose the following integer linear programming (MILP) where x_{ij} is a binary variable equal to 1 if machine i processes job J_j; 0 otherwise.

Constraints (2) allow job j to be assigned to at most one machine. Constraints (3) allow the assignment of at most R_k resources from machine i to the jobs. The constraint (4) express the ε-approach bounds.

MILP has mn binary variables and $n + mKH + 1$ constraints, with $1 \leq H \leq n$.
Remarks: If $H = n$ then every maximal job subset contains only one job, and the problem is trivial to solve; And if $H = 1$ then all jobs overlap and the problem reduces to a multidimensional knapsack problem MKP (Martello 1990).

$$
\begin{aligned}
\text{Maximise :}\quad & \sum_{i=1}^{m}\sum_{j=1}^{n_A} w_j\ x_{ij} && && (1)\\
\text{subject to:}\quad & \sum_{i=1}^{m} x_{ij} \le 1 && j = 1,\dots,n && (2)\\
& \sum_{j\in L_h} r_{jk}x_{ij} \le R_k && i = 1,\dots,m;\quad k = 1,\dots,K;\quad h = 1,\dots,H && (3)\\
& \sum_{i=1}^{m}\sum_{j=1}^{n_B} w_j\ x_{ij} \ge Q_{\mathcal{B}} && && (4)\\
& x_{ij} \in \{0,1\} && i = 1,\dots,m,\quad j = 1,\dots,n && (5)
\end{aligned}
$$

Given Q_B, to compute a strict Pareto solution, we first solve $Pm|CO, s_j, f_j, r_{jk}, Z^B \ge Q_B|Z^A$. Let $(\hat{Z^A}, \hat{Z^B})$ be the obtained optimal solution. We then solve the inverse problem $Pm|CO, s_j, f_j, r_{jk}, Z^A \ge \hat{Z^A}|Z^B$. The computed solution is then optimal Pareto solution, denoted by $(\hat{Z^A}, \hat{Z^{B\prime}})$.

Decomposition approach: We cannot expect the linear relaxation of MILP is good, since it neglects the fact that the resource units are divided among the different machines. We then apply a `Dantzig-Wolfe` decomposition scheme to model MILP, following the *Caprara et al.*'s approach used to solve Resource Allocation Problem *RAP* (Caprara, Furini and Malaguti 2013). The master problem of the resulting column generation approach may have a huge number of columns, so a branch-and-price approach for its solution seems reasonable. These algorithms are essentially branch-and-bound algorithms where, at each node of the search tree, variables (columns) of the problem are generated by applying the column generation technique to address the linear relaxation of the problem, eventually augmented by branching constraints (see (Desaulniers, Desrosiers and Solomon 2006) for a complete survey of column generation methods).

3 Pareto set enumeration

ε-constraint approach with different values of $Q_B \in \{0,\dots,Q_B\}$ is used to generate the set of strict Pareto solutions. With each value of Q_B we compute $(\hat{Z^A}, \hat{Z^{B\prime}})$, and we add this solution to the set of strict solutions $\mathcal{S}$. We then set $Q_B = \hat{Z^{B\prime}} + 1$ and iterate. If no feasible solution is obtained then stop and $\mathcal{S}$ is the exact Pareto front.

Proposition 1. The number of strict Pareto optimal solutions is bounded by $O(W)$ where $W = max(\sum_{j\in\mathcal{N}^A} w_j; \sum_{j\in\mathcal{N}^B} w_j)$.

4 Computational experiments

We implement our algorithms in $C++$ language and experiments have been driven on a workstation with a 2.2 Ghz Intel Core i7 processor and 16 GB of memory and a time limit of 3600 seconds (1 hour). We use IBM ILOG CPLEX Optimization Studio version 12.8 to solve the MILP, Master and Pricing Model.

Algorithms under study have been carried out with 60 instances (adapted benchmark proposed of Angelelli et al. (Angelelli, Bianchessi and Filippi 2014)). Number of jobs $n \in \{100, 120, 150, 180, 200\}$, number of parallel machines $m \in \{4, 7, 10, 15\}$, without lost of generality, we normalize the capacity of each renewable resource to $R_1 = R_2 = R_3 = \{2, 4, 6\}$, for a total of $5\times4\times3 = 60$ instances. 50% of n are jobs of agent A. By choosing 50% of jobs for each agent, we are therefore interesting in solving the most difficult problems.

Fig. 1 gives the computational time in seconds: a simple call to CPlex using the MILP and the Branch&Price to compute one strict Pareto optimal solution. We can conclude that Branch&Price is very efficient. For example, with 200 jobs, 15 machines and 3 types of resources, Branch&Price needs 2 sec where MILP model needs 768 sec.

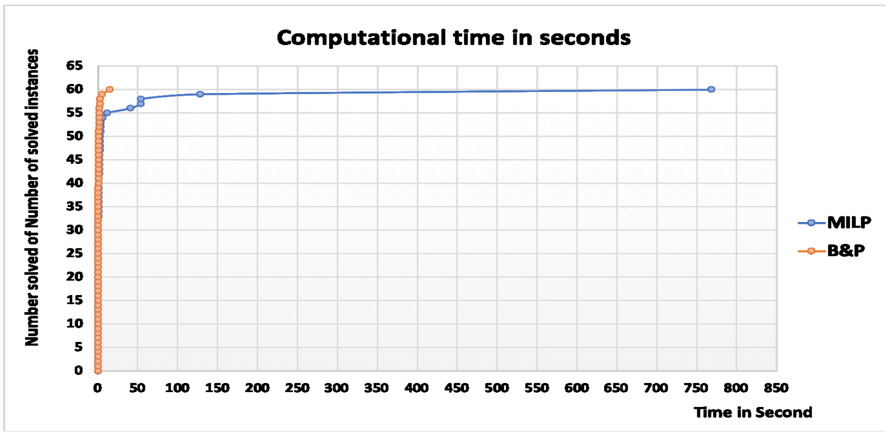


Fig. 1. Computational time in seconds

The proposed methods can be easily generated to the case of more than two agents.

This work is in progress and it will be interesting to develop lower bound to speed up the convergence of the Branch&Price. Other methods such as Meta-heuristics and heuristics providing high-quality solutions with a low computational running time to solve large size instances will be developed.

References

Martello, S., 1990, “Knapsack problems: algorithms and computer implementations", *Wiley-Interscience series in discrete mathematics and optimization.*

Habib, M., R. McConnell, C. Paul, L. Viennot, 2000, “Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing", *Theoretical Computer Science*, Vol. 234, pp. 59-84.

Desaulniers, G., J. Desrosiers, M.M. Solomon, 2006, “Column generation", *Springer Science & Business Medi*, Vol. 5.

Agnetis, A., G. de Pascale, M.Pranzo, 2009, “Computing the Nash solution for Scheduling Bargaining Problems", *International Journal of Operational Research*, Vol. 1, pp. 54-69.

Caprara, A., F. Furini, E.Malaguti, 2013, “Uncommon Dantzig-Wolfe reformulation for the temporal knapsack problem", *INFORMS Journal on Computing*, Vol. 25, pp. 560-571.

Agnetis A., J.-C. Billaut, S. Gawiejnowicz, D. Pacciarelli, A. Soukhal, 2014, “Multiagent Scheduling, Models and Algorithms", *Springer-Verlag*, Berlin Heildelberg New York.

Angelelli E., N. Bianchessi, C. Filippi, 2014, “Optimal interval scheduling with a resource constraint”, *Computers & Operations Research*, Vol. 51, pp. 268-281.

Cordeiro D., P.-F. Dutot, G. Mounié, D. Trystram, 2011, “Tight Analysis of Relaxed Multi-Organization Scheduling Algorithms", *In Proceedings of the 25th IEEE International Parallel & Distributed Processing Symposium (IPDPS), IEEE Computer Society*, Anchorage, AL, USA, pp. 1177-1186.

Zahout B., A. Soukhal, P. Martineau, 2017, “Fixed jobs scheduling on a single machine with renewable resources", *MISTA'2017*, Kuala Lumpur, Malaysia, pp. 1-9.

List of participants

- Abu-Marrul, Victor (`victor.cunha@tecgraf.puc-rio.br`)
- Agnetis, Alessandro (`agnetis@diism.unisi.it`)
- Akker, Marjan van den (`J.M.vandenAkker@uu.nl`)
- Andres, Carlos (`candres@omp.upv.es`)
- Antuori, Valentin (`vantuori@laas.fr`)
- Araujo, Thiago (`thiago.giachetto@gmail.com`)
- Arben ,Ahmeti (`e1228512@student.tuwien.ac.at`)
- Arkhipov, Dmitry (`miptrafter@gmail.com`)
- Artigues, Christian (`artigues@laas.fr`)
- Azami, Zenouzagh Hamed (`aezamihamed@gmail.com`)

- Berthier, Jeremy(`j.berthier@emse.fr`)
- Battaia, Olga (`Olga.battaia@kedgebs.com`)
- Baur, Niels-Fabian (`baur@bwl.uni-hildesheim.de`)
- Bendotti, Pascale (`pascale.bendotti@lip6.fr`)
- Benoist, Thierry (`tbenoist@localsolver.com`)
- Berg, Fleur van den (`fleurvdberg@hotmail.com`)
- Berghman, Lotte (`l.berghman@tbs-education.fr`)
- Berlinska, Joanna (`joanna.berlinska@amu.edu.pl`)
- Bianco, Lucio (`bianco@dii.uniroma2.it`)
- Bigler, Tamara (`tamara.bigler@pqm.unibe.ch`)
- Billaut, Jean-Charles (`jean-charles.billaut@univ-tours.fr`)
- Blaise, Léa (`lblaise@localsolver.com`)
- Bocquillon, Ronan (`ronan.bocquillon@univ-tours.fr`)
- Bold, Matthew (`m.bold1@lancaster.ac.uk`)

- Brahimi, Nadjib (nadjib.brahimi@mines-nantes.fr)
- Braun, Oliver (obraun@gmail.com)
- Brauner, Nadia (nadia.brauner@grenoble-inp.fr)
- Bredael, Dries (dries.bredael@ugent.be)
- Brendan, Hill (brendan.hill@gmail.com)
- Briand, Cyril (briand@laas.fr)
- Briskorn, Dirk (briskorn@uni-wuppertal.de)
- Brisoux, Devendeville Laure (laure.devendeville@u-picardie.fr)
- Brouwer, Roel (R.J.J.Brouwer@uu.nl)
- Brozova, Helena (brozova@pef.czu.cz)

- Caillard, Simon (simon.caillard@u-picardie.fr)
- Caramia, Massimiliano (caramia@dii.uniroma2.it)
- Carlier, Jacques (carlier@utc.fr)
- Charles, Mehdi (mehdi.charles@decisionbrain.com)
- Chen, Bo (b.chen@warwick.ac.uk)
- Chernykh, Ilya (idchern@math.nsc.ru)
- Chevroton, Hugo (hugo.chevroton@univ-tours.fr)
- Chrétienne, Philippe (chretiennep@gmail.com)
- Coelho, José (jcoelho@uab.pt)
- Cohen, Izack (izack.cohen@biu.ac.il)
- Creemers, Stefan (s.creemers@ieseg.fr)

- Dall'olio, Giacomo (giacomo.dallolio@tum.de)
- Dauzère-Pérès, Stéphane (dauzere-peres@emse.fr)
- Davari, Morteza (morteza.davari@skema.edu)
- Delavernhe, Florian (florian.delavernhe@u-bourgogne.fr)
- Della Croce, Federico (federico.dellacroce@polito.it)
- Demeulemeester, Erik (Erik.Demeulemeester@econ.kuleuven.be)
- Deppert, Max (made@informatik.uni-kiel.de)
- Detienne, Boris (boris.detienne@u-bordeaux.fr) Detienne Boris
- Durand, Martin (martin.durand@lip6.fr)

- Eynde, Rob van (rob.vaneynde@ugent.be)

- Fabry, Quentin (qfabry@laas.fr)
- Falq, Anne-Elisabeth (anne-elisabeth.falq@lip6.fr)
- Fejfar, Jiří (fejfar@pef.czu.cz)
- Fernandez-Viagas, Victor (vfernandezviagas@us.es)
- Fink, Andreas (andreas.fink@hsu-hamburg.de)
- Flores, Gomez Mario (mario.flores@emse.fr)
- Flynn, Mike (m.k.flynn@soton.ac.uk)
- Fondrevelle, Julien (julien.fondrevelle@insa-lyon.fr)
- Fouilhoux, Pierre (pierre.fouilhoux@lipn.fr)

- Garrigues, Mathilde (mathilde.garrigues@ariane.group)
- Gerard, Olivier (olivier.gerard@u-picardie.fr)
- Gerhards, Patrick (patrick.gerhards@hsu-hh.de)
- Ghafiki, Kaoutar (ghafiki.kaoutar@gmail.com)
- Gilenson, Miri (miray.g@gmail.com)
- Giordani, Stefano (stefano.giordani@uniroma2.it)
- Gnägi, Mario (mario.gnaegi@pqm.unibe.ch)
- Godet, Arthur (arth.godet@gmail.com)
- Goncalves, Gilles (gilles.goncalves@univ-artois.fr)
- González-Neira, Eliana María (eliana.gonzalez@javeriana.edu.co)
- Grebennik, Igor (igorgrebennik@gmail.com)
- Grigoriu, Liliana (liliana.grigoriu@upb.ro)
- Griset, Rodolphe (rodolphe.griset@edf.fr)
- Gu, Hanyu (hanyu.gu@uts.edu.au)
- Guan, Xin (xin.guan@ugent.be)
- Gul, Serhat (serhat.gul@tedu.edu.tr)
- Gurevsky, Evgeny (evgeny.gurevsky@univ-nantes.fr)

- Hadj Salem, Khadija (hadj.salem@inesctec.pt)
- Haït, Alain (alain.hait@isae.fr)
- Hanen, Claire (claire.hanen@parisnanterre.fr)

- Haroune, Meya (meya.haroune@etu.univ-tours.fr)
- Hazir, Oncu (oncu.hazir@rennes-sb.com)
- Hebrard, Emmanuel (hebrard@laas.fr)
- Herer, Yale (yale@technion.ac.il)
- Hermans, Ben (ben.hermans@kuleuven.be)
- Herroelen, Willy (willy.herroelen@kuleuven.be)
- Hill, Alessandro (ahill29@calpoly.edu)
- Hoogeveen, Evita (evitahoogeveen@gmail.com)
- Houssin, Laurent (houssin@laas.fr)
- Hryhoryeva, Maryia (Maryia.hryhoryeva@gmail.com)
- Huguet, Marie-Jose (huguet@laas.fr)

- Ikli, Sana (sana.ikli@recherche.enac.fr)

- Jami, Neil (neil.jami@itwm.fraunhofer.de)
- Jansen, Klaus (klausjansen2@gmail.com)
- Jouglet, Antoine (antoine.jouglet@hds.utc.fr)
- Jourdan, Valentine (v.jourdan@outlook.fr)
- Józefowska, Joanna (jjozefowska@cs.put.poznan.pl)
- Justkowiak, Jan-Erik (jan-erik.justkowiak@uni-siegen.de)
- Juvin Carla, (cjuvin@laas.fr)

- Kahler, Kai (Kai.Kahler@web.de)
- Kalkan, Nazlı (nazli.kalkan@tedu.edu.tr)
- Karnebogen, Mareike (mareike.karnebogen@tu-clausthal.de)
- Kedad-Sidhoum, Safia (safia.kedad_sidhoum@cnam.fr)
- Kergosien, Yannick (yannick.kergosien@univ-tours.fr)
- Khramova, Antonina (antonina.khramova@gmail.com)
- Klein, Nicklas (nicklas.klein@unibe.ch)
- Knust, Sigrid (sigrid@informatik.uni-osnabrueck.de)
- Kolisch, Rainer (Rainer.Kolisch@tum.de)
- Krivonogova, Olga (krivonogova.olga@gmail.com)
- Kubiak, Wieslaw (wkubiak@mun.ca)

- Kutlag, Berfin (berfin.kutlag@tedu.edu.tr)

- Laborie, Philippe (laborie@fr.ibm.com)
- Lahrichi, Youssef (youssef.lahrichi.contact@gmail.com)
- Lamy, Damien (damien.lamy@emse.fr)
- Lange, Julia (julia.lange@wiwi.uni-kl.de)
- Lemański, Tomasz (tomasz.p.lemanski@doctorate.put.poznan.pl)
- Leus, Roel (Roel.Leus@kuleuven.be)
- Li, Feifei (gltfeifei@buu.edu.cn)
- Lopez, Pierre (pierre.lopez@laas.fr)
- Lorenzo, Lowell (Lowell.Lorenzo@up.edu.ph)
- Lucet, Corinne (corinne.lucet@u-picardie.fr)

- Malapert, Arnaud (arnaud.malapert@unice.fr)
- Martens, Annelies (annelies.martens@ugent.be)
- Matuschke, Jannik (jannik.matuschke@kuleuven.be)
- Melchiors, Philipp (philippmelchiors@hotmail.com)
- Meloni, Carlo (carlo.meloni@poliba.it)
- Messabis, Mohamed El Habib (mohamed-el-habib.messabis@dauphine.eu)
- Meunier, Hugo (meunierhugo@hotmail.fr)
- Mischek, Florian (fmischek@dbai.tuwien.ac.at)
- Mokhtari, Nada (nmokhtar@insa-rennes.fr)
- Moos, Michael (michael.moos@itwm.fraunhofer.de)
- Müller, David (david.mueller@uni-siegen.de)
- Munier, Kordon Alix (Alix.Munier@lip6.fr)
- Musliu, Nysret (musliu@dbai.tuwien.ac.at)

- Naber, Anulark (anulark@gmail.com)
- Nattaf, Margaux (margaux.nattaf@grenoble-inp.fr)
- Nekoueian, Rojin (Rojin.nekoueian@ugent.be)

- Ozturk, Onur (oozturk@uottawa.ca)

- Page, Daniel (drpage@pagewizardgames.com)

- Paolo, Detti (`paolo.detti@unisi.it`)
- Pass-Lanneau, Adèle (`adele.pass-lanneau@polytechnique.org`)
- Penz, Louise (`louise.penz@emse.fr`)
- Perrachon, Quentin (`quentin.perrachon@univ-ubs.fr`)
- Perraudat, Antoine (`antoine.perraudat@emse.fr`)
- Pesch, Erwin (`erwin.pesch@uni-siegen.de`)
- Petra, Pavlickova (`pavlickovap@pef.czu.cz`)
- Ploton, Olivier (`olivier.ploton@univ-tours.fr`)
- Portoleau, Tom (`tom.portoleau@gmail.com`)
- Postek, Krzysztof (`k.s.postek@tudelft.nl`)
- Posthoorn, Jan (`j.i.posthoorn@uu.nl`)
- Potts, Chris (`C.N.Potts@soton.ac.uk`)
- Pranzo, Marco (`marco.pranzo@unisi.it`)
- Prunet, Thibault (`thibault.prunet@emse.fr`)

- Rault, Tifenn (`tifenn.rault@bocqfamily.fr`)
- Reinke, Max (`max.reinke@tu-clausthal.de`)
- Riviere, Louis (`louis.riviere@laas.fr`)
- Robbes, Alexis (`alexis.robbes@univ-tours.fr`)
- Rodoplu, Melek (`melek.rodoplu@emse.fr`)
- Rozycki, Rafal (`rafal.rozycki@cs.put.poznan.pl`)

- Sa, Shibasaki Rui (`ruishibasaki@gmail.com`)
- Saeedi, Pedram (`pedram.saeedi@kuleuven.be`)
- Salvatore, Alessio (`a.salvatore@iac.cnr.it`)
- Satic, Ugur (`ugursatic@gmail.com`)
- Schmidt, Günter (`gj.schmidt@yahoo.de`)
- Sedding, Helmut A. (`helmut@sedding.net`)
- Servranckx, Tom (`tom.servranckx@ugent.be`)
- Shabtay, Dvir (`dvirs@bgu.ac.il`)
- She, Bingling (`phd16bs@mail.wbs.ac.uk`)
- Shen, Liji (`liji@liji.de`)

- Shtub, Avraham (shtub@ie.technion.ac.il)
- Simonin, Gilles (gilles.simonin@imt-atlantique.fr)
- Snauwaert, Jakob (jakob.snauwaert@ugent.be)
- Song, Jie (jieson.Song@ugent.be)
- Soukhal, Ameur (ameur.soukhal@univ-tours.fr)
- Stürck, Christian (christian.stuerck@hsu-hh.de)
- Su, Bentao (subentao@nuaa.edu.cn)
- Subrt, Tomas (subrt@pef.czu.cz)
- Szwarcfiter, Claudio (claudioszw@gmail.com)

- T'kindt, Vincent (tkindt@univ-tours.fr)
- Talens, Carla (cartafa@us.es)
- Tamssaouet, Karim (karim.tamssaouet@bi.no)
- Tang Ning, (ning.tang@lip6.fr)
- Tarasov Ilia, (Ilia.TARASOV@isae-supaero.fr)
- Tellache Nour, Elhouda (nour.tellache@gmail.com)
- Thevenin, Simon (simon.thevenin@imt-atlantique.fr)
- Torba, Rahman (rahman.torba@etu.emse.fr)
- Trautmann, Norbert (norbert.trautmann@pqm.unibe.ch)
- Tremblet, David (trembletd@gmail.com)

- Ulusoy, Gündüz (gunduz@sabanciuniv.edu)
- Urgo, Marcello (marcello.urgo@polimi.it)
- Uzunoglu, Aykut (aykut.uzunoglu@wiwi.uni-augsburg.de)

- Vacher, Blandine (blvacher@gmail.com)
- Vanhoucke, Mario (mario.vanhoucke@vlerick.com)
- Vaseghi, Forough (forough.vaseghi@ugent.be)
- Velte,n Sebastian (sebastian.velten@itwm.fraunhofer.de)

- Wan, Dan (jiushiwo233@gmail.com)
- Watermeyer, Kai (kai.watermeyer@tu-clausthal.de)
- Weber, Hendrik (hendrik.weber@tum.de)

- Weber, Jens (jens.weber@ilmg.com)
- Węglarz, Jan (jan.weglarz@cs.put.poznan.pl)
- Weiß, Christian (christian.weiss@itwm.fraunhofer.de)
- Winter, Felix (winter@dbai.tuwien.ac.at)
- Wu, Qiao (qiao.wu@student.kuleuven.be)
- Wu, Fei (fwu0966@gmail.com)

- Yelles, Chaouche Abdelkrim Ramzi (karim.yelles@hotmail.com)
- You, Weibao (youweibao0528@163.com)
- Younes, Nawel (nawelyounes@gmail.com)
- Yu, Yining (yining.yu@student.kuleuven.be)
- Yugma, Claude (yugma@emse.fr)

- Zahout, Boukhalfa (boukhalfa.zahout@univ-tours.fr)
- Zimmermann, Juergen (juergen.zimmermann@tu-clausthal.de)
- Zufferey, Nicolas (n.zufferey@unige.ch)

List of sponsors

EURO WG-PMS
The EURO Working Group on Project Management and Scheduling

EURO
The Association of European Operational Research Societies

INSA
Institut national des sciences appliquées de Toulouse

LAAS-CNRS
Laboratoire d'analyse et d'architecture des systèmes

Springer
Scientific publisher

TBS
Toulouse Business School

Author Index

Abu-Marrul, Victor, 17–20
Agnetis, Alessandro, 21–24, 129–132
Akker, Marjan van den, 13
Amaro, Ricardo, 89–92
Andre, Virginie, 276–279
Angius, Alessio, 25–28
Annelies, Martens, 308–311
Antuori, Valentin, 29–32
Aouam, Tarik, 296–299
Arkhipov, Dmitry, 33–36
Artigues, Christian, 37–44, 61–64, 260–263

Battaïa, Olga, 33–36, 332–335
Baur, Niels-Fabian, 45–48
Bendotti, Pascale, 248–251
Benini, Mario, 21–24
Benoist, Thierry, 61–64
Berghman, Lotte, 129–132
Berlińska, Joanna, 49–52
Bianco, Lucio, 53–56
Bigler, Tamara, 57–60
Billaut, Jean-Charles, 268–271, 276–279
Blaise, Léa, 61–64
Bocquillon, Ronan, 268–271
Bold, Matthew, 65–68
Boudhar, Mourad, 336–339
Bouznif, Marwane, 340–343
Boyaci, Burak, 65–68
Brahimi, Nadjib, 357–360
Brauner, Nadia, 141–145
Briand, Cyril, 129–132
Brisoux Devendeville, Laure, 146–149
Brisoux, Devendeville Laure, 73–76
Brown, Lachlan, 178–181
Brozova, Helena, 69–72, 316–319

Caballero-Villalobos, Juan Pablo, 162–165
Caillard, Simon, 73–76
Caramia, Massimiliano, 53–56
Carlier, Jacques, 77–80
Chen, Bo, 300–303
Chernykh, Ilya, 81–88, 194–197
Chrétienne, Philippe, 248–251
Clautiaux, François, 211–214
Coelho, José, 89–92
Cohen, Izack, 93–95
Creemers, Stefan, 96–101

Dall’olio, Giacomo, 102–105
Dauzère-Pérès, Stéphane, 16
Delavernhe, Florian, 106–109
Della Croce, Federico, 110–117
Delorme, Xavier, 118–121
Demeulemeester, Erik, 280–282
Deppert, Max, 122–124
Deroussi, Laurent, 202–205
Detienne, Boris, 211–214
Detti, Paolo, 21–24
Dhib, Cheikh, 174–177
Dolgui, Alexandre, 252–255, 357–360
Dupas, Rémy, 186–189
Durand, Martin, 125–128

Essodaigui, Siham, 29–32
Eynde, Rob van, 344–348

Fabry, Quentin, 129–132
Falq, Anne-Elisabeth, 133–136
Fernandez-Viagas, Victor, 137–140, 328–331
Fleury, Gérard, 118–121
Fontan, Florian, 141–145
Fouilhoux, Pierre, 133–136, 248–251
Framinan, Jose M., 137–140

Gerard, Olivier, 146–149
Gerhards, Patrick, 150–153
Gilenson, Miri, 154–157
Giordani, Stefano, 53–56
Gnägi, Mario, 57–60, 158–161
Goerigk, Marc, 65–68
González-Neira, Eliana María, 162–165
Grangeon, Nathalie, 202–205
Grattz Rodríguez, Ibeth, 162–165
Grebennik, Igor, 186–189
Gribkovskaia, Irina, 17–20
Guan, Xin, 166–169
Guillaume, Romain, 260–263
Gul, Serhat, 198–201
Gurevsky, Evgeny, 252–255, 357–360

Hadj Salem, Khadija, 170–173
Hall, Nicholas, 300–303
Hamacher, Silvio, 17–20
Hamaz, Idir, 264–267
Haroune, Meya, 174–177
Hazir, Oncu, 198–201
Haït, Alain, 332–335
Hebrard, Emmanuel, 29–32
Herer, Yale T., 320–327
Hermans, Ben, 21–24
Hill, Brendan, 178–181
Houssin, Laurent, 264–267
Huguet, Marie-José, 29–32

Ikli, Sana, 182–185

Jacko, Peter, 283–286
Jansen, Klaus, 122–124
Jimenez, Jose-Fernando, 162–165
Jouglet, Antoine, 77–80, 340–343

Kalaida, Nadiia, 186–189
Kalkan, Nazli, 198–201
Kanet John Jack, 236–239
Karnebogen, Mareike, 190–193
Kedad-Sidhoum, Safia, 133–136
Kergosien, Yannick, 276–279
Khramova, Antonina, 194–197
Kirkbride, Chris, 65–68
Kirkbride, Christopher, 283–286
Kolisch, Rainer, 102–105, 236–239, 353–356
Krivonogova, Olga, 85–88
Kutlag, Berfin, 198–201

Laborie, Philippe, 14
Lacomme, Philippe, 118–121
Lahrichi, Youssef, 202–205
Lamy, Damien, 118–121, 206–210
Lazarev, Alexander, 332–335
Lefebvre, Henri, 211–214
Lemaire, Pierre, 141–145
Lemanski, Tomasz, 215–218
Leus, Roel, 349–352
Lopez, Pierre, 41–44
Lorenzo, Lowell, 219–227
Lucet, Corinne, 73–76, 146–149

Malapert, Arnaud, 228–231
Mancel, Catherine, 182–185
Manzini, Massimo, 25–28
Martens, Annelies, 232–235
Martineau, Patrick, 361–364
Martinelli Rafael, 17–20
Mecler, Davi, 17–20
Melchiors, Philipp, 236–239
Meloni, Carlo, 240–243
Mohamed Babou, Hafedh, 174–177
Molina-Pariente, Jose M., 137–140
Mongeau, Marcel, 182–185

Nace, Dritan, 340–343
Nanne, Mohamedade, 174–177
Nattaf, Margaux, 228–231
Neron, Emmanuel, 174–177
Nguyen, Alain, 29–32
Norre, Sylvie, 202–205
Novak, Ana, 178–181

Olive, Xavier, 182–185
Ozturk, Onur, 244–247

Paredes Astudillo, Yenny Alexandra, 162–165
Pascual, Fanny, 125–128
Pass-Lanneau, Adèle, 248–251
Pavlickova, Petra, 69–72
Perez-Gonzalez, Paz, 328–331
Pietrowicz, Stéphane, 340–343
Pirogov, Aleksandr, 252–255
Ploton, Olivier, 256–259
Polo Mejía, Oliver, 41–44
Portoleau, Tom, 260–263
Postek, Krzysztof, 93–95
Pranzo, Marco, 21–24, 240–243
Puerto Ordóñez, Nicolás Eduardo, 162–165

Quilliot, Alain, 37–40
Quinton, Félix, 264–267

Racheslon, Emmanuel, 182–185
Rault, Tifenn, 170–173, 268–271
Reinke, Max, 272–275
Rieck, Julia, 45–48
Robbes, Alexis, 170–173, 276–279
Rossi, André, 106–109, 252–255
Rozycki, Rafal, 215–218
Rydval, Jan, 69–72

Saeedi, Pedram, 280–282
Sahli, Abderrahim, 77–80
Salassa, Fabio, 110–113
Samà, Marcella, 240–243
Satic, Ugur, 283–286
Scholz, Adam, 178–181
Sedding, Helmut A., 287–290
Servranckx, Tom, 291–299
Sevaux, Marc, 106–109

Shabtay, Dvir, 154–157
She, Bingling, 300–303
Shtern, Shimrit, 93–95
Shtub, Avraham, 320–327
Snauwaert, Jakob, 304–307
Song, Jie, 308–311
Soukhal, Ameur, 174–177, 361–364
Stuckey, Peter, 37–40
Stürck, Christian, 312–315
Subrt, Tomas, 69–72, 316–319
Szwarcfiter, Claudio, 320–327

T'kindt, Vincent, 110–117, 256–259
Talens, Carla, 137–140, 328–331
Tarasov, Ilia, 332–335
Tellache, Nour Elhouda, 336–339
Thevenin, Simon, 206–210
Toussaint, Hélène, 37–40
Trautmann, Norbert, 57–60, 158–161

Urgo, Marcello, 25–28

Vacher Blandine, 340–343
Vanhoucke, Mario, 15, 89–92, 166–169, 232–235, 291–299, 304–311, 344–348
Vié, Marie-Sklaerder, 349–352

Węglarz, Jan, 215–218
Waligora, Grzegorz, 215–218
Weber, Hendrik, 353–356

Yalaoui, Farouk, 336–339
Yelles-Chaouche, Abdelkrim R., 357–360

Zahout, Boukhalfa, 361–364
Zimmermann, Jürgen, 190–193, 272–275
Zufferey, Nicolas, 349–352