Modeling and managing disjunctions in scheduling problems*

Patrick ESQUIROL, Marie-Jo HUGUET and Pierre LOPEZ

Laboratoire d'Analyse et d'Architecture des Systèmes du C.N.R.S.

7, avenue du Colonel Roche - 31077 Toulouse Cédex - FRANCE

In this paper we present a method for modeling and managing various constraints encountered in task scheduling problems. Our approach aims at characterizing feasible schedules through the analysis of the set of constraints and their interaction, regardless to any optimization criteria. This analysis is achieved by a constraint propagation process on a constraint graph and produces both restricted domains for the decision variables and an updated formulation of the initial constraints.
The graphs usually used to model temporal constraints seem to be limited because they only allow the representation of *strict precedence* relations between two tasks. In order to model a larger variety of temporal constraints, particularly any constraint that connects two events (start- or finish-time of a task), we propose a model called a *Time-Bound-On-Node* graph in which each task is featured by two nodes. Then it becomes possible to handle constraints on task durations, due for example to flexibilities in resource utilization. This kind of graph is not new and has already been investigated in related works on project planning and constraint satisfaction problems. But its processing and interpretation deserved to be developed, particularly for our purpose, which is the search for the necessary conditions of feasibility.
With respect to conjunctive temporal constraints, the analysis is achieved with a polynomial algorithm based on the longest path search on a conjunctive TBON graph, yielding the necessary and sufficient conditions of feasibility.
Taking account of resource constraints leads to defining disjunctive constraints. To this end, disjunctive sets of arcs are introduced, making the TBON graph non conjunctive. In this case, a complete characterization of feasibility cannot reasonably be faced, due to the combinatorial feature. Nevertheless, a polynomial algorithm that applies reduction and deletion rules on the non conjunctive part of the graph is proposed to restart new propagations on the conjunctive part until all deductions have been made.

Keywords : time graph, task scheduling, constraint propagation, feasibility analysis

1. Introduction

Generically, "task scheduling problems" may cover an array of applications : project planning, manufacturing shop scheduling or student course scheduling. In all these cases, one

* Published in *Journal of Intelligent Manufacturing*, 1995, Vol.6, pp.133-144.

must decide where to locate a set of tasks in the time, each task needing one or several resources during processing.

Addressing scheduling problems may be based on several approaches. Conventional approaches envisage the scheduling issue as an optimization problem and propose exact or heuristic techniques aiming to find the best solution with regard to some criteria which is supposed to be relevant (Graves, 1981), (GOThA, 1993). Since it might be ambitious to reduce the knowledge on objectives and the quality of the solutions through use of a single criterion only, more recent approaches have emphasized the importance of combining both analytic knowledge and domain-specific knowledge when humans and scheduling software systems have to co-operate (expert systems, decision aid systems) (Kusiak, 1989). Here the advantages and disadvantages of the above mentioned approaches will not be reviewed. It may be stated however that constraints must play a key role because they implicitly define the feasibility of the solutions, and solutions need to be feasible prior to being perfect.

Thus this work is not directly concerned with some solving methods, like Branch & Bound optimization procedures for example (Herroelen and Demeulemeester, 1992). Our main goal is to prepare for the final phase of solution generation by revealing inconsistencies between constraints and/or restricting as much as possible the domain of decision variables, for obtaining the real consistent sets of decisions. In this paper, this is referred to as the *analysis/generation* (vs. global solving) approach. It has several advantages :

- It leads to a clear distinction between what is already decided implicitly, due to the combined effect of constraints and what is worth deciding due to the preferences or optimization criteria. This distinction cannot be made in a one-phase decision procedure.

- It is more flexible when scheduling is solved in real-time or in co-operation with human decision makers. New task arrivals, disturbances during execution, real-state information, human decisions can easily be integrated as new constraints, whose consistency is checked by a simple constraint propagation, without re-building a global solution.

In §2, we present some temporal constraint graphs and in particular our model, the TBON graph, for modeling a wide class of time constraints. §3 is dedicated to constraint propagation in the case of conjunctive graphs. When the solution feasibility is expressed by disjunctive temporal constraints, it leads to analysing constraint propagation on non conjunctive TBON graphs. This is discussed in §4 and illustrated in §5. Of course a complete constraint propagation on non conjunctive graphs would not be not realistic and we would rather propose a partial but sound characterization of feasible solutions. It is based on an iterative procedure applying deletion and simplification rules to the non conjunctive part of the graph,

which may enforce propagation on the conjunctive part. An algorithm is proposed and some computational results are given in §6. The conclusion is presented in §7.

2. Time constraints in scheduling problems

The purpose of this paper is to formulate constraint satisfaction rules in task scheduling problems. As we are only focusing on feasibility, a generic statement is proposed, discarding all aspects of the optimization problem.

2.1. *Generic statement for task scheduling problems*

A task scheduling problem is defined by a 4-uple (T,R ,V,C) where :

- T is a set of tasks
- R is a set of resources
- V is a set of decision variables
- C is a set of constraints

Given a set of tasks T that can be performed by a set of resources R , the problem lies in determining V relative to each constraint of C. It may lead to defining different decision problems, that are often linked to one another and causing the heterogeneous nature of the sets of decision variables and constraints in real-life problems.

2.1.1. *The pure time allocation problem*

The pure time allocation problem is the most conventional form of task scheduling problems. Let n be the number of tasks. Task durations $\{D_i,\ i = 1,...,n\}$ are generally known and the set V of decision variables is $S = \{S_i,\ i = 1,...,n\}$, where S_i is the start-time of task i. Therefore, if needed, the set F of finish-times, $\{F_i, i = 1,...,n\}$, can be derived from the duration equalities : $D_i = F_i - S_i$, $i = 1,...,n$. When durations are not fixed (e.g. resource allocation dependent), the solution is characterized by both start-time S_i and finish-time F_i for each task i, or at least by two time variables among $\{S_i, F_i, D_i\}$.

Two classes of time constraints will be distinguished :

- *Absolute temporal localization.*

  These constraints restrict the temporal domain of one event, essentially start- or finish-event of a task (e.g. $\underline{S}_i \le S_i \le \overline{S}_i$ states one restricted time interval $[\underline{S}_i,\ \overline{S}_i]$ for start-time S_i). Release dates, due dates, available or frozen periods on a calendar, are instances of

this class of constraints. They are unary constraints in the sense that only one time variable is involved in the constraint. Also, it does not mean that only one unary inequality suffices. It is often necessary to use conjunctive or disjunctive sets of unary inequalities to restrict the domain of a temporal variable.

- *Relative temporal localization*.

  They represent temporal relations between two different events. A canonical form of these constraints appears as *potential inequalities* (Roy, 1970, §VIII-A-3), e.g. $S_j - S_i \geq a_{ij}$, where S_i and S_j represent the dates of two events and a_{ij}, a minimum difference between start-time of tasks i and j. They are binary constraints in the sense that two time variables are involved in the constraint. Precedence constraints, minimum/maximum distance constraints, minimum/maximum duration constraints specifically belong to this class and can easily be represented by such potential inequalities (Elmaghraby and Kamburowski, 1992).

It can easily be shown that when they are expressed by potential inequalities, relative temporal localization constraints can cover absolute localization constraints, just by forcing one of the time bound to be a constant.

A representation of time constraints between two tasks can be found in (Bartusch et al., 1988), called *time lags*. Surprisingly, the authors do not consider the case of constrained variable durations, which are a simple instance of potential inequalities between the start- and finish-event of a task.

2.1.2. *Resource management problems*

Resource management problems primarily derive from limitations on resource availability, whatever their nature (limitations on type, intensity or energy) ; these limitations have impact the temporal location of tasks.

In a *resource allocation problem*, the description of tasks leads to alternatives in the choice of the precise subset of resources which will perform task i. Here we consider the most general case in which each task simultaneously needs several resources to be processed (i.e., one man, one tool, and one machine). Let G_i be a disjunctive set of alternatives needed to perform i. Each alternative g of G_i can be defined by a couple (R_i^g, D_i^g), $1 \leq g \leq |G_i|$, where :

- R_i^g is a subset of resources : $R_i^g \subset R$;
- D_i^g is the duration of i when performed by R_i^g.

A solution of the pure allocation problem will be a set of n allocation values, $V = \{v_1,...,v_n\}$, one for each task, such that each v_i is a choice of one alternative of G_i.

The resource allocation problem is often combined with a *resource sharing problem*, due to the limited availability of each resource. When the same resource is assigned to several tasks, sharing conflicts may arise. To address this issue, a minimum of two tasks must be located in disjoined time intervals (sequencing decision). To avoid conflicts one can also modify allocation decisions, thus highlighting the close relationship between the two previous problems.

In the most general case, called the *cumulative resource problem*, conflicts are well-characterized by determining of all the minimal sets of conflicting tasks, called *conflict sets* in (Bellman et al., 1982), or *forbidden sets* in (Bartusch et al., 1988). A necessary and sufficient condition for solving all conflicts is to add one precedence relation between two tasks of each conflict set. As these sets may have nonempty intersections, one precedence relation can solve several conflict sets at a time. Thus any algorithm that gradually solves conflicts by means of precedence relations must update these sets dynamically.

In the particular case of the *disjunctive resource problem*, where any two tasks may compete for the same shared resource (e.g. single machine scheduling), conflict solving leads to a strictly ordered sequence for each set of tasks sharing the same disjunctive resource.

Thus, solving resource conflicts requires determining a partial or strict order amongst some tasks belonging to particular sets. These decisions can be *explicitly* represented by means of dedicated variables (e.g. one Boolean variable for each pair of tasks that must be sequenced, or one rank variable, integer, for each task in the case of a disjunctive resource problem (Amamou et al., 1992). They can also be *implicitly* expressed by means of disjunctive time constraints between time bound variables. In the following we focus on this second way of modeling, because on the one hand, it does not increase the number of decision variables (V amounts to the set $S \cup F$ of start- and finish-times) and on the other, these constraints can be modeled as disjunctive sets of elementary time constraints (see §2.4.2).

2.2. *Activity, project and constraint networks*

The modeling and analysis of time constraints play a key role in the literature of scheduling and project management. The first conventional models based on graphs were the activity-on-arc and activity-on-node graphs (Dibon, 1970), (Elmaghraby, 1977). Since each activity is represented by one entity of the graph only (node or arc), these graphs only capture temporal constraints between some pairwise activities, such as precedence constraints. When

constraints are more specific (i.e., minimum/maximum distances between two events (start- or finish-times), dummy activities, or subactivities (with possible negative durations) are introduced in order to accomodate the precedence relations.

We see two common drawbacks in these graphs : (1) they do not represent variable-durations activities ; (2) they only consider resource sharing constraints as disjunctive constraints. It is shown later (§2.4.2) that disjunctive constraints may also arise when several allocation-dependent durations are specified, or when several disjoined time intervals (typically a discontinuous calendar) are possible to perform tasks.

More recently, a model called "project network with time lags" has been proposed in (Bartusch et al., 1988). Time lag constraints express the minimum and/or maximum distances between start- and/or finish-times of any two jobs. However, the authors do not directly address formal properties and computational aspects in their initial model (network with typed constraints), but in a more standard representation, the *constraint digraph*, which is nothing less than an activity-on-node graph. Again, duration constraints, that are in fact time lags on the same job, are not taken into account.

Elmaghraby and Kamburowski (1992) have put forward an activity network supporting "general precedence relations" (GPRs). They make the assumption that activities do not have a fixed duration, but can be "compressed" or "expanded" from their "normal" duration, at a price. The goal is to analyze the feasibility and then the flexibility due to variable durations in order to get the cheapest schedule. To handle variable durations, they represent each activity i ($1 \leq i \leq n$) by two "event"-nodes, labelled 2i-1 for the start-event node and 2i for the finish-event node, respectively. The set of possible GPRs is {SS, FF, SF, FS, BS, BF, SE, FE}, where S stands for start-time, F for finish-time, B for the begin-event of the project (node 0) and E for its end-event (node 2n+1). Graphically, each node states its name j and the computed time bounds $[\underline{t}_j, \overline{t}_j]$, of the event. Each arc is either a GPR or a minimum/maximum duration for some activity.

The main advantage of this model is to express minimum/maximum duration constraints. This model is closely akin to our representation given hereafter. However, interpretation is slightly different, since the processing of this graph by any longest/shortest path algorithm (the authors recommend a modified version of Bellman's one (Bellman, 1958)) only updates the minimum/maximum values $[\underline{t}_j, \overline{t}_j]$ for the start/finish-times of the activities. Initial duration constraints are not updated although they could be, in order to extract more information on the solution feasibility.

The difference becomes quite clear when one accepts that a graph-based constraint propagation algorithm aims essentially at enforcing *consistency* between constraints, which

can affect both node labels (such as time bounds on events) and arc labels (duration constraints or distance constraints between two events). This idea is central in the work presented below.

Constraint networks represent a general framework for researches in temporal Constraint Satisfaction Problems (CSPs) (Montanari, 1974), (Dechter et al., 1991). In a temporal constraint network, each node i represents a variable t_i and each arc represents a double-constrained distance between two nodes. For instance, an arc from t_i to t_j valuated by $[\underline{a}_{ij}, \overline{a}_{ij}]$ stands for the constraint : $\underline{a}_{ij} \leq t_j - t_i \leq \overline{a}_{ij}$. When applied to scheduling problems, it allows the modeling of variable durations and some particular resource allocation constraints, but not sharing constraints. Moreover, this model only considers temporal consequences of the resource assignment problem. For example, let task A processed either on machine 1 with a duration included in [4,6], or on machine 2 with a duration included in [5,7]. In a constraint network, the duration of A is then included in [4,7]. Thus, the machine dependency is lost.

Here our goal is to be able to represent, resource sharing and allocation constraints, and processing times that are not well-known, in order to better cover real-life constraints. Thus we introduce the *Time-Bound-On-Node* (*TBON*) graph. The main feature of this graph is that it associates two nodes with each task. These nodes represent the time bounds of the task (start- and finish-times). For us, this graph is the most general representation for processing time constraints in scheduling problems : it combines the advantages of activity networks (two nodes for each task permits the modeling of any distance constraints between two events) and the deductive power of constraint networks (both node and arc labels are updated).

2.3. *The TBON graph*

Given the definition of a graph (Carré, 1979, §II-2-2) and a set of n tasks, a TBON graph $G(X,A)$ consists of :

- a finite set $X = \{x / x=1,...,2n\}$ whose elements are called *nodes*. To each node x is associated a time bound B_x which represent either the start- or the finish-time of a task. Each bound B_x is labelled with an interval $[\underline{B}_x, \overline{B}_x]$ where $\underline{B}_x$ and $\overline{B}_x$ define the minimum and the maximum values of time bound B_x, respectively ;
- a subset A of the Cartesian product $X \times X$ the elements of which are called *arcs*. An arc (x, y) with weight a_{xy} belongs to A if the potential inequality $B_y - B_x \geq a_{xy}$ is a constraint in the problem formulation.

Conventions :

- As in (Elmaghraby and Kamburowski, 1992), task i (i = 1,...,n) will have its start-time S_i numbered 2i-1 and its finish-time F_i numbered 2i. Let *number*(B) be this numbering function, where B stands for either S_i or F_i. For each node x, the inverse function *number*$^{-1}$(x) (x = 0,...,2n) is defined by :

  - if x = 0, number^{-1}(x) = 0 ;

  - if x is odd, number^{-1}(x) = $S_{(x+1)/2}$;

  - if x is even, number^{-1}(x) = $F_{x/2}$.

- For the clarity, a node will be referred to as its number or its time bound, thanks to number and number^{-1} functions.
- The absence of an arc is equivalent to an arc whose weight is -∞ (in practice, ∞ is some very large number).
- The absence of a label of an arc means that its corresponding weight is zero.

Note that the constraint $\underline{S}_i \leq S_i \leq \overline{S}_i$ might be re-written as the pair of potential inequalities : $S_i - 0 \geq \underline{S}_i$ and $0 - S_i \geq -\overline{S}_i$. Therefore we assume that the following representations are fully equivalent (see Fig.1) :

- A node B_X is labelled with a time window $[\underline{B}_X, \overline{B}_X]$ (Rep.1) ;

- There exists an arc with weight $\underline{B}_X$ from the origin node to B_X and an arc from B_X to the origin node with weight $-\overline{B}_X$ (Rep.2). This representation is called the *O-graph*.

The TBON representation is close to conventional ones, mostly because of the readability of information. 2n nodes are needed. Nevertheless, it then exhibits a dissimilarity in the constraint modeling : unary constraints are associated with nodes and binary constraints with arcs. From a computational view-point, constraint representation in an O-graph is more homogeneous and needs less data structures because all kinds of constraints are associated with arcs. Conversely, the number of arcs is multiplied by 4n (two arcs per bound, one for the minimum value and one for the maximum value). 2n+1 nodes are needed.

Usually, since activity networks address the field of project planning, for which one main goal is to minimize the total project duration, the network mentions both a node *origin* and a node *end*. We could imagine TBON graph with these two particular nodes (it could be named an "OE-graph"). In fact, in activity networks, if the origin-node is labelled by 0, the end-node is labelled with the total duration ; let F* denote it. When F* is fixed as a deadline, every possible ending activity i has its finish-node connected to the project end-node by an arc which states that the activity must end before the deadline : $F^* - F_i \geq 0$. It is easy to show that such a

constraint can also be expressed by an arc between the finish-node and the project origin-node, just by rewriting this inequality : $0 - F_i \geq - F^*$.

The same idea is also applicable if the activity networks fail to mention any finish-node (because durations are known). Then the inequality is rewritten : $0 - S_i \geq - F^* + D_i$, which is denoted by one arc from the start-node to the project origin-node.

Finally, whatever the original formulation of the constraints on the ending of tasks, i.e., either fixed global deadline or separate due dates, an O-graph is sufficient to take them into account. The use of a project end-node is certainly more common in the project planning area, but we do not want to address this sole class of task scheduling problems.

When considering scheduling problems without initial deadline constraints, it is usual to propose one arbitrary bound, e.g. an upper bound computed by a fast queuing algorithm, or *greedy* algorithm (Carlier and Pinson, 1989). The TBON graph remains applicable if this bound can still be assumed as an absolute "limit" (any lateness is forbidden).

For the remainder of the paper, the TBON representation is considered for modeling and the O-graph for implementation purposes (§3 and 4).

2.4. *Conjunctive and non conjunctive TBON graphs*

An inference process based on constraint propagation through a TBON graph may be used to fully characterize the admissible schedules, by only considering the time constraints which can be expressed as a conjunctive set of potential inequalities. On the other hand, as soon as constraints need the use of disjunctive sets of potential inequalities, the problem complexity increases, due to the combinatorial feature of these constraints. It is now necessary to present both conjunctive and non conjunctive parts of a TBON graph for modeling scheduling problems under resource constraints.

2.4.1. *Conjunctive part*

In the *conjunctive part* of a graph, denoted by A_C, all the constraints must be simultaneously satisfied. We denote by *satisfy-arc*(x,y), the elementary constraint satisfaction function of the potential inequality associated with arc (x,y) : satisfy-arc(x,y) is true iff $B_y - B_x \geq a_{xy}$. The overall constraint satisfaction function on A_C, named *satisfy*(A_C), can be written as :

$$\text{satisfy}(A_C) = \bigcap_{\substack{x,y=1 \\ x \neq y}}^{2n} \text{satisfy-arc}(x,y)$$

where $\bigcap$ stands for the logical AND operator.

A candidate solution to the problem is an assignment of bounds B_X such that each constraint must be verified. For example, consider three tasks i, j, k with the precedence constraint : both i and j must be completed before k can be initiated. The modeling is as in Fig.2.

2.4.2. *Non conjunctive part*

The *non conjunctive part* (Roy, 1970, §VIII-B-3), denoted by A_{NC}, consists of disjunctive sets of constraints. Let NCl be one of these disjunctive sets, and *satisfy-disj*(NCl), the logical satisfaction function of NCl. This function is true if at least one of the potential inequality of NCl is true :

$$\text{satisfy-disj(NCl)} = \bigcup_{(x,y)\in NC_l} \text{satisfy-arc(x,y)} \qquad (1)$$

where $\bigcup$ stands for the logical OR operator.

Thus the global satisfaction of A_{NC} is :

$$\text{satisfy}(A_{NC}) = \bigcap_{l=1}^{r} \text{satisfy-disj(NCl)} \qquad \text{with } r = |A_{NC}|.$$

A candidate solution is an assignment of all bounds such that each disjunctive set of constraints is satisfied.

Note that A_{NC} may possess several arcs in parallel between two nodes x and y, each instance belonging to one different disjunctive set. Therefore, in order to distinguish parallel arcs, labels must include both a weight and the name of the disjunctive set the arc belongs to. For instance, label "b,NC_1" in arc (x,y) states that the arc is valuated by b and is included in disjunctive set NC_1.

Note also that each disjunctive set of arcs NCl may contain one or more occurrences of arcs (x,y), but when two arcs link the same nodes in the same direction (multi-digraph), they must have two different valuations. Thus, in order to fully characterize one arc of a disjunctive set, both the pair of nodes (x,y) and its valuation b will be specified, yielding arc (x,y,b).

The form (1), called *standard form*, allows the representation of the following problems :

* *Resource sharing problem*

For instance, consider the example illustrated in Fig.3 (Rep.1) : three tasks i, j, k share the same machine. Label ",NC_1" in arcs (F_j,S_i) and (F_i,S_j) states that these arcs are included in disjunctive set NC_1 (duration is 0).Then we have to solve a conflict, sequencing i before j or conversely. The same reasoning is applied to i and k, and j and k. The conflict sets are : {i,j}, {i,k}, {j,k}. It yields the following disjunctive sets :

$(S_i - F_j \geq 0$ OR $S_j - F_i \geq 0)$ AND $(S_i - F_k \geq 0$ OR $S_k - F_i \geq 0)$

AND $(S_k - F_j \geq 0$ OR $S_j - F_k \geq 0)$

In Rep.2, we consider also these three tasks, but now sharing a resource available in an amount of two (cumulative problem). The conflict set is : {i,j,k}, yielding the following disjunctions :

$S_i - F_j \geq 0$ OR $S_j - F_i \geq 0$ OR $S_i - F_k \geq 0$ OR $S_k - F_i \geq 0$ OR $S_k - F_j \geq 0$ OR $S_j - F_k \geq 0$

* *Allocation-dependent transportation times*

Task i precedes task j. If i and j are processed on the same machine, there is no slack period between the finish of i and the start of j. In turn, if i and j use a different machine, this slack period is equal to α, in order to take account of a transportation time between the two machines. The disjunction is : $S_j - F_i \geq 0$ OR $S_j - F_i \geq \alpha$ (see Fig.4).

However, some temporal constraints are not directly modeled using (1). For example :

* *Resource allocation*

Task i can be processed either on Machine1 with duration D_i^1 in $[\underline{D}_i^1, \overline{D}_i^1]$, or on Machine2 with duration D_i^2 in $[\underline{D}_i^2, \overline{D}_i^2]$. The disjunctive set of constraints is :

$(F_i - S_i \geq \underline{D}_i^1$ AND $S_i - F_i \geq - \overline{D}_i^1)$ OR $(F_i - S_i \geq \underline{D}_i^2$ AND $S_i - F_i \geq - \overline{D}_i^2)$.

* *Calendar*

Task i starts either between $\underline{S}_i^1$ and $\overline{S}_i^1$ or between $\underline{S}_i^2$ and $\overline{S}_i^2$, which is to be written as follows :

$(S_i - 0 \geq \underline{S}_i^1$ AND $0 - S_i \geq - \overline{S}_i^1)$ OR $(S_i - 0 \geq \underline{S}_i^2$ AND $0 - S_i \geq - \overline{S}_i^2)$.

For the two previous constraints, the disjunctive sets cannot be simply represented on a TBON graph. On behalf of conversion rules, they may be transformed into sets of the standard form. Thus, for the resource allocation modeling, the disjunctive set of constraints :

$(F_i - S_i \geq \underline{D}_i^1$ AND $S_i - F_i \geq - \overline{D}_i^1)$ OR $(F_i - S_i \geq \underline{D}_i^2$ AND $S_i - F_i \geq - \overline{D}_i^2)$

is then equivalent to four disjunctions in the standard form (represented in Fig.5) :

$(F_i - S_i \geq \underline{D}_i^1$ OR $F_i - S_i \geq \underline{D}_i^2)$ AND $(F_i - S_i \geq \underline{D}_i^1$ OR $S_i - F_i \geq - \overline{D}_i^2)$ AND

$(S_i - F_i \geq - \overline{D}_i^1$ OR $F_i - S_i \geq \underline{D}_i^2)$ AND $(S_i - F_i \geq - \overline{D}_i^1$ OR $S_i - F_i \geq - \overline{D}_i^2)$.

Remark : During this stage of transformation, it is possible to simplify some disjunctive sets, with arcs in parallel. For example, as indicated in Fig.6 :

$$(F_i - S_i \geq \underline{D}_i^1 \text{ OR } F_i - S_i \geq \underline{D}_i^2) \Rightarrow F_i - S_i \geq \min(\underline{D}_i^1, \underline{D}_i^2) ;$$

$$(S_i - F_i \geq -\overline{D}_i^1 \text{ OR } S_i - F_i \geq -\overline{D}_i^2) \Rightarrow S_i - F_i \geq -\max(\overline{D}_i^1, \overline{D}_i^2).$$

3. Constraint propagation through a conjunctive TBON graph

3.1. *Introduction*

In this section, we present a logical process called *constraint propagation* capable of reaching the goal of characterization of admissible schedules. This process progressively transforms the initial formulation of the problem into an equivalent one, but whose search space of feasible solutions is more restricted.

The constraint propagation procedure is a deductive mechanism ; it combines constraints, and then expresses more restrictive characteristics of feasible solutions. It also reveals possible inconsistencies in the problem formulation. This procedure consists in suppressing from the search space of admissible solutions, the incompatible values of variables facing the constraints. It is based on the detection of the longest paths in a graph (Gondran and Minoux, 1984).

3.2. *Updating time constraints*

As indicated in §2, the time constraints on a TBON graph are represented by time windows associated with nodes and by weights on the arcs. To characterize the feasible solutions, our constraint propagation process leads to updating both node and arc valuations on a TBON graph.

The admissible schedules are then characterized not only by a node label (such as activity-on-arc and activity-on-node graphs) but also by the set of potential constraints, like duration or sequencing constraints. In scheduling problems, it is a new way of considering constraint propagation.

3.3. *Propagation algorithm*

The propagation algorithm is based on the search for the longest paths in a graph : Floyd-Warshall's all-pairs-longest-path Algorithm (FWA) is used. It computes the longest path between each couple of nodes and also detects positive circuits so as to reveal inconsistencies (Dreyfus, 1969), (Carré, 1979, Chapter III, page 122). FWA is the following :

```
for z from 0 to 2n
  for x from 0 to 2n
    if a_XZ + a_ZX > 0 then
      exit                                  /* positive circuit detected */
    else
      if a_XZ ≠ - ∞ then
        for y from 0 to 2n
          a_XY := Max (a_XZ + a_ZY, a_XY)   /* arc valuation updating */
        endfor
      endif
    endif
  endfor
endfor
```

with :

- n : number of tasks (2n+1 nodes in the graph)
- a_{XY} : valuation of arc (x,y) ; it corresponds to the potential inequality : $B_Y - B_X \geq a_{XY}$.

To apply this algorithm, all constraints have to be represented as potential inequalities. The processing of constraints can be then simplified by using an O-graph. The algorithm replaces initial value a_{XY} with the value of the longest path from x to y, even if there is no explicit constraint in the initial formulation. The derived graph is a *complete* graph, namely, a graph in which each couple of nodes is connected by an arc.

In simple cases, like conjunctive graphs, FWA computes the complete graph with the tightest constraints. It fully characterizes the admissible schedules from the initial formulation with an $O(n^3)$ computational complexity.

4. Constraint propagation through a non conjunctive TBON graph

4.1. *Introduction*

In order to tackle sharing resource constraints, one can analyze the conflicts associated with tasks using the same resource and then to isolate conflicting sets of tasks. Such an approach is used in (Erschler et al., 1979), where an iterative procedure is applied to simplify the disjunctive sets of arcs which represent the conflicts ; sets reduced to a singleton can be added to the conjunctive graph, which may restart propagations in the conjunctive part of the graph. This procedure is sound, complete, but of exponential complexity. To prevent this drawback, one way is to give up the completeness, not realistic in large scale problems (Esquirol, 1987), (Lopez, 1991).

Considering non conjunctive graphs and continuous domains in the CSP area, certain authors have proposed algorithms for the processing of some particular disjunctions (resource allocation) which do terminate but which may fail to detect any inconsistencies (if any) (Dechter et al., 1991).

We give below a procedure which uses FWA and a simplification procedure of the non conjunctive part of the graph. Obviously, it is not complete anymore, but yet it may be useful for solving disjunctions in some scheduling problems. Remark that we will not consider a particular form of disjunctions since we have shown (§2.4.2) that in a TBON graph, arcs of a disjunctive set may not necessarily connect two common nodes.

4.2. *Algorithm*

Remind that A_C is the conjunctive part of the graph, A_{NC} is the superset of the disjunctive sets of arcs $\{NC_l, l = 1,...,r\}$: $A_{NC} = NC_1 \cup NC_2 \cup ... \cup NC_r$, with $A_C \cup A_{NC} = A$.

The algorithm we propose below applies FWA each time an arc can be added to A_C. The rules are the following :

Rule 1 : simplification by deletion of inconsistencies.

If an arc (x,y,b) of a disjunctive set NC_l is such that $a_{yx} + b > 0$, a circuit is detected which means that (x,y,b) is not consistent with A_C ; therefore it must be removed from NC_l.

Let SIMPLIFY_DISJ(x,y,b,NC_l) be the procedure which removes arc (x,y,b) from the disjunctive set NC_l.

Rule 2 : deletion of dominated constraints.

If an arc (x,y,b) of a disjunctive set NC_l is such that $a_{xy} \geq b$, one constraint of NC_l is dominated by arc (x,y) of A_C and NC_l is satisfied. Therefore NC_l can be removed from A_{NC}.

Let REMOVE_DISJ(NCl,A_{NC}) be the procedure which removes the disjunctive set NCl of A_{NC}.

Rule 3 : propagation.

If a disjunctive set NCl is a singleton {(x,y,b)}, it must be added to A_C ; in this case, it possibly restarts the propagation through the conjunctive graph and FWA must be reapplied.

Let FW(A_C) be the function which returns the result following application of FWA to the conjunctive graph G(X,A_C).

---

repeat
  again ← *false*
  A_C ← FW(A_C)
  for each NCl ∈ A_{NC}
    for each (x,y,b) ∈ NCl
      if b + a_{yx} > 0 then
        SIMPLIFY_DISJ(x,y,b,NCl)
        if |NCl| = 1 then
          again ← *true*
          A_C ← A_C + NCl
        endif
      endif
      if a_{xy} ≥ b then
        REMOVE_DISJ(NCl, A_{NC})
      endif
    endfor
  endfor
until again = *false*

The worst-case complexity of this algorithm is O(n^3rq), where r is the number of disjunctive sets of A_{NC}, and q is the maximum range over all disjunctions. This evaluation is a very pessimistic upper bound since : if the non-conjunctive procedure does not cause anything, the main loop will be achieved in a single pass and the global complexity remains that of FWA ; if it is efficient, numerous simplifications will decrease r and/or q.

Remark : Deleting one arc in some disjunctive subsets must be applied simultaneously to all subsets to which this arc belongs. This is why we need to manage both forward and backward links between arcs and subsets. Thus in practice, *while..endwhile* control loops are implemented in place of *for..endfor* loops, in order to speed up the algorithm.

5. Example

By way of example, consider a job-shop scheduling problem. The elements are a set of machines {1,2,3} and a set of jobs {1,2,3,4} to be scheduled. Each job i consists of two operations, (i,1) and (i,2). The problem has then eight operations. This problem is represented graphically via the TBON graph in Fig.7. Each operation processes a set of identical parts. A schedule is a feasible resolution of the resource constraints when no two operations ever occupy the same machine simultaneously.

Machines 1 and 3 are devoted to milling. Suppose for example that it is possible to cut simultaneously two sheets of metal by superposing them. Processing times on machines 1 and 3 may then be divided by two : as a result, two minimum durations are associated with each operation using one of these machines.

A tabular representation of the data is shown in Table 1 which lists the operation processing times, earliest start and due date constraints and the operation machine assignments (routing).

Table 2 displays the results of our algorithm which leads to a conjunctive graph (an initial endpoint is associated with a row and a terminal endpoint with a column).

All disjunctions derived from machine use (eight initially) are removed. The operation sequences are :

machine 1 : $11 \rightarrow 21$
machine 2 : $41 \rightarrow 31 \rightarrow 12 \rightarrow 22$
machine 3 : $42 \rightarrow 32$

The maximum duration of operations 11, 21, 32 and 42 is 8, 7, 5 and 6 respectively, thus preventing a minimum duration of 14, 12, 8 and 8, respectively.

6. Computational results

The preceding algorithm has been programmed in C language. The executable code represents about 32 ko and has been tested on a Sun Sparc Server 470, running under UNIX operating system. Considering the different algorithm loops, three characteristics may impact CPU time :

- n, the number of tasks (or, to be more accurate, K = 2n+1, number of nodes in the graph) ;
- r, the maximum number of disjunctive sets of arcs ;
- q, the maximum number of arcs per disjunction.

To evaluate the efficacy of this algorithm and to easily check the propagation results, we built several samples as follows :

First, two examples of small dimensions have been tested :

- example 1. the job-shop problem referred to in §5 (K = 17 ; r = 12 ; q = 2),
- example 2. the resource-constrained project scheduling problem considered in (Erschler et al., 1979) and described in Table 3 (K = 17 ; r = 3 ; q = 6) ;

Then increasingly large examples are generated by duplication of the previous basic examples (number of "occurrences" in Table 4).

Since the graph explored by the algorithm is complete, the complexity really increases in $O(n^3)$. Because of the block-diagonal form of the arc-matrix after duplication, the results of the propagation process remain easy to check.

In Table 4, CPU times are given for both examples duplicated from 1 to 150. Amongst the different cases, the maximum number of nodes is 3000 before memory overflow. The last column lists the percentage of disjunction resolution ($\delta = 1$ when none are left).

7. Conclusion

In this paper, the efficiency of the TBON graph has been shown for representing various types of constraints in the scheduling field (duration constraints, modeling of both assignment and sequencing problems, ...). It may be regarded as a generalization of most models proposed for addressing scheduling problems, since it combines the concepts of activity-on-arc and activity-on-node.

Given the conjunctive problems, the well-known Floyd-Warshall's algorithm, sound and complete, allows propagation over all constraints and characterizes the feasible schedules in polynomial time.

In the disjunctive case, an algorithm provides answers to the schedule feasibility. It is based on the same structure as the previous one ; although it is not complete, its complexity is still polynomial and can offer a good trade-off in complex problems, for the characterization of feasible schedules.

In any case, this characterization step can be employed to reduce the search space of a solution generator based on heuristic rules or optimization procedures (Bel et al., 1989).

Acknowledgements

The authors are indebted to an anonymous referee whose comments and suggestions improved this article.

References

Amamou, M., Happiette, M. and Staroswiecki, M. (1992), Decomposition of the single machine scheduling based on the notion of semi-rigid sub-sequences, in *Proceedings of International Conference on Automation Technology*, Taiwan, July.

Bartusch, M., Möhring, R.H. and Radermacher, F.J. (1988), Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, **16**, 201-240.

Bel, G., Bensana, E., Dubois, D., Erschler, J. and Esquirol, P. (1989), A knowledge based approach to industrial job-shop scheduling. In *Knowledge-based systems in manufacturing*, A. Kusiak (Ed.), Taylor & Francis, 207-246.

Bellman, R.E. (1958), On a routing part problem. *Quart. Appl. Math.*, **16**, 87-90.

Bellman, R., Esogbue, A.O. and Nabeshima, I. (1982), *Mathematical aspects of scheduling and applications* ; Pergamon Press, Oxford, (Chapter XII : "The Job-shop scheduling problem").

Carlier, J. and Pinson, E. (1989), An algorithm for solving the Job-shop problem. *Management Science*, **35**, 164-176.

Carré, B. (1979), *Graphs and networks* ; Clarendon Press, Oxford.

Dechter, R., Meiri, I. and Pearl J. (1991), Temporal constraint networks. *Artificial Intelligence*, **49**, 61-95.

Dibon, M. (1970), *Ordonnancement et potentiels / Méthode M.P.M.* ; Hermann, Paris.

Dreyfus, S.E. (1969), An appraisal of some shortest-path algorithms. *Operations Research*, **17**, 395-412.

Elmaghraby, S.E. (1977), *Activity networks : Project planning and control by network models* ; John Wiley & Sons, New-York.

Elmaghraby, S.E. and Kamburowski, J. (1992), The analysis of activity networks under generalized precedence relations (GPRs). *Management Science*, **38**, 1245-1263.

Erschler, J., Fontan, G. and Roubellat, F. (1979), Potentiels sur un graphe non conjonctif et analyse d'un problème d'ordonnancement à moyens limités. *RAIRO Recherche Opérationnelle*, **13**, 363-378.

Esquirol, P. (1987), Règles et processus d'inférence pour l'aide à l'ordonnancement de tâches en présence de contraintes, Thèse de Doctorat de l'Université Paul Sabatier, Toulouse.

Gondran, M. and Minoux, M. (1984), *Graphes et algorithmes* ; Eyrolles, Paris, (Chapitre II : "Le problème du plus court chemin").

GOThA (1993), *Les problèmes d'ordonnancement*. RAIRO Recherche Opérationnelle, **27**, 77-150.

Graves, S.C. (1981), A review of production scheduling. *Operations Research*, **29**, 646-675.

Herroelen, W.S. and Demeulemeester, E. (1992), Recent advances in Branch-and-Bound procedures for resource-constrained project scheduling problems, *Summer School on Scheduling Theory and Its Applications*, 249-279, Château de Bonas, France, September.

Kusiak, A. (1989), *Knowledge-based systems in manufacturing* ; Ed., Taylor & Francis, New-York.

Lopez, P. (1991), Approche énergétique pour l'ordonnancement de tâches sous contraintes de temps et de ressources, Thèse de Doctorat de l'Université Paul Sabatier, Toulouse.

Montanari, U. (1974), Networks of constraints : Fundamental properties and applications to picture processing. *Information sciences*, **7**.

Roy, B. (1970), *Algèbre moderne et théorie des graphes* ; Dunod, Paris, (tome II).

TABLES

		$\underline{S}_{ij}$		$\bar{F}_{ij}$		$\underline{D}_{ij}$		routing	
Operation		1	2	1	2	1	2	1	2
	1	3	-	-	16	7 or 14	3	1	2
Job	2	3	-	-	18	6 or 12	2	1	2
	3	1	-	-	16	5	4 or 8	2	3
	4	0	-	-	12	6	4 or 8	2	3

Table 1. A job-shop problem

\	0	S_{11}	F_{11}	S_{12}	F_{12}	S_{21}	F_{21}	S_{22}	F_{22}	S_{31}	F_{31}	S_{32}	F_{32}	S_{41}	F_{41}	S_{42}	F_{42}
0	0	2	9	11	14	9	15	15	17	6	11	11	15	0	6	6	10
S_{11}	-3	0	7	8	11	7	13	13	15	3	8	8	12	-3	3	3	7
F_{11}	-10	-8	0	1	4	0	6	6	8	-4	1	1	5	-10	-4	-4	0
S_{12}	-13	-11	-4	0	3	-4	2	3	5	-7	-2	-2	2	-13	-7	-7	-3
F_{12}	-16	-14	-7	-5	0	-7	-1	0	2	-10	-5	-5	-1	-16	-10	-10	-6
S_{21}	-10	-8	-1	1	4	0	6	6	8	-4	1	1	5	-10	-4	-4	0
F_{21}	-16	-14	-7	-5	-2	-7	0	0	2	-10	-5	-5	-1	-16	-10	-10	-6
S_{22}	-16	-14	-7	-5	-2	-7	-1	0	2	-10	-5	-5	-1	-16	-10	-10	-6
F_{22}	-18	-16	-9	-7	-4	-9	-3	-3	0	-12	-7	-7	-3	-18	-12	-12	-8
S_{31}	-7	-5	2	5	8	2	8	8	10	0	5	5	9	-7	-1	-1	3
F_{31}	-12	-10	-3	0	3	-3	3	3	5	-6	0	0	4	-12	-6	-6	-2
S_{32}	-12	-10	-3	-1	2	-3	3	3	5	-6	-1	0	4	-12	-6	-6	-2
F_{32}	-16	-14	-7	-5	-2	-7	-1	-1	1	-10	-5	-5	0	-16	-10	-10	-6
S_{41}	-1	1	8	11	14	8	14	14	16	6	11	11	15	0	6	6	10
F_{41}	-7	-5	2	5	8	2	8	8	10	0	5	5	9	-7	0	0	4
S_{42}	-8	-6	1	3	6	1	7	7	9	-2	3	4	8	-8	-2	0	4
F_{42}	-12	-10	-3	-1	2	-3	3	3	5	-6	-1	0	4	-12	-6	-6	0

Table 2. Complete graph after constraint propagation

		$\underline{S}_{ij}$				$\overline{F}_{ij}$				$\underline{D}_{ij}$			
Operation		1	2	3	4	1	2	3	4	1	2	3	4
Job	1	2	-	-	-	-	-	-	13	2	4	4	3
	2	0	-	-	-	-	-	-	12	1	5	3	2

		resource consumptions							
		q_{ij}^{k1}				q_{ij}^{k2}			
Operation		1	2	3	4	1	2	3	4
Job	1	2	2	3	2	1	1	4	2
	2	2	4	1	1	1	2	4	7

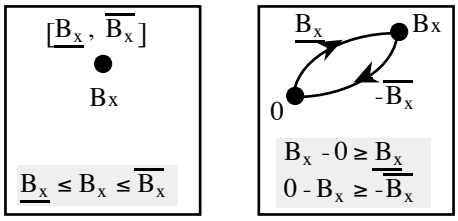
resource capacity	
q^{k1}	q^{k2}
8	9

Table 3. A resource-constrained project scheduling problem

example	occurrences	K	r	q	CPU sec.	δ
1	1	17	12	2	0	1
	10	161	120	2	5.6	
	50	801	600	2	125.2	
	100	1601	1200	2	538.4	
	150	2401	1800	2	1184.5	
2	1	17	3	6	0	0.9
	10	161	30	6	2.3	
	50	801	150	6	54.1	
	100	1601	300	6	243.2	
	150	2401	450	6	526.8	

Table 4. Experimental results

FIGURES



Rep.1 Rep.2

Figure 1.

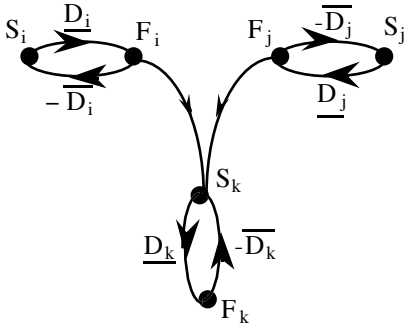
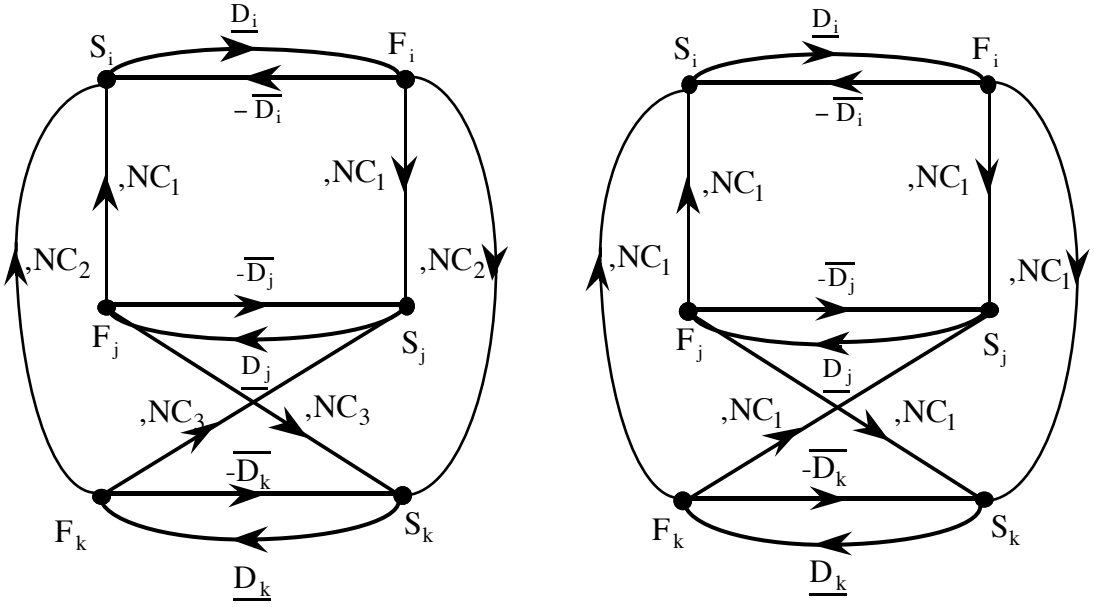


Figure 2.



Rep.1 Rep.2

Figure 3.

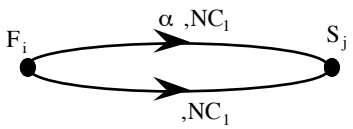


Figure 4.

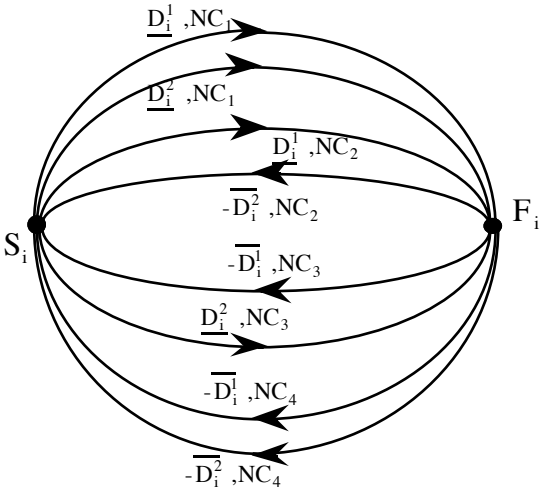


Figure 5.

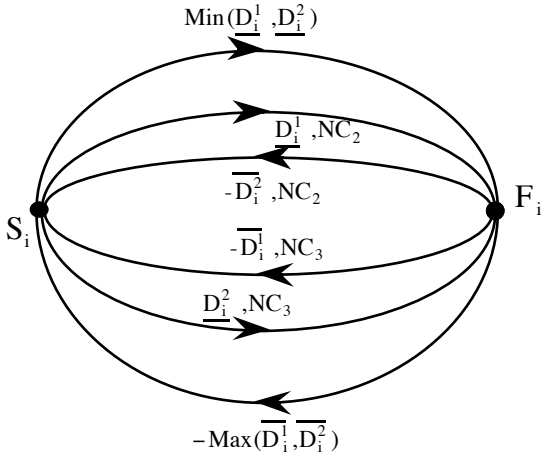


Figure 6.

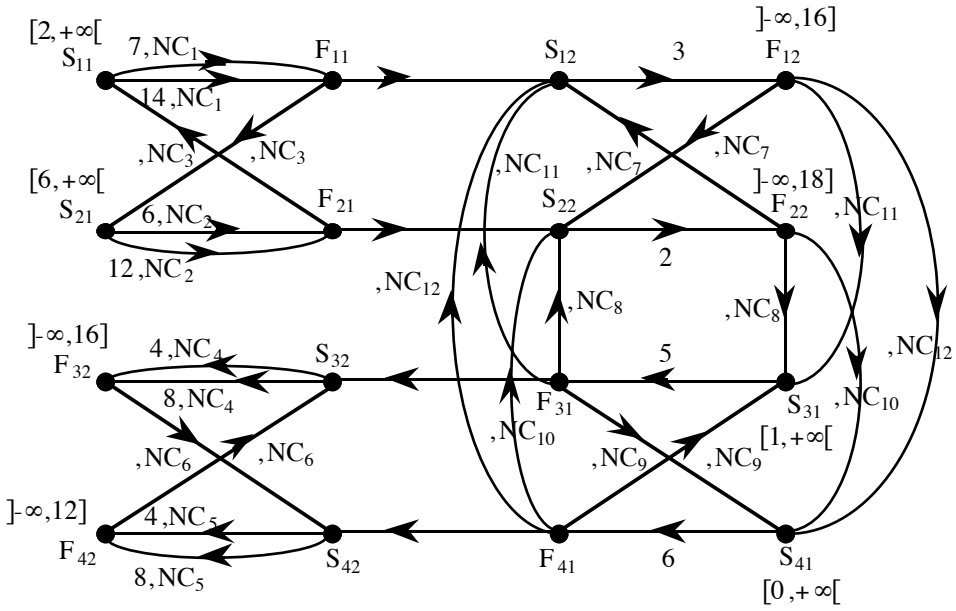


Figure 7.

CAPTIONS TO FIGURES

Figure 1. TBON vs. O-graph
Figure 2. Conjunctive TBON graph
Figure 3. Resource sharing
Figure 4. Transportation times
Figure 5. Resource assignment
Figure 6. Simplification
Figure 7. Job-shop problem represented via a TBON graph