AN INTEGRATED CONSTRAINT-BASED MODEL FOR TASK SCHEDULING AND RESOURCE ASSIGNMENT

MJ. Huguet[1,2] – P. Lopez[1]

[1] *LAAS - CNRS*
7 av. du Colonel Roche
31077 TOULOUSE Cedex 4 - FRANCE

[2] *Université de Pau*
Dept. Informatique LIA
64 000 PAU - FRANCE

e-mails: huguet@laas.fr, lopez@laas.fr

Abstract: We propose an extension of classical scheduling models in order to tackle resource assignment constraints. It is based on the interaction of rules dedicated to the propagation of time constraints for scheduling problems on the one hand, and the simplification of resource assignment on the other hand. We present a polynomial approximation algorithm to limit the decision space in terms of resource alternatives and time. It is evaluated on classical scheduling instances in which resource flexibility has been introduced.
Key words: Task Scheduling - Resource Assignment - Temporal Constraints - Constraint Modeling - Constraint Propagation

1. PROBLEM STATEMENT

The problem under study consists of a set of N tasks to be performed by a set of K renewable resources. Pre-emption is not allowed. For the sake of simplicity, a task is supposed to be performed by a single resource (mono-resource). Resources can be disjunctive (process only one task at a time) or cumulative (may process several tasks simultaneously). To each task is associated a set of resources that are able to process it and the task duration may depend on the resource on which it is assigned to.

A task i is characterized by its starting time st_i, its finishing time ft_i, and the set of allocated resources A_i; processing times are associated to each alternative of this set.

Thus to take account of resource allocation while focusing on time, the scheduling and allocation problem is modeled by a set of decision variables $V = \{st_i, ft_i \mid i \in T\}$. Using both starting and finishing time variables allows the consideration of not-fixed durations for unsolved assignment problems. We model all binary temporal constraints in the standard form (by adding in V an origin time variable denoted by x_0): $x_j - x_i \geq a_{ij}$, where $x_i, x_j \in V$ and a_{ij} is a parameter with a positive, negative or null value.

We consider the following constraints:

Limit times constraints: the starting time (resp. finishing time) of a task $i \in T$ is included between an earliest time est_i (resp. eft_i) and a latest time lst_i (resp. lft_i): $(st_i - x_0 \geq est_i) \wedge (x_0 - st_i \geq -lst_i)$ (resp. $(ft_i - x_0 \geq eft_i) \wedge (x_0 - ft_i \geq -lft_i)$).

General precedence constraints: It includes the classical precedence relations (e.g., routing).
For instance: a task j cannot be performed before the previous task i is finished: $st_j - ft_i \geq 0$.

It also concerns the basic *irreducible* temporal relations as those described by Allen (overlaps, meets, starts, …).

Resource-dependent duration constraints: The description of a task leads to alternatives on the resource on which it is assigned to; to each alternative is associated a duration bounded by a minimal value and a maximal value. A task i can be processed on resources in A_i with duration included in $[p_i^k, P_i^k]$, that is:
$(ft_i - st_i \geq p_i^k) \wedge (st_i - ft_i \geq -P_i^k)$, $\forall k \in A_i$.
This can be aggregated in: $(ft_i - st_i \geq \min_{k \in A_i}(p_i^k)) \wedge (st_i - ft_i \geq -\max_{k \in A_i}(P_i^k))$. If there are some disjunctions in the intervals $[p_i^k, P_i^k]$, for all $]a,b[$ such that $]a,b[\notin \bigcup_{k \in A_i} [p_i^k, P_i^k]$ ($]a,b[$ is called a **hole**) we must take

into account the additional *non-conjunctive set of temporal constraints* (temporal constraints linked by $\vee$): $st_i - ft_i \geq -a \vee ft_i - st_i \geq b$.

For instance, let consider a task i which can be processed either on resource m_1 with duration [2,4] or on resource m_2 with duration [7,10] (]4,7[is a hole in the set of permitted intervals). This can be modeled by the following temporal constraints: $(ft_i - st_i \geq 2) \wedge (st_i - ft_i \geq -10) \wedge (st_i - ft_i \geq -4 \vee ft_i - st_i \geq 7)$.

Resource sharing constraints: Resource allocation and task scheduling problems are linked together by resource sharing constraints, due to the limited availability of each resource. Here we consider resource sharing constraints only between tasks allocated to a given resource. We denote by T_a the set of assigned tasks: $T_a = \{i \in T / |A_i| = 1\}$. The resource sharing constraints can be modeled via the concept of critical set of conflicting tasks [2][4].

Definition. A set E_k is a *critical set of conflicting tasks* $i \in T_a$ for a resource $k \in K$, if

$$\exists t, \sum_{i \in E_k | st_i \leq t \leq ft_i} q_i^k > Q_k \text{ and } \forall t, \forall j \in E_k, \sum_{i \in E_k \setminus \{j\} | st_i \leq t \leq ft_i} q_i^k \leq Q_k$$

where Q_k is the *capacity* of resource k and q_i^k ($q_i^k \leq Q_k$) states for the *intensity* demanded from a task i to be completed on the resource k.

As E_k is minimal, it is necessary and sufficient to sequence two tasks within the critical set to solve a conflict. Note that to each critical set E_k corresponds a *non-conjunctive set of temporal constrains* representing the possibility for sequencing the conflicting tasks of E_k.

In *disjunctive problems*, like job-shops or flow-shops, the capacity of a resource $k \in K$ is $Q_k = 1$, and for a task i, $q_i^k = 1$, $\forall k \in A_i$. Every critical set consists of only two tasks and the search for conflicting tasks is quite simple as it consists in enumerating couples of tasks sharing the same resource.

For instance, consider three allocated tasks {i, j, l} sharing the same resource k simultaneously. There are three critical sets for resource k: $\{i,j\}$, $\{i,l\}$, $\{j,l\}$ they are represented by three non-conjunctive sets of temporal constraints (with two constraints in each non-conjunctive set):
$(st_j - ft_i \geq 0 \vee st_i - ft_j \geq 0) \wedge (st_l - ft_i \geq 0 \vee st_i - ft_l \geq 0) \wedge (st_l - ft_j \geq 0 \vee st_j - ft_l \geq 0)$.

In *cumulative problems*, the search for critical sets requires an exponential complexity (partitioning problem). For solving such a conflict it is necessary and sufficient to sequence two tasks within each critical set. In the general case, the search for critical sets requires an exponential complexity. However in a practical way, giving initial time windows which weakly overlap each other greatly reduces this complexity.
With such a formalism, we are able to model disjunctive problems, and cumulative problems as well. However, to limit the size of the model and to perform as efficiently as possible the constraint propagation rules, we only consider disjunctive problems in the sequel.

2. MODELING

Our model is at the intersection of studies in scheduling theory [8] and Temporal Constraint Satisfaction Problems [3][1], especially for the modeling of the resource-dependent duration constraints.

The problems where all the constraints have to be simultaneously satisfied are called *conjunctive* problems. They correspond to *Simple Temporal Problems*. Such problems arise for time location problems which are not subject to resource constraints. When some constraints are described by a logical disjunction of constraints, the problem is said to be *non-conjunctive*.

Hence, the set of constraints can be written: $C = C_c \wedge C_{nc}$ where C_c is the conjunctive set of temporal constraints and C_{nc} is the set of non-conjunctive sets of temporal constraints.

Let now detail the expression of each of these sets:

- $C_c = \bigwedge_{(x_i, x_j) \in V^2} c_{ij}$ where c_{ij} is the temporal constraint $x_j - x_i \geq a_{ij}$;
- $C_{nc} = \bigwedge_d C_{nc}^d$ with d the number of the non-conjunctive set.

The resource-dependent duration constraints and the resource sharing constraints lead to non-conjunctive sets of constraints. For resource-dependent duration constraints when some intervals are pairwise disjoint to each hole is associated a non-conjunctive set with two binary constraints. For resource sharing constraints in the disjunctive case, there are only two temporal constraints in each non-conjunctive set. Then, each non-conjunctive set C_{nc}^{d} can be written: $C_{nc}^{d} = c_{ij} \vee c_{lm}$ with c_{ij} and c_{lm} two binary constraints. The number of non-conjunctive sets d is bounded by the maximum number of critical sets plus the maximum number of holes, that is $K.N.(N-1)/2 + N.(K-1)$.

3. CONSTRAINT PROPAGATION ALGORITHM

We propose an algorithm for characterizing the feasible schedules and the feasible resource assignments. This algorithm is based on several propagation rules. Each rule is concerned with the removal of inconsistencies.

R1. Propagation on the conjunctive set of constraints C_c by achieving 3-consistency (e.g., Floyd and Warshall's all-pairs-longest-path algorithm).

R2. Simplification of the set of non-conjunctive sets of constraints C_{nc} compared with C_c (removal of inconsistent constraints, removal of redundant non-conjunctive set) [5].

These two rules are devoted to scheduling or assignment problems. They are based on the constraints modeling only. In the case of assignment problems, theses two rules correspond to algorithm "Upper-Lower Tightening" proposed in [9].

R3. Simplification of resource sharing constraints for yet assigned tasks using *energetic adjustments* [6]. This rule is dedicated to the disjunctive scheduling problem. It is based on the analysis of sets of tasks competing for the same resource to be performed.

R4. Simplification of the resource assignment constraints by interpretation of temporal deductions due to R1 and R2. This rule is dedicated to the assignment problem. In fact, deductions on feasible durations have been made by rules R1 and R2; Rule R4 expresses these deductions in terms of resource assignment.
For instance, let consider a task i which can be processed either on resource m_1 with duration $[2,4]$ or on resource m_2 with duration $[7,10]$. If the deduced duration (by applying R1) is $[6,9]$ then by applying R2 on the non conjunctive set $(st_i - ft_i \geq -4 \vee ft_i - st_i \geq 7)$, the constraint $st_i - ft_i \geq -4$ is inconsistent. Rule R4 then removes the assignment m_1 for task i and update the feasible duration for task i on resource m_2: $[7,9]$.

R5. Propagation of resource assignment constraints by removing not feasible alternatives from not yet assigned tasks. For each assigned task on a given resource k, the rule considers each of not yet assigned tasks for which alternatives include resource k. It checks the consistency of the assignment of the task to resource k by applying R3. If the assignment is inconsistent, k is removed from the set of alternatives for the task.
For instance, let consider a task i which can be processed either on resource m_1 or on resource m_2, and a task j which must be processed on resource m_1. If the assignment m_1 for the task i is inconsistent by applying R3 then it removed from A_i.
Rules R4 and R5 only consider the resource assignment constraints. In these rules, when a task is assigned, one searches for new resource sharing constraints.

In our constraint propagation algorithm, rules from R1 to R5 are iteratively applied until no more deduction appear or an inconsistency is detected. For the control of the algorithm, an important feature is that rule R1, of time complexity in $O(|V|^3)$, is too much time consuming in comparison with other rules. Therefore a good strategy will restart R1 as less as possible. From our experiments the best strategy follows the skeleton:

```
loop
        R1
        loop
                R2
                R3
                R4
                R5
        until no more deduction or inconsistency
until no more deduction or inconsistency
```

Our constraint propagation algorithm is sound but not complete; it is able to classify a problem as inconsistent but cannot prove its consistency. As a result, it is referred as an *approximation algorithm*.

4. EXPERIMENTS

Our experiments concern classical Job-Shops in which we modify the instances to introduce an alternative resource on which the operation can also be performed, and Hybrid Flow-Shops (HFS) which consist of a number of stages in series with a number of machines in parallel at each stage. For each instance, the previous algorithm is embedded in a more general procedure that aims to compute a lower bound on the makespan. The procedure starts with a scheduling horizon H initially given by a trivial lower bound on the makespan (e.g., the greatest sum of the minimum durations of the job operations for each resource). At this step, if the constraint propagation does not detect an inconsistency this bound is stored as the best lower bound found and the procedure stops. If the instance is proved to be infeasible by the constraint propagation algorithm the procedure is repeated increasing H. If not, we repeat the procedure lowering H. The procedure stops with the minimum value for H such that the instance is not proved to be infeasible.

```
H ← Initial Value
loop
        Constraint Propagation
        if an inconsistency is detected then
                increase H
        else
                if H = Initial Value then
                        exit
                else
                        decrease H
                end if
        end if
until H is minimum
```

The current experiments give some results for the lower bound on the makespan and for the set of feasible decision. We present here some results for HFS. Each HFS problem is characterized by a number of stages, a number of feasible resources for each stage, and a number of jobs. We only consider HFS instances for which there are different durations at each stage. The following HFS can be found at http://192.54.142.54/~grangeon. with the best lower bound given in column "LB1".

		size	LB1	LB2	H_{min}	CPU time H_{min}
	ro1	# stages: 2 (1,2) # jobs: 8	34	11	34	0.44 sec
	ro2	# stages: 2 (2,1) # jobs: 50	2791	146	2791	34.47 sec
	ro3	# stages: 3 (3,2,3) # jobs: 15	211	161	175	9.06 sec
	ro6	# stages: 5 (1,1,2,2,6) # jobs: 50	2489	2489	2489	115.72 sec

Column "LB2" provides the lower bound of the makespan by assigning each task to the resource with minimal duration (the initial value for H); Column "H_{min}" gives the minimum scheduling horizon for which our algorithm cannot deduce an inconsistency; Column "iter H_{min}" gives the number of loops made to find the minimal value of H; Column "CPU time H_{min}" gives the CPU time obtained to apply the constraint propagation algorithm on the problem with the scheduling horizon H_{min}.
Moreover, we apply our algorithm on classical Job-Shop problems (Fisher and Thompson's instances) in which we modify the instances so as to introduce an alternative resource. To each task i we have the duration p_i, we introduce two alternatives : the first one with the duration in $[p_i, 2p_i]$ and the second one with duration included in $[4p_i, 6p_i]$.

		size	LB2	H_{min}	CPU time H_{min}
	ft02	# jobs: 6 # op/job: 2	15	45	0.06 sec
	ft52	# jobs: 5 # op/job: 2	15	32	0.06 sec
	ft06	# jobs: 6 # op/job: 6	47	67	2.93 sec
	ft10	# jobs: 10 # op/job: 10	655	910	65.94 sec

However, to compare our approach with other works, we need to extend our model to tackle problems referred as JSSPRSA in [7] in which a task need several resources simultaneously to be performed (*multi-resource problem*) and there are some alternatives for resources.

The evaluation of our approach cannot be restricted to the computing of the lower bound on the makespan for which dedicated methods outperform our procedure. We also need to consider the tightening of the feasible decision space by constraint propagation by the percentage of not detected inconsistencies.

REFERENCES

[1] S. BELHADJI, A. ISLI, *Temporal constraint satisfaction techniques in job shop scheduling problem solving*, Constraints: an International Journal, 3, pp. 203-211, 1998.

[2] R. BELLMAN, A.O. ESOGBUE, I. NABESHIMA, *Mathematical aspects of scheduling and applications*, Pergamon Press, 1982.

[3] R. DECHTER, I. MEIRI, J. PEARL, *Temporal constraint networks*, Artificial Intelligence, 49, pp. 61-95, 1991.

[4] J. ERSCHLER, G. FONTAN, F. ROUBELLAT, *Potentiels sur un graphe non conjonctif et analyse d'un problème d'ordonnancement à moyens limités*, RAIRO/Operations Research, 13, pp. 363-378, 1979.

[5] P. ESQUIROL, M.J. HUGUET, P. LOPEZ, *Modeling and managing disjunctions in scheduling problems*, Journal of Intelligent Manufacturing, 6, pp. 133-144, 1995.

[6] P. LOPEZ, P. ESQUIROL, *Consistency enforcing in scheduling: A general formulation based on energetic reasoning*, 5th International Workshop on Project Management and Scheduling, pp. 155-158, Poznan (Poland), April 11-13, 1996.

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

[8] M. PINEDO, *Scheduling: Theory, Algorithms and Systems*, Prentice Hall, 1995.

[9] E. SCHWALB, R. DECHTER, *Processing disjunctions in temporal constraint networks*, Artificial Intelligence, 93, pp. 29-61, 1997.