Mixed Task Scheduling and Resource Allocation Problems

Marie-José Huguet[1,2] and Pierre Lopez[1]

[1] LAAS-CNRS, 7 av. du Colonel Roche
F-31077 Toulouse cedex 4, France
{huguet,lopez}@laas.fr
[2] Université de Pau, Dept Informatique
F-64000 Pau, France

Abstract. This paper addresses a mixed task scheduling and resource allocation problem. This mixed problem is successively treated as a temporal constrained problem on the one hand and as a time and resource constrained problem on the other hand. In the second case we propose an integrated approach for the propagation of resource allocation constraints and classical task scheduling constraints. These two view-points lead to the design of two independent propagation algorithms. Their relative pruning power is evaluated on classical scheduling instances in which resource flexibility has been introduced. The experiments show two main things: the time and resource based algorithm obviously outperforms the strictly time-based algorithm; out of all resource allocation mechanisms, the most efficient is based on an energetic reasoning.

1 Introduction

The mixed task scheduling and resource allocation (in short TSRA) problem consists of a set of jobs, a set $\mathcal{T}$ of tasks and a set $\mathcal{R}$ of renewable resources. A precedence relation, denoted by $\prec$, groups together tasks in chains, each chain corresponding to a job. Tasks are performed without interruption by only one resource of $\mathcal{R}$ and when two tasks are performed on the same resource they must be sequenced. A task $i \in \mathcal{T}$ is characterized by its starting time st_i, its finishing time ft_i, and the set of allowed resources $\mathcal{R}_i \subseteq \mathcal{R}$. For a task i, its duration depends on the resource $r \in \mathcal{R}_i$ it is assigned to: hence it is denoted by $p_{i,r}$. We denoted by $\mathcal{T}_r$ the set of tasks to be performed on resource r and by $\mathcal{P}_r$ the set of tasks that may be processed on resource r.

This paper more particularly concerns Job Shop (or Flow Shop) problems with resource allocation constraints, so-called *job shop (flow shop) scheduling with multi-purpose machines* [2, 7]. Some efficient algorithms are known for each independent problem (scheduling and allocation) but they are unable to reach optimality for the mixed problem: heuristic approaches are investigated in [1, 3, 7]. Since constraint-based approaches have been proved to be an efficient and flexible way for tackling scheduling problems, an extension taking account of allocation constraints seems a

promising way of research. To our knowledge, the only works that investigated a constraint-based approach for TSRA problems have been proposed so far in [6] and [10]. In [10], a time window is associated to each task for each possible assignment. Constraint propagation mechanisms lead to the updating of task time-windows which may suppress some possible resource assignment. A specific rule dedicated to resource allocation is proposed: resources that may be allocated to several tasks are aggregated in a cumulative resource. For instance, let i, j, k be three tasks that may be assigned on resource r_1 or r_2. In the disjunctive case, no more than two tasks can be simultaneously performed on the set of resources $\{r_1, r_2\}$. An aggregated resource of capacity 2 is associated to this set and propagation for cumulative scheduling is applied. In this paper, as in [6], resource allocation is considered as processing time constraints. Some constraint propagation mechanisms are close to those developed in [10]; however specific constraint propagation mechanisms based on energetic reasoning [8] are proposed. In relation to [6], this paper addresses new constraint propagation mechanisms for resource allocation and proposes experiments on TSRA instances.

In the following, we consider the TSRA problem from two points of view: as a Temporal Problem, and as a Time and Resource Constrained Problem. In each case, we present some constraint propagation mechanisms to handle resource allocation constraints and the integration of these mechanisms with classical constraint propagation for task scheduling. The last part concerns some experiments to compare the relative pruning power of the proposed constraint propagation mechanisms.

2 Task Scheduling and Resource Allocation as Temporal Problems

2.1 Constraint Modeling

In the problem under study we distinguish pure time constraints (limit times and precedences) and resource constraints (allocation and sharing). The resource constraints can easily be modeled as temporal constraints as well: an allocation decision is related to the processing time of the task assigned on the resource under consideration; a sharing constraint yields a necessary sequencing for all pairs assigned to a given resource.

Let $V = \{st_i, ft_i \mid i \in \mathcal{T}\}$ be the set of decision variables. The set of constraints is expressed by *potential inequalities* C_{ij} between two variables x_i and x_j: $x_j - x_i \geq b_{ij}$ ($b_{ij} \in \mathbb{R}$):

– *Limit time constraints*: the starting time (resp. finishing time) of a task i is included between an earliest time est_i (resp. eft_i) and a latest time lst_i (resp. lft_i): $(st_i - x_0 \geq est_i) \wedge (x_0 - st_i \geq -lst_i)$ where x_0 denotes the time origin variable.

– *Precedence constraints*: in the routing of a job, $i \prec j$ (i.e. task j may be performed after task i is completed) is modelled by: $(st_j - ft_i \geq 0)$.

– *Resource allocation constraints*: for instance, for a task i such that $\mathcal{R}_i = \{r_1, r_2\}$ with processing times in $[\underline{p}_{i,r_1}, \overline{p}_{i,r_1}]$ on r_1 and in $[\underline{p}_{i,r_2}, \overline{p}_{i,r_2}]$ on r_2, the duration of task i belongs to $[min(\underline{p}_{i,r_1}, \underline{p}_{i,r_2}), max(\overline{p}_{i,r_1}, \overline{p}_{i,r_2})]$: $(ft_i - st_i \geq min(\underline{p}_{i,r_1}, \underline{p}_{i,r_2}))$ $\wedge$ $(st_i - ft_i \geq -max(\overline{p}_{i,r_1}, \overline{p}_{i,r_2}))$. Moreover if $\overline{p}_{i,r_1} < \underline{p}_{i,r_2}$, values in $]\overline{p}_{i,r_1}, \underline{p}_{i,r_2}[$ are not allowed: $(st_i - ft_i \geq -\overline{p}_{i,r_1}) \vee (ft_i - st_i \geq \underline{p}_{i,r_2})$.

– *Resource sharing constraints*: each task needs one resource to be performed and each resource can process only one task a time. Each pair of tasks (i, j) assigned to a common resource r must be sequenced: $(st_j - ft_i \geq 0) \vee (st_i - ft_j \geq 0)$ $\forall i, j \in \mathcal{J}_r$.

2.2 Constraint Propagation

The previous set of constraints is divided into two sub-sets: limit times and precedences form a conjunctive set of potential inequalities (all constraints must be satisfied) while resource constraints form a set of disjunctive sets of potential inequalities (at least one constraint of each disjunctive set must be satisfied). The conjunctive set of potential inequalities may be represented by an oriented and labelled graph $G = (X, E)$ in where the set of nodes X is associated to the set of variables V, and an edge from x_i to x_j labelled by b_{ij} is in the set of edges E if the potential inequality $x_j - x_i \geq b_{ij}$ exists.

Propagation on the Conjunctive Set of Potential Inequalities

The Floyd-Warshall (FW) algorithm applied on a graph $G = (X, E)$ achieves 3-consistency by computing the longest path between each couple of nodes. It may then deduce the tightest constraints on the conjunctive set of potential inequalities. Its time complexity is in $O(|X|^3)$. Despite its polynomial complexity, the FW algorithm is too much time consuming for large size problems; it is worth applying the Bellman-Ford (BF) algorithm when searching for limit times updating only. Indeed, BF algorithm achieves 2-consistency in $O(|X|.|E|)$. However BF algorithm do not handle binary constraints that may appear in the disjunctive sets of potential inequalities, particularly task durations and precedences. Hence we provide an extension, termed BF*, that integrates this particular processing. It computes in $O(|X|^2)$ a longest paths between some pairs of variables via x_0.

Propagation on the Overall Set of Constraints
To tackle the overall set of constraints, an algorithm has been proposed in [5]. This *Temporal Constraint Propagation* (TCP) algorithm is strongly related to the *Upper-Lower Tightening* algorithm proposed in the TCSP area [11]. It integrates FW algorithm and simplification rules applied on the set of disjunctive sets of potential inequalities: a rule removes a potential inequality found inconsistent relatively to the conjunctive constraints; another rule removes a disjunctive set which contains a potential inequality already satisfied by conjunctive constraints. Removing potential inequalities may resolve some conflicts in resource sharing or suppress some inconsistent resource allocations.

TCP and TCP* are two algorithms based on FW and BF* respectively. They iterate the procedure on the conjunctive set and the simplification rules on the set of disjunctive sets until an inconsistency is detected or no more deduction can be derived. However, since resource constraints are modelled by temporal constraints, the semantics of the constraints is forgotten and the algorithm cannot consider the specificity of TSRA (see section 4).

3 Task Scheduling and Resource Allocation as Time and Resource Constrained Problems

In this part, we consider separately time and resource constraints. Our goal is to define constraint propagation mechanisms for resource allocation and analyze their interaction with the propagations obtained on time and sharing constraints. Two *Dedicated Constraint Propagation* algorithms are devised. They are based either on FW procedure (DCP) or on BF* procedure (DCP*).

3.1 Constraint propagation for task scheduling

We retain two classical rules to detect forbidden precedences between each pair of conflicting tasks. Rule FP-1 is based on temporal reasoning [4] and rule FP-2 is based on energetic reasoning [8].

– *Rule FP-1*: for $i, j \in \mathcal{J}_r$, if $lft_j - est_i < \underline{p}_{i,r} + \underline{p}_{j,r}$ then $i \not\prec j$ (i.e. $j \prec i$).

The amount $-(lft_j - est_i) + \underline{p}_{i,r} + \underline{p}_{j,r}$ corresponds to the length of the path from st_j to ft_i via ft_j, x_0 and st_i in the conjunctive set of constraints. If this path produces an inconsistency with the alternative sequence $i \prec j$ then such a sequencing is infeasible.

In the disjunctive case, the energy required by i on r over an interval Δ, termed $w_{i,r}^{\Delta}$, is given by the intersection of Δ with the processing of i. This energy is minimal when the processing of i is realized for positions that overlap Δ as less as possible.

– *Rule FP-2*: for $i, j \in \mathcal{J}_r$, if $lft_j - est_i < \underline{p}_{i,r} + \underline{p}_{j,r} + \sum_{l \in \mathcal{J}_r \setminus \{i,j\}} \underline{w}_{l,r}^{[est_i, lft_j]}$ then $i \not\prec j$ (i.e. $j \prec i$).

The eventual sequencing decisions obtained with FP-1 and FP-2 yield adjustments of the release date of i and the due date of j.

Rule FP-2 subsumes rule FP-1 (in disjunctive case); conversely, FP-2 is much more time consuming than FP-1. Thus, in DCP algorithms rule FP-1 is first applied to suppress some precedences between conflicting tasks and rule FP-2 is applied on the precedences still not proved infeasible.

3.2 Constraint propagation for resource allocation

We propose four rules, FA-1 up to FA-4, to deduce forbidden assignments.

– *Rules FA-1* and *FA-2* are respectively based on FP-1 and FP-2 to detect infeasible sequencing on a resource r for a pair of tasks (i, j) such that $i \in \mathcal{J}_r$ and $j \in \mathcal{P}_r$: if $i \not\prec j$ and $j \not\prec i$ then task j cannot be assigned to r.

As for rules FP-1 and FP-2, FA-1 is first applied to suppress some assignments and rule FA-2 only considers remaining undecided assignments.

– *Rule FA-3* suppresses assignments having an inconsistent duration relatively to the deduced duration. For a task i that may be performed on resource r with duration in $[\underline{p}_{i,r}, \overline{p}_{i,r}]$, let $[\underline{p}_i, \overline{p}_i]$ be the duration deduced by temporal constraint propagation, if $[\underline{p}_{i,r}, \overline{p}_{i,r}] \cap [\underline{p}_i, \overline{p}_i] = \varnothing$, then task i cannot be assigned to r.

– *Rule FA-4* considers all the tasks in $\mathcal{J}_r$ and one task j of $\mathcal{P}_r$, if there is an interval Δ such that $W_r^\Delta \geq \sum_{l \in \mathcal{J}_r} \underline{w}_{l,r}^\Delta$ and $W_r^\Delta < \sum_{l \in \mathcal{J}_r} \underline{w}_{l,r}^\Delta + \underline{w}_{j,r}^\Delta$ then task j cannot be assigned to r, where $W_r^\Delta = ft_\Delta - st_\Delta$ denotes the maximal available energy a disjunctive resource r can provide on a given time interval Δ. To simplify the implementation of rule FA-4, we only consider the interval Δ covering all the tasks: $\Delta = [\min_{l \in \mathcal{J}_r \cup \{j\}} (est_l), \max_{l \in \mathcal{J}_r \cup \{j\}} (lft_l)]$ on which the energy consumption of every task l is its processing time. Rule FA-4 is then: if $W_r^\Delta \geq \sum_{l \in \mathcal{J}_r} p_{l,r}$ and $W_r^\Delta < \sum_{l \in \mathcal{J}_r} p_{l,r} + p_{j,r}$ then task j cannot be assigned to r.

The implementation of rule FA-4 depends on the order in which tasks are assigned to a resource. For instance, after a task is added to a set $\mathcal{J}_r$, one may not find an inconsistency whereas it would arise before adding that task. To prevent this pitfall, we would have to consider, in the worst case, all subsets of $\mathcal{J}_r$ including at least two

tasks. In practice, tasks in the subsets have to intersect each other and to intersect with the task considered in $\mathscr{P}_r$.

In the DCP and DCP* algorithms, rules are iterated until a fix-point is reached, that is either no more deduction can be derived or an inconsistency is detected. The following order is selected in our strategy: procedure for time constraint propagation, rules dedicated to resource allocation (with FA-1 before FA-2), and rules dedicated to resource sharing constraints (with FP-1 before FP-2).

4 Experiments

Our experiments concern Job Shop problems with resource allocation constraints from [7]. Set "sdata" corresponds to classical Job Shop problems (Fisher and Thompson, and Lawrence instances); in sets "edata", "rdata" and "vdata", alternative resources may process the tasks. In "edata" instances, the average number of possible resources is close to the initial job-shop problems; this average number grows for "rdata" and "vdata" instances. We also consider Flow Shop ("FS") problems with resource allocation constraints from [9, 12].

In all these benchmarks, the task duration is fixed (for each task i on a resource r: $\underline{p}_{i,r} = \overline{p}_{i,r} = p_{i,r}$) and is independent of the resource allocated to process it (for each task i and for each resource $r \in \mathscr{R}_i$: $p_{i,r} = p_i$). Hence these instances correspond to a simplification of our TSRA problem defined in section 1. In [7, 9, 12] the aim is to determine the minimum value of the completion time of jobs (makespan minimization) denoted by C_{max}^{OPT}. Our constraint propagation mechanisms are able to classify a problem as inconsistent (but cannot prove its consistency); since they do not find an optimal solution, they are embedded in a more general procedure that aims to compute the smallest value of job due-dates such that the propagation mechanisms do not detect an inconsistency. This value, denoted by C_{max}^{DED}, is a lower bound of C_{max}^{OPT}. To study the impact of the proposed constraint propagation mechanisms, we evaluate the ratio between C_{max}^{OPT} and C_{max}^{DED}: $(C_{max}^{OPT} - C_{max}^{DED}) / C_{max}^{OPT}$. The smaller this ratio is, the more powerful the constraint propagation rules. If $C_{max}^{DED} = C_{max}^{OPT}$ our constraint propagation rules detect all inconsistencies; if $C_{max}^{DED} < C_{max}^{OPT}$ some inconsistencies still remain in the instances (the constraint propagation is not complete).

For some instances, the results in [7, 9, 12] do not reach the optimal value of makespan but a lower bound, denoted by C_{max}^{LB}. The ratio $(C_{max}^{LB} - C_{max}^{DED}) / C_{max}^{LB}$ may be negative when our constraint propagation improves the lower bound of the makespan.

In the general procedure, we test several combinations of the proposed constraint propagation rules:

– TCP and TCP*: we apply the general temporal constraint propagation rules (with either FW or BF* for the conjunctive part of the temporal constraints).

– DCP and DCP*: in these combinations, we apply either FW or BF* for time constraint propagation, all the rules for resource allocation (FA-1 to FA-4), and rules FP-1 and FP-2 for resource sharing.

– DCP with only one rule for resource allocation: that is, DCP/1 with FA-1, DCP/2 with FA-2, DCP/3 with FA-3, and DCP/4 with FA-4. One applies FW for time constraint propagation, and FP-1 and FP-2 for the propagation of resource sharing constraints.

In the following tables, we summarize the results given by these combinations. For "sdata" instances, we did not test the DCP/1 up to DCP/4 algorithms since they are pure scheduling problems. Column "Time (s)" gives the average CPU time in seconds to find a lower bound of the makespan; column "ratio1" represents the average of $(C_{max}^{OPT} - C_{max}^{DED})/C_{max}^{OPT}$ (if C_{max}^{OPT} is known); column "ratio2" gives the positive average of $(C_{max}^{LB} - C_{max}^{DED})/C_{max}^{LB}$ and the column "ratio3" provides the negative average of the same ratio. In brackets we indicate the number of instances from which we compute the corresponding ratio.

	SDATA (43 instances)		EDATA (43 instances)			
Algo.	ratio1 (%)	Time (s)	ratio1 (%)	ratio2 (%)	ratio3 (%)	Time (s)
TCP	33.00	3311.19	32.21 (25)	16.59 (17)	-2.98 (1)	2201.84
TCP*	42.01	129.49	42.57 (25)	27.42 (18)	0.00 (0)	87.41
DCP	2.38	2067.94	2.30 (25)	0.76 (13)	-2.06 (5)	1474.46
DCP*	10.21	65.02	8.04 (25)	1.44 (15)	-2.58 (3)	56.66

	RDATA (43 instances)				VDATA (43 instances)		
Algo.	ratio1 (%)	ratio2 (%)	ratio3 (%)	Time (s)	ratio1 (%)	ratio2 (%)	Time (s)
TCP	30.03 (15)	27.76 (25)	-2.01 (3)	2716.82	26.64 (26)	31.45 (17)	1729.58
TCP*	32.93 (15)	27.16 (28)	0.00 (0)	129.45	26.79 (26)	31.45 (17)	130.89
DCP	23.47 (15)	25.93 (24)	-2.53 (4)	2044.01	25.79 (26)	31.45 (17)	2078.29
DCP*	23.47 (15)	24.73 (26)	-2.68 (2)	168.02	25.80 (26)	30.18 (17)	186.97

	FS [12] (169 instances)		FS [9] (63 instances)	
Algo.	ratio1 (%)	Time (s)	ratio1 (%)	Time (s)
TCP	42.36	121.31	32.62	738.81
TCP*	52.52	1.30	38.78	17.75
DCP	3.86	124.04	6.39	565.69
DCP*	3.86	7.48	6.39	26.42

These experiments show a large impact of DCP relatively to TCP in "sdata", "edata" and Flow Shop benchmarks. In "edata" instances for "ratio1", C_{max}^{DED} deduced by DCP is 2.3% away from C_{max}^{OPT} whereas C_{max}^{DED} deduced by TCP is 32.2% away from C_{max}^{OPT}. Moreover on these benchmarks DCP is usually faster than TCP. However for "rdata" and "vdata" instances, DCP weakly improves the lower bound of makespan deduced by TCP; in fact these instances seem to be less hard than "edata" instances and constraint propagation does not deduce much information (few inconsistencies are detected).

The comparisons between TCP and TCP*, and between DCP and DCP* show a weak difference between the relative power of FW and BF*. For instances in "edata"

and "ratio1": C_{max}^{DED} obtained by TCP is 32.21% away from C_{max}^{OPT} whereas for TCP* C_{max}^{DED} is 42.57% away from C_{max}^{OPT} ; and C_{max}^{DED} deduced by DCP is 2.3% away from C_{max}^{OPT} whereas 8.04% away for DCP*. However, the CPU time consumed by BF* is much smaller than the consumption of FW. It could possibly be interesting to first apply BF* and, when there is no more deduction, apply FW.

	EDATA (43 instances)				FS [12] (169 instances)		FS [9] (63 instances)	
Algo.	ratio1 (%)	ratio2 (%)	ratio3 (%)	Time (s)	ratio1 (%)	Time (s)	ratio1 (%)	Time (s)
DCP	2.30 (25)	0.76 (13)	-2.06 (5)	1474.46	3.86	124.04	6.39	565.69
DCP/1	4.23 (25)	1.07 (16)	-2.92 (2)	1343.17	3.86	123.95	6.39	564.43
DCP/2	4.23 (25)	1.07 (16)	-2.92 (2)	1433.96	3.86	123.86	6.39	564.99
DCP/3	3.80 (25)	1.03 (17)	-2.21 (3)	1407.82	3.86	123.88	6.39	564.52
DCP/4	2.44 (25)	0.81 (14)	-2.38 (4)	1349.24	3.86	124.16	6.39	564.73

Whatever the rule used in DCP/1 to DCP/4 for the Flow Shop instances, there is no difference for C_{max}^{DED} deduced by DCP with all rules. Actually in these instances, the rules devoted to resource allocation do not run: all the deductions are made by the propagation of resource sharing constraints. Moreover in 65.08% out of the Flow Shop instances from [9] and in 82.25% out of the examples from [12], the value C_{max}^{DED} obtained by DCP is equal to the optimal makespan C_{max}^{OPT}. In "edata" instances, the relative power of sole rule FA-1 (or FA-2) is poor relatively to DCP with all rules (in these instances, there is no difference between FA-1 and FA-2 whereas FA-2 subsumes FA-1). The most efficient rule is FA-4 which deduces by itself a lower bound nearer than the lower bound deduced by DCP including all the rules.

To improve our experiments on the relative pruning power of our rules, we have now to generate our own benchmarks with unrelated (and variable) durations for unassigned tasks.

Conclusions

This paper presents a constraint-based approach for mixed task scheduling and resource allocation problem. This problem can be addressed either as a temporal constrained problem or as a time and resource constrained problem. We propose some new constraint propagation mechanisms to handle resource allocation and we integrate these mechanisms with classical constraint propagation processes for time constraints and resource sharing constraints. We present some constraint propagation mechanisms that lead to the removing of some task assignments. The implementation of such mechanisms still needs to be improved to make the results independent on the order constraints are handled. Other deduction processes have been defined to determine inconsistent allocations between pairs of tasks and propagate these new constraints. These constraint propagation rules have to be implemented to evaluate their efficiency. Moreover, as we did not find in the literature benchmarks corresponding to our TSRA hypotheses (variable durations and unrelated durations

for resource allocation) we have now to generate new instances to strictly evaluate the pruning power of our rules.

References

1. Brandimarte, P.: Routing and scheduling in a flexible Job Shop by Tabu search. Annals of Oper. Res. 41 (1993) 157-183
2. Brucker, P, Jurish, B., Krämer, A.: Complexity of scheduling problems with multipurpose machines. Annals of Oper. Res. 70 (1997) 57-73
3. Dauzère-Pérès, S., Paulli, J.: An integrated approach for modeling and solving the general multiprocessor job shop scheduling problem using tabu search. Annals of Oper. Res. 70 (1997) 281-306
4. Erschler, J., Roubellat, F., Vernhes, J-P.: Characterizing the set of feasible sequences for n jobs to be carried out on a single machine. Europ. J. of Oper. Res. 4 (1980) 189-194
5. Esquirol, P., Huguet, M-J., Lopez, P.: Modeling and managing disjunctions in scheduling problems. J. of Intelligent Manufacturing 6 (1995) 133-144
6. Huguet, M-J., Lopez, P.: An integrated constraint-based model for task scheduling and resource assignment. CP-AI-OR-99, Ferrara, Italy (1999)
7. Hurink, J., Jurisch, B., Thole, M.: Tabu search for the Job Shop scheduling problem with multipurpose machines. OR Spektrum 15 (1994) 205-215
8. Lopez, P., Esquirol, P.: Consistency enforcing in scheduling: a general formulation based on energetic reasoning. 5th Int. Workshop on Project Management and Scheduling, Poznan, Poland, pp. 155-158 (1996)
9. Néron E.: Du Flow Shop Hybride au Problème Cumulatif. Thèse de Doctorat, Université de Technologie de Compiègne (1999)
10. Nuijten, W.: Time and resource constrained scheduling: A constraint satisfaction approach. PhD dissertation, Eindhoven University (1994)
11. Schwalb, E., Dechter, R.: Processing disjunctions in temporal constraint networks. Art. Intelligence 93 (1997) 29-61
12. Vignier A.: Contribution à la résolution des problèmes d'ordonnancement de type monogamme, multimachine ("Flow Shop Hybride"). Thèse de Doctorat, Université de Tours (1997)