A constraint-based procedure for scheduling and allocation with unrelated machines

Marie-José Huguet[1,2] and Pierre Lopez[2]
[1]*INSA, Toulouse, France*
[2]*Laboratoire d'Analyse et d'Architecture des Systèmes, CNRS, Toulouse, France*
huguet@laas.fr, lopez@laas.fr

Abstract

This paper addresses the resolution of mixed Task Scheduling and Resource Allocation problems in an integrated way. Time and resource constraint propagation is specified on top of a solving procedure directed by a backtracking algorithm. To evaluate the approach, experiments are reported on randomly generated instances. The main features of these instances are general precedence constraints, unrelated and versatile machines.

1. Problem statement

Introduction. This paper addresses an integrated constraint-based approach to solve mixed Task Scheduling and Resource Allocation (in short TSRA) problems. A TSRA problem consists of a set of tasks to be performed by a set of renewable resources. Resources are disjunctive, i.e., they process only one task at a time, and are called machines. To each task is associated a set of machines able to process it knowing that every task is processed without interruption within a time window on only one machine. Moreover, certain tasks may be linked together by general precedence constraints. Finally, the task duration depends on the machine on which the task is performed, i.e., the machines are unrelated. The objective is to derive necessary conditions on task scheduling and resource allocation for the existence of a schedule for which the makespan is lower than or equal to an upper bound. According to the notation scheme given in [2], the problem is equivalent to $RMPM|prec, r_i, d_i|-$ (MPM stands for multi-purpose machines) which is NP-complete considering NP-hardness of $P|p_i = 1, outtree|L_{max}$.

Some efficient algorithms are known for each independent problem, scheduling and allocation, but they are unable to reach optimality for the mixed problem. For an integrated solving of some TSRA problems, heuristic approaches are investigated in [1, 3, 6]. Since constraint-based approaches have been proved to be an efficient and flexible way for tackling scheduling problems, an extension taking into account 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 [5, 9, 11]. In these papers task durations are resource-dependent. In [9], a time-window is associated with every task for each possible allocation. Constraint propagation leads to narrowing time-windows; this updating may suppress some alternatives for the allocation. Moreover, specific developments involve the aggregation in a cumulative resource of the set of resources that may be allocated to several tasks. This paper follows the principles presented in [5] and proposes a backtracking algorithm to solve TSRA problems in an integrated way.

Typology. In [11], TSRA problems are cast into four categories. The distinctions between categories are made according to two underlying features of the resources which are the *versatility* (i.e., does a resource may execute various kinds of tasks?) and the *similarity* (i.e., does the duration of a task depend on the resource used to achieve it?). Hybrid Flow Shop problems [8] fall into the category of *TSRA problems with nonversatile and similar (identical) resources*. Job Shop problems with alternative sets of resources [9] are included in the category of *TSRA problems with nonversatile and nonsimilar (unrelated) resources*. Multi-Purpose Machines Problems considered in [6] and Multi-Processor Job Shop problems [3] are linked to *TSRA problems with versatile and*

identical resources. Finally the most general type of problems refers to *TSRA problems with versatile and unrelated resources*; to the best of our knowledge, they are only studied in [5] for which the authors propose time and resource constraint propagation mechanisms. In the sequel, we are concerned by solving this last category TSRA problems.

Modelling. A set T of tasks has to be achieved by a set M of machines. To a task $i \in T$ are associated its start time st_i, its finish time ft_i, and the set of allowed machines $?_i \subseteq ?$; its duration, denoted by $p_{i,\mu}$, depends on the machine $\mu \in ?_i$ it is allocated to. It follows that the duration of task i belongs to the interval $[\min_{\mu \in M_i} p_{i,\mu}, \max_{\mu \in M_i} p_{i,\mu}]$. Let T_μ be the set of tasks to be performed on machine μ. For a task $i \in T_\mu$, one has $st_i \geq r_i$ and $st_i + p_{i,\mu} \leq d_i$ where r_i stands for the release date and d_i for the due date. A precedence relation between two tasks i ($i \in T_\mu$) and j is written $st_j \geq st_i + p_{i,\mu}$. Resource capacity constraints state that two tasks assigned to a common machine must be sequenced: $\forall i, j \in T_\mu$, $(st_j - ft_i \geq 0) \vee (st_i - ft_j \geq 0)$.

2. Constraint propagation

Precedences, limit times, and durations form a set of time constraints on which graph algorithms (e.g., Floyd-Warshall or Bellman-Ford procedures) can deduce the tightest adjustments of start and finish times in a polynomial complexity. For the processing of resource constraints, one way is to model these constraints like disjunctions of time constraints and to extend graph algorithms to handle them [5]. This extension consists in removing the disjunctions if an inconsistency with the initial time constraints appears. These deletions, realised following a method strongly related to the *Upper-Lower Tightening* algorithm proposed in the area of Temporal Constraint Satisfaction Problems [10], may resolve some conflicts in resource sharing or suppress some inconsistent resource allocations. The main drawback of this approach is to forget the semantics of the resource constraints and thus to do not exploit the specificity of joint scheduling and allocation. Another way is to separately consider time and resource constraints. The goal is to specify constraint propagation mechanisms for resource allocation and analyse their interaction with the propagations obtained on time constraints.

Task scheduling. We retain two classical rules to detect precedences between each pair of conflicting tasks. Rule (1) is based on temporal reasoning [4]: for $i, j \in T_\mu$, if $d_j - r_i < p_{i,\mu} + p_{j,\mu}$ then $j \prec i$. Rule (2) is based on energetic reasoning [7]. In the disjunctive case, the energy required by i on μ over an interval Δ, termed $w_{i,\mu}^{\Delta}$, is given by the intersection of Δ with the processing of i. This energy is minimal, termed $\underline{w}_{i,\mu}^{\Delta}$, when the processing of i is realised for positions that overlap Δ as less as possible. Rule (2) is expressed as follows: for $i, j \in T_\mu$, if $d_j - r_i < p_{i,\mu} + p_{j,\mu} + \sum_{l \in T_\mu \setminus \{i,j\}} \underline{w}_{l,\mu}^{[r_i, d_j]}$ then $j \prec i$. The sequencing decisions obtained with rules (1) and (2) yield adjustments of r_i and d_j and hence start again the time propagation process.

Other propagation rules may be applied in order to enforce sequencing conditions, mainly selections obtained by edge-finding involving sets of tasks. At present time this type of rules has not been yet implemented in our procedures for solving TSRA problems.

Resource allocation. Three rules to deduce forbidden assignments are proposed. Rules (3) and (4) are respectively based on rules (1) and (2) and their dual forms inverting i and j. Their principle

is as follows (let P_μ be the set of tasks that may be performed on machine μ): for a machine μ and for a pair of tasks (i, j) such that $i \in T_\mu$ and $j \in P_\mu$, if $j \prec i$ and $i \prec j$ are jointly deduced applying rule (1) or rule (2), then j cannot be assigned to μ (i.e., μ is removed from set $?_j$). Rule (5) suppresses assignments involving an inconsistent duration relatively to the current duration derived from time constraints. For a task i that may be performed on machine μ (i.e., $i \in P_\mu$) with duration $p_{i,\mu}$, let $[\underline{p}_i, \overline{p}_i]$ be the duration interval deduced by temporal constraint propagation: if $p_{i,\mu} \notin [\underline{p}_i, \overline{p}_i]$ then i cannot be assigned to μ.

Other rules based on energetic reasoning over sets of tasks or on interdependencies of resource allocation are proposed in [5]. At present time these rules are not used in this study.

The following order is selected in our strategy [5]: time constraint propagation, resource allocation, and task scheduling. In the disjunctive case, rule (2) always subsumes rule (1); conversely, rule (2) is much more time consuming than rule (1). Therefore rule (1) is first applied to suppress some precedences between conflicting tasks and rule (2) is applied on the precedences still not proved infeasible. In the same way, rule (3) is first applied to suppress some allocations and rule (4) only considers remaining undecided allocations. Rules are applied until a fix-point is reached, that is either no more deduction can be derived or an inconsistency is detected.

3. Resolution and experiments

The usual procedure of backtrack, known as chronological backtrack, consists in traversing the search tree in depth-first to try to find one solution or all the solutions. It successively instantiates the problem variables in a predefined order; this order is deduced from instantiation heuristics involving variable and/or value ordering which aim to prune the tree and accelerate the search for a solution [11]. Here constraint propagation is used on top of the resolution to limit the size of the domain of the variables and not during the tree search procedure; furthermore it provides a lower bound of the makespan, denoted by LB. Note that for the sake of simplicity of the developed backtracking procedure it stops when the first solution is obtained: this solution gives an upper bound (UB) of the makespan.

The backtracking procedure implements an integrated resolution: decisions of scheduling and allocation are simultaneous. To perform that, we choose a task, schedule it by fixing its start time, then allocate a resource to it, and so on for all the tasks, according to the predefined order. For variable ordering, tasks are sorted in increasing order of their start times and secondly in decreasing order of their delivery times to break the ties. Value ordering is achieved considering increasing time for temporal variable and increasing demand for resource variables.

To our knowledge no instances of mixed TRSA problems with versatile and unrelated resources exist in the literature, even though such problems are frequently encountered in enterprise organisations. Hence, we randomly generate instances to evaluate our approach. For the random generation of consistent TSRA problems, two features are to take into account to ensure the existence of a solution: the choice of consistent precedence constraints (no cycle) and loose deadlines for all the tasks.

In a first type of instances, hundreds of problems from 10 to 50 tasks are generated. The ratio of the number of tasks over the number of resources (R1) is from 1.8 to 3 while the ratio illustrating the maximum number of resource alternatives per task (R2) varies from 2 to 4. The maximum number of non-redundant precedence constraints has been chosen to be equal to the number of tasks. From the experiments it can be noticed that the larger R1 is, the lower the average number of instances solved to optimality (e.g., for 20 tasks, we obtain 26% of optimal solutions with R1=1.8, 18% with R1=2.2, 10% with R1=2.8). The impact of R2 on this number of

solved instances is not so clear; nevertheless the relative deviation $\Delta = {(UB - LB)}/{LB}$ increases with R2 (e.g., for 30 tasks and R1=2.3, $\Delta = 0.15$ with R2=2, $\Delta = 0.18$ with R2=3, $\Delta = 0.23$ with R2=4). If we now consider the number of solutions for which the obtained makespan is near the optimum, e.g., for 20 tasks and R1 equal to 1.8, 2.2, and 2.8, it yields respectively 48%, 34%, and 23% of solutions with $\Delta < 0.08$ (for the same value of Δ, 30 tasks and R1 equal to 2, 2.3, and 2.7, we obtain 31%, 25%, and 16% of solutions). For information the average CPU time needed to reach a solution is 0.56 s for problems involving 30 tasks.

In a second type of instances R1 is between 5 and 7; all other parameters remain unchanged from the first type of instances. For these instances, our current procedure does not compute solutions for which Δ is of high quality. The average CPU time is 2.9 s for problems of 50 tasks.

4. Discussion

In this work we address Task Scheduling and Resource Allocation problems with versatile and unrelated resources. A solving procedure supported by constraint propagation, chronological backtracking and ordering heuristics has been developed. The analysis of first experiments on randomly generated instances shows that despite the simplicity of the solving procedure (we search only for the first solution) the results in terms of number of problems solved to optimality are encouraging for instances where the ratio of the number of tasks over the number of resources is not too high. When this ratio becomes higher (i.e., the number of resource conflicts is greater) our approach is not yet powerful enough. However we claim that this can easily be improved in further works, mainly delving the research in two directions: implementing efficient constraint propagation rules such as edge-finding; devising a branch-and-bound procedure to increase the quality of the makespan.

References

[1] Brandimarte, P. (1993). Routing and scheduling in a flexible Job Shop by Tabu search, *AOR.*, 41, 157-183.

[2] Brucker, P., Jurisch, B., and Krämer, A. (1997). Complexity of scheduling problems with multi-purpose machines, *AOR.*, 70, 57-73.

[3] Dauzère-Pérès, S. and Paulli, J. (1997). An integrated approach for modeling and solving the general multiprocessor job shop scheduling problem using Tabu search, *AOR.*, 70, 281-306.

[4] Erschler, J., Roubellat, F., and Vernhes, J-P. (1980). Characterizing the set of feasible sequences for n jobs to be carried out on a single machine, *EJOR.*, 4, 189-194.

[5] Huguet, M-J. and Lopez, P. (2000). Mixed task scheduling and resource allocation problems. CP-AI-OR'00, pp. 71-79, Paderborn, Germany.

[6] Hurink, J., Jurisch, B., and Thole, M. (1994). Tabu search for the Job Shop scheduling problem with multipurpose machines, *OR Spektrum*, 15, 205-215.

[7] Lopez, P. and Esquirol, P. (1996). Consistency enforcing in scheduling: a general formulation based on energetic reasoning, PMS'96, pp. 155-158, Poznan, Poland.

[8] Néron, E., Baptiste, P., and Gupta, J.N.D. (2000). Solving Hybrid Flow Shop problem using energetic reasoning and global operations, Research Report, Univ. Techn. Compiègne.

[9] Nuijten, W. (1994). Time and resource constrained scheduling: A constraint satisfaction approach, PhD dissertation, Eindhoven University.

[10] Schwalb, E. and Dechter, R. (1997). Processing disjunctions in temporal constraint networks, *AI*, 93, 29-61.

[11] Sellami, I., Huguet, M-J., and Lopez, P. (2001). A backtracking algorithm for solving mixed task scheduling and resource allocation problems, ETFA'2001, pp. 741-744, Antibes, France.