A backtracking algorithm for solving mixed task scheduling and resource allocation problems

I. Sellami[1] – MJ. Huguet[1,2] – P. Lopez[1]

1. LAAS-CNRS, 7 avenue du Colonel Roche, 31077 Toulouse Cedex 4, France
2. INSA, 135 avenue de Rangueil, 31077 Toulouse Cedex 4, France
{isellami, huguet, lopez}@laas.fr

***abstract* - This paper addresses the solving of mixed Task Scheduling and Resource Allocation Problems in an integrated way using a backtracking algorithm. Several ordering heuristics are proposed to improve the efficiency of this algorithm. Experiments show the impact of these heuristics on the quality of the first solution obtained. We also compare our integrated approach with a sequential solving of scheduling and allocation problems.**

I. INTRODUCTION

This paper addresses mixed Task Scheduling and Resource Allocation (in short TSRA) problems, and more particularly Job Shop or Flow Shop problems with resource allocation constraints. The TSRA problem under study consists of a set of tasks to be performed by a set of renewable resources. Pre-emption is not allowed. For the sake of simplicity, a task is supposed to be performed by a single resource. Resources can be disjunctive (i.e. they process only one task at a time) or cumulative (i.e. they may process several tasks simultaneously). Tasks are linked together in chain by precedence constraints. To each task is associated a group of resources that are able to process it and the task duration may depend on the resource of the group it is assigned to.

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, 2, 3, 6]. 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 [4, 5] and [9]. In both these papers task durations are resource-dependent. In [9], a time-window is associated to each task for each possible allocation; constraint propagation mechanisms lead to the updating of task time-windows which may suppress some possible resource allocation. Moreover, specific developments involve the aggregation in a cumulative resource of the group of resources that may be allocated to several tasks. In [4, 5] the choice of resource allocation is considered as processing time constraints and some constraint propagation mechanisms, particularly based on energetic reasoning [7], are proposed. In this paper we propose an integrated approach, to solve jointly resource allocation and task scheduling problems.

In the following, we present a constraint-based model for TSRA problems and we propose a classification of these problems. In the next part, we develop a backtracking algorithm to solve TSRA problems in an integrated way. In this algorithm we more accurately study several variable and value ordering heuristics [10]. The last part concerns some experiments to compare our integrated approach with the method consisting in sequentially solving scheduling and allocation, and to evaluate the impact of heuristics on the solution obtained.

II. PROBLEM MODELLING AND TYPOLOGY

2.1. Problem Modelling

In the TSRA problem, a set T of tasks has to be achieved by a set R of resources. To model this problem, we need time and resource variables: let $\mathrm{V}_T = \{s_t, t \in \mathrm{T}\}$ be the set of time variables where s_t represents the starting time of task t; and let $\mathrm{V}_R = \{a_t, t \in \mathrm{T}\}$ be the set of resource variables where a_t represents the resource assigned to task t. In TSRA problems under consideration, we also distinguish time constraints and resource constraints. The time constraints are the following:

- *duration constraints*: the processing time of a task t may depend on the resource it is assigned to; it is denoted by p_{t,a_t}.
- *limit time constraints*: some tasks have to respect some release and due dates; for instance: $s_t \geq r_t$ and $s_t + p_{t,a_t} \leq c_t$ where r_t is the release date of task t and c_t its due date.
- *precedence constraints*: they are used for instance to reflect the routings. If a task i precedes a task j the associated constraint is $s_j \geq s_i + p_{i,a_i}$.

The resource constraints correspond to the allocation and to capacity restrictions.

- *resource allocation constraints*: the resource allocated to a task t belongs to a group of feasible resources denoted by G_t. This is expressed by: $a_t \in G_t$, $t \in \mathrm{T}$, $G_t \subseteq \mathrm{R}$. Here we limit ourselves to unary resource allocation constraints; binary constraints such as difference (for instance $a_i \neq a_j$ that is two tasks i and j cannot be allocated to the same resource) are not considered.
- *resource capacity constraints*: at any time, for a given resource, the quantities consumed of this resource by tasks for their processing must be lower than its capacity. Then, the capacity constraint is:

$$\sum_{t \in T / a_t = r} q_{t\,r} \le C_r \text{ at every time}$$

where $q_{t\,r}$ is the quantity of resource r used by a task t and C_r the capacity of this resource. Note that this constraint illustrating resource sharing constraints only concerns allocated tasks. We assume that both $q_{t\,r}$ and C_r are constant in time. Moreover we only consider unit quantity of resources, that is $q_{t\,r} = 1, \forall t \in T, \forall r \in G_t \,/\, a_t = r$.

2.2. Typology

Based on the model presented above, we distinguish four types of TSRA problems. The distinctions are made according to two underlying features of the resources which are the versatility and the equivalence.

- *TSRA problems with nonversatile and equivalent resources*. A resource is nonversatile if it can execute only a specific kind of tasks. In our model it means that a resource can belong to a single group of resources. Hence, for two groups of resources G_t and $G_{t'}$ either $G_t = G_{t'}$ or $G_t \cap G_{t'} = \varnothing$. In a group G_t resources are equivalent if the duration of a task t is the same whatever the resource of G_t used to achieve it. For example, problems like Hybrid Flow Shop [8] correspond to this kind of TSRA problems.
- *TSRA problems with nonversatile and nonequivalent resources*. In a group G_t resources are nonequivalent if the duration p_{t,a_t} of task t depends on the resource allocated to it. For instance, Job Shop problems with alternative sets of resources considered by [9] are included in this category.

In [8, 9], TSRA problems with nonversatile resources are modelled as scheduling problems with cumulative constraints. For instance, let i, j, k be three tasks that may be assigned on the same group of resources $\{r_1, r_2\}$, that is $G_i = G_j = G_k = \{r_1, r_2\}$. An aggregate resource can be associated to this group of which the global capacity is equal to the sum of the individual capacities of all two resources r_1 and r_2. In the disjunctive case (i.e., the capacities of r_1 and r_2 are equal to 1), the capacity of this aggregate resource is then 2 (i.e., no more than two tasks can be simultaneously performed on it). When the resources are nonequivalent, the task duration on the aggregate resource is the minimal duration on the group of resources. Then the TSRA problem can be studied like scheduling problem with cumulative resources.

- *TSRA problems with versatile and equivalent resources*. The versatility of resources means that a resource can achieve various types of tasks. In our model, it means that a resource may belong to several groups; i.e. the intersection of each two different groups of resources G_t and $G_{t'}$ may be empty or not. For example, [3, 6] deal with this kind of problems also called Multi-Purpose Machines Problems or Multi-Processor Job Shop problems.
- *TSRA problems with versatile and nonequivalent resources*. In these problems, a resource can belong to several groups of resources and task durations depend on the resource tasks are assigned to. This kind of TSRA problems is the most general one. To our knowledge, they are only studied in [4, 5] where the authors propose time and resource constraint propagation mechanisms. In the following part, we are concerned by solving such TSRA problems.

III. Resolution techniques

3.1. Introduction

The mixed TSRA problems are treated, in this paper, as Constraint Satisfaction Problems (CSP) [11]. Therefore, we exploited some CSP techniques to develop the resolution algorithms: backtrack algorithm, constraint propagation and instantiation heuristics.

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 instantiates the variables of the problem successively in a predefined order. Moreover, tests of consistency can be carried out, at the time of the instantiation of a variable, between this instantiation and those carried out on the former variables in the order, referring to the constraints, so as to check the consistency of current instantiation.

Usually, constraint propagation is used on top of the resolution to limit the size of the domain of the variables, or during the resolution to anticipate the failures and speed up the search for solutions. An instantiation heuristic involves a variable and/or value ordering. It aims to prune the tree and accelerate the search for the first solution.

3.2. Principle of the algorithm

For the developed algorithm, we were interested in the quality of the first obtained solution. We used, as mentioned previously, the backtrack algorithm, the constraint propagation (we limit ourselves to propagate only the temporal constraints before any resolution), and we were particularly interested in task ordering and value (time and resource) ordering heuristics.

Furthermore, we aimed to realize an integrated resolution: decisions of scheduling and allocation are simultaneous. To perform that, we choose a task, schedule it by fixing its starting time (i.e. by instantiating a temporal variable), then allocate a resource to it (i.e. by instantiating a resource variable), and so on for all the tasks, according to the predefined order. It is then necessary to give an order on the tasks and the resources.

3.3. Instantiation heuristics

I) Variable ordering: To obtain the task ordering, we have to calculate priority coefficients of the tasks related to their depth in the precedence graph. Practically for each task t there are two coefficients of priority: one on the basis of the predecessor tasks denoted by a_t (minimum time required for the execution of the preceding tasks) and the other on the basis of the successors denoted by b_t (minimum time required for the execution of the following tasks). Following the chosen heuristics, tasks can be ordered on the increasing or decreasing values of these coefficients.

Example - In this example, we handle five tasks and we observe only temporal constraints. The precedence graph is illustrated in Fig. 1. With these constraints, and under the assumption of unary processing times, we are able to calculate the priority coefficients (as shown in Table 1).

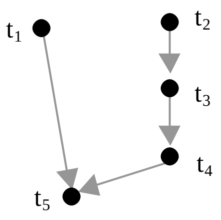

Figure 1. Precedence Graph.

	t_1	t_2	t_3	t_4	t_5
a_t	0	0	1	2	3
b_t	1	3	2	1	0

Table 1. Priority coefficients.

Note that to calculate these coefficients, it is better to work on a reduced precedence graph (redundant constraints are omitted). Transformation of the initial problem into a problem without redundant temporal constraints is quite simple since we consider only precedence constraints. In this example, according first to the increasing values of a_t, second to the decreasing values of b_t, the classification of tasks is t_2, t_1, t_3, t_4, t_5.

In case of equality of priority coefficients the order is established according to the supplied capacity of the group of resources that can process the concerned task. The task which has the highest priority is the one of which the supplied capacity of the corresponding group of resources is the lowest. Once the above coefficients have been obtained they can be directly used to propagate the temporal constraints. Indeed they make it possible to bring up to date the domain of the temporal variables. If we denote by $[\underline{s}_t, \overline{s}_t]$ the current domain of time variable s_t, $\forall t \in \mathcal{T}$, the deduced domain is $s_t \in [Max(\underline{s}_t, a_t), Min(\overline{s}_t, D-1-b_t)]$ (D is the maximum deadline i.e. the scheduling horizon or the latest finishing time of all the tasks).

II) Value Ordering: For time variables, we consider the increasing order of the values for the starting time, and for resource variables, the value ordering is related on the demands and the capacity of the resources.

3.4. Conclusion

The unfolding of the resolution algorithm follows the steps cited below. In input, different variables of the problem with the constraints that bind them have to be given. Then follows a step of variable and value ordering with the chosen heuristic (the reduction of the precedence graph and the calculation of the priority coefficients are achieved before) and temporal constraint propagation using the pre-calculated priority coefficients. After that, the backtrack algorithm is performed on the pre-ordered and pruned search tree. Finally, the first solution (if it exists) is then obtained.

We characterize a family of heuristics according to the following criteria:

- *variable ordering*: tasks are sorted in decreasing or increasing order of coefficient a_t then decreasing or increasing order of coefficient b_t, or conversely.
- *value ordering*: the values of resource variables are sorted in decreasing order of their demand then in increasing order of their capacity, or conversely. For time variables, the order of their values follows the increasing dates.

The instantiation heuristics are static i.e. determined before any resolution. The constraint propagation relates only to the temporal constraints and is realized on top of the resolution. In the next section, we present different experiments to evaluate the developed heuristics of instantiation for the TSRA problems.

IV. EXPERIMENTS

To our knowledge no example of the most general mixed TRSA problems (versatile and nonequivalent resources) exists in the literature. Hence, in order to evaluate the resolution techniques proposed in §3, we randomly generate TSRA problems.

4.1. Random generation of problems

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 a loose deadline for all the tasks.

For a given number of tasks (from 10 up to 200), a hundred of problems with different hardness have been generated. The total supplied average-capacity has been chosen to be proportional to the number of tasks (25%) and the number of precedence constraints has been chosen between 150% and 200% of the number of tasks.

4.2. Tested algorithms

We have tested three different heuristics (termed 1, 2 and 3) for the backtrack algorithms with the integrated approach. To compare this integrated approach for the resolution of mixed TSRA problems, three backtrack algorithms of resolution based on an approach by decomposition in subproblems have been tested. In two of them (4 and 5), the allocation problem is solved independently before the scheduling problem. In the latest one (6), the scheduling is solved independently before the allocation.

	Approach	Instantiation Heuristic
(1)	integration	*Task Ordering*: increasing values of a_t then decreasing values of b_t *Resource Ordering*: increasing capacity then decreasing demand
(2)	integration	*Task Ordering*: decreasing values of b_t then increasing values of a_t *Resource Ordering*: idem (1)
(3)	integration	*Task Ordering*: decreasing values of

		a_t then increasing values of b_t *Resource Ordering*: idem (1)
(4)	decomposition: allocation then scheduling	No use of heuristic
(5)	decomposition: allocation then scheduling	*Task Ordering*: idem (1) *Resource Ordering*: idem (1)
(6)	decomposition: scheduling then allocation	*Task Ordering*: idem (1) *Resource Ordering*: idem (1)

For a given number of tasks we compared on average the algorithms' performances for a hundred of generated problems. For the evaluation of these algorithms, the basic criteria that we took into account are the run time of the algorithm to obtain the first solution and the problem makespan that is the total duration for the processing of all the tasks of the problem.

4.3. Results

In view of the experiments, the procedures at hand for solving TSRA problems can be cast into two categories: *fast* and *slow* algorithms. For the *fast algorithms*, which are (1), (2), (4) and (5), we compared the run times and the makespans. It appears first that the algorithms with the integrated approaches (1) and (2) are equivalent; second, the introduction of heuristics improved the run time and the quality of the first obtained solution of the algorithm compared with an approach by decomposition: allocation then scheduling. Algorithms (1) and (2) give a better run time than algorithm (4) but worse than (5) but the quality (in terms of makespan value) of the first obtained solution is better for the algorithms with the integrated approach; those with a decomposition approach give quickly a very high makespan, which attends, for a hundred of tasks for example, the triple in comparison with the integrated approach. Indeed, with this decomposition approach where allocation is done before scheduling, scheduling decisions are the first to be backtracked on, in case of failure, by delaying the starting time of the tasks what causes the deterioration of the quality of the first obtained solution but does not take much run time.

For the *slow algorithms*, that is (3) and (6), we compared only the makespan with a fast algorithm (1). For all these slow algorithms, the first obtained solution has a better quality in comparison with fast ones. But the run time can be prohibitory (in a one-week delay, the slow algorithms do not inevitably find the first solution). However, the algorithm with an integrated approach (3) gives a better quality of solution. The slowness of these algorithms can be explained, for algorithm (3), by the fact that the tasks are ordered by the decreasing priority which increases the number of failures and, for algorithm (6) that in case of failure, the backtrack on the decisions of scheduling are often not sufficient and even decisions of allocation would have to be reviewed.

V. CONCLUSION

In this paper we have addressed general Task Scheduling and Resource Allocation problems with versatile and nonequivalent resources. We have presented a backtracking algorithm to solve these mixed TSRA problems in an integrated way: both problems – scheduling and allocation – are simultaneously solved. Our integrated approach has been compared with sequential ones – scheduling then allocation or conversely – and we have tested the impact of several heuristics for variable and value ordering onto the first solution obtained. We were more particularly interested in the run time of algorithms to obtain the first solution and the total duration of tasks for this solution. To lead these experiments, we have randomly generated some TSRA problems. These results are really encouraging: with the proposed integrated approach the first solution is rapidly obtained and the total duration of tasks is lower than those of the solutions obtained with decomposition approaches. According to heuristics, we give importance either to the speed to obtain a solution or to the quality of the solution.

It would be interesting to carry out some tests on real problems for which we know the best solution or on most constrained problems. To improve our algorithm it would be interesting to use dynamic heuristics for variable and value ordering and to apply constraint propagation during the resolution. Moreover, other constraint propagation, for instance propagation of resource constraints [4, 5], would improve the results of our algorithms if included in them.

VI. REFERENCES

[1] P. Brandimarte, "Routing and scheduling in a flexible Job Shop by Tabu search", *Annals of Oper. Res.* 41, pp. 157-183, 1993.

[2] P. Brucker, B. Jurish, A. Krämer, "Complexity of scheduling problems with multipurpose machines", *Annals of Oper. Res.* 70, pp. 57-73, 1997.

[3] S. Dauzère-Pérès, J. Paulli, "An integrated approach for modeling and solving the general multiprocessor job shop scheduling problem using tabu search", *Annals of Oper. Res.* 70, pp. 281-306, 1997.

[4] Huguet MJ., Lopez P.: An integrated constraint-based model for task scheduling and resource assignment. CP-AI-OR-99, Ferrara, Italy, 1999.

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

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

[7] 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.

[8] Néron E., Baptiste P., Gupta J.N.D.: Solving Hybrid Flow Shop problem using energetic reasoning and global operations, Research Report, Université Technologique de Compiègne, 2000.

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

[10] Smith S., Cheng C.,: Slack-based heuristics for constraint satisfaction scheduling. AAAI-93, pp. 139-144, Washington, 1993.

[11] Tsang E.P.K.: Foundations of Constraint Satisfaction. Academic Press Ltd., London, 1993.