Re-planning support system for make-to-order production with reserved resources

Caroline Thierry, Jacques Lamothe, V Galvagnon

Re-planning support system for make-to-order production with reserved resources

C. THIERRY†‡*, J. LAMOTHE§ and V. GALVAGNON†¶

Make-to-order production is often organized into projects with their own objectives. Such projects use strategic and critical resources that must be reserved at a strategic or a tactical planning level. When external resources need to be taken into consideration, they must be reserved based on their set of free time windows. A medium-term planning support system for the manager of a project where external resources are critical is proposed. By providing explanations for project inconsistencies, this system enables the decision-maker to restore consistency. After creating the initial medium-term plan, planning activity essentially consists in updating the existing schedule of a project when unexpected events arise. These may be due to internal or external disturbances. Indicators are proposed to help the manager follow the dynamic impact the disturbances will have on the project.

1. Introduction

Production of goods in manufacturing companies is traditionally controlled by long-, medium- and short-term planning. Medium-term planning of batch production is focused on optimization of the manufacturing process in terms of resources and amounts produced. In make-to-order production, the complexity required in the production of innovative or highly technical products demand means that optimization be managed through a project structure. In addition, a project often shares renewable resources with other projects of the same or other firms. These may be external to the firm and dependence on them means the project is subject to its 'environment' (suppliers, customers, subcontractors, other projects).

This paper focuses on medium-term project planning, targeting make-to-order production that uses subcontracted physical, renewable resources (figure 1). Such external resources are so critical that their unavailability has a significant impact both on project duration and organization. They are shared between projects originating in different firms and must be reserved based on the information communicated by the subcontractors. This information is a set of free time windows.

†Office National d'Études et de Recherches Aérospatiales, BP 4025, F-31055 Toulouse Cedex 4, France.

‡Université Toulouse II le Mirail, F-31058 Toulouse, cedex 9, France.

§Ecole des Mines d'Albi Carmaux, Centre de Génie Industriel, Campus Jarlard, F-81013 Albi CT Cedex 09, France.

¶Access Commerce, Le stratège, Bât B2, rue Ampère BP 555, F-31674 Labège Cedex, France.

*To whom correspondence should be addressed. E-mail: thierry@univ-tlse2.fr

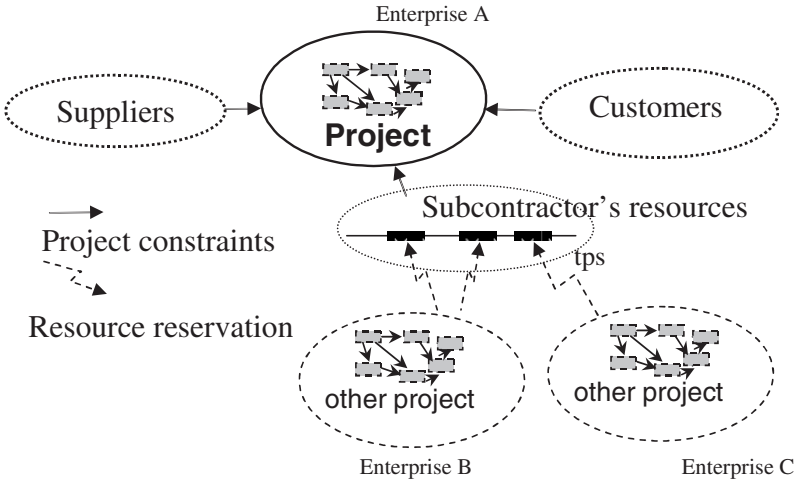


Figure 1. Resource constraints.

At this stage of planning, examining a given project work–breakdown–structure will enable identification of several possibilities of task sequencing. Therefore, a manager selects an initial partial sequence that can be changed during the project life. Moreover, the tasks considered at this planning level are aggregated ones. Therefore, within a given project, if several aggregated tasks use the same resource, we consider that they have been sequenced into the initial partial sequence.

Consequently, a project manager has to schedule the tasks of a project according to the time slots available for the external resources. This particular project-scheduling problem is referred to as the reserved resources project scheduling problem (RRPSP).

Once the project is underway, the planning function regularly reschedules the project taking into account events that occur. In such circumstances, a variety of responses can be considered: modifying the internal resource allocation or the initial partial sequence of tasks, negotiating new time windows for the external resources or new delivery dates from suppliers or new due dates. Practically, a manager may not readily define the set of possible actions to be taken. However, by analysing the circumstances, he/she will certainly propose some measures to be considered.

The paper is organized as follows. In section 2, a brief literature overview is provided. Section 3 proposes a re-planning support system (figure 2) for project managers facing the RRPSP (section 3.1). This system concentrates mainly on the case where no solution to the scheduling problem exists. The objective is to provide the user with a suitable explanation for the inconsistency so that the problem can be modified and a solution found. First, problem inconsistency and the causes of this inconsistency are detected (section 3.2). Indicators are then proposed (section 3.3) to help the decision-maker in the restoration process. Hence, the re-planning process leads to a redefinition of the problem constraints themselves. It will be up to the user to decide on any modification to the production process or to negotiate with the various actors in the project environment using the explanations provided (section 3.4). Industrial experience is reported in section 4. Finally, section 5 is reserved for conclusions.

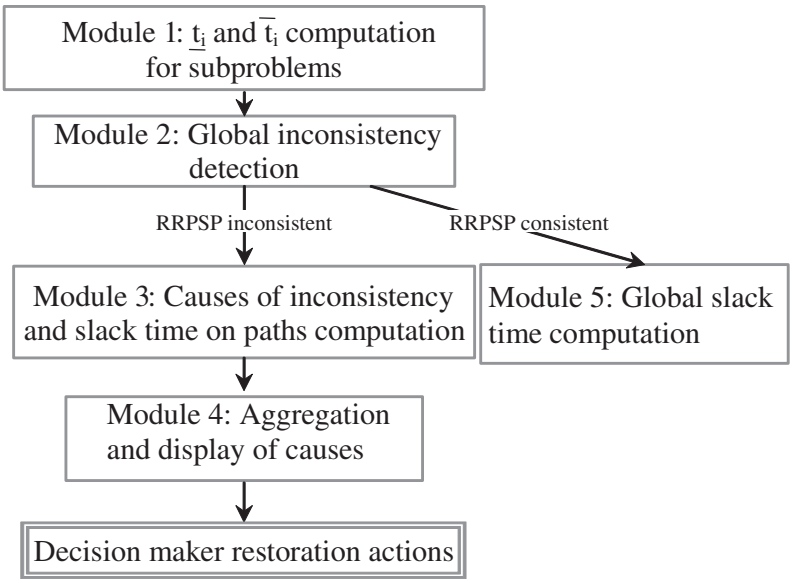


Figure 2. Structure of the re-planning support system.

2. Literature review

In make-to-order production, project scheduling models and methods have been used extensively (Kolisch 2001). In the field of project scheduling techniques (e.g. Kelley 1961, Kolisch and Padman 2001, Neumann *et al.* 2003), numerous studies (e.g. Brucker *et al.* 1999, Weglarz 1999) have focussed on the management of renewable resources that may be used or even shared by projects (resource-constrained project scheduling problem, i.e. RCPSP, and resource-constrained multi-project scheduling problem, i.e. RCMPSP). In the RCMPSP context, priority rules-based scheduling approaches are used and adapted to a multi-project perspective (Kurtulus and Narula 1985, Speranza and Vercellis 1993, Yang and Sum 1997). When various alternatives for the allocation of internal resources exist, the multi-mode resource scheduling problem is relevant. Optimal procedures or heuristics (Boctor 1996, Elmaghraby 1977, Salewski *et al.* 1997) have been proposed to select the appropriate modes for a RCPSP problem. Other approaches suggest modelling the different alternatives of work breakdown structure on an activity network (Golenko-Ginsburg and Blokh 1997). They also adopt resolution through heuristics. In all of these cases, it is assumed that information on permissible changes to the project structure can be adequately expressed, which is not the case in the present study.

In the field of project scheduling with time windows, Chen *et al.* (1997) distinguish two types of time window constraints per activity: a single available continuous time or a series of admissible dates per activity. Then, forward and backward procedures are extended to compute earliest or latest starting dates and available margins. Moreover, Neumann *et al.* (2003) synthesize a great amount of work on resource constraint project scheduling with time windows and resources and develop the notion of a calendar. A calendar is a function $b: R^+ \rightarrow \{0; 1\}$ such that $b(t)$ expresses whether or not time t is available. Such a function can be applied to a task or to a time lag between two tasks in an 'Activity on Node' network. The authors propose a label-correcting algorithm that computes earliest and latest schedules. This algorithm does not converge if inconsistencies exist.

As a dynamic environment is considered, another approach referred to as 'reactive scheduling' consists in revising or re-optimizing a baseline schedule. Different approaches to this reactive scheduling are described in Leus (2003):

- 'Schedule repair action' techniques that focus on a quick schedule consistency restoration.
- '(Full) rescheduling' approaches that involve 'a full scheduling pass of that part of the project that remains to be executed at the time the reaction is initiated'.

As the present study is application oriented, the schedule repair action technique which is the most commonly used in practice has been adopted: we deal with repairing the existing predictive schedule in order to take into account the current state of the system. In this approach, when schedule breakage occurs, human schedulers use their experience to update information so that processing can continue. The problem being studied proposes a re-planning support system designed to take into consideration reserved resource constraints due to the multi-project and multi-enterprise context. This so-called reserved resources project scheduling problem (RRPSP) resembles the problem addressed in Chen *et al.* (1997) and the calendars of Neumann *et al.* (2003). However, the importance here is that an activity has to be performed in a set of time slots, whereas Chen *et al.* only consider a single time slot. Moreover, problem inconsistencies are detected and an attempt is made at explaining the sources of these inconsistencies.

3. Re-planning support system

3.1. *Reserved Resources Project Scheduling Problem (RRPSP)*

3.1.1. *RRPSP model*

Let us call I the number of tasks, T_0 an initial task and T_I a final task. The variables of the problem are the starting dates t_i of tasks T_i ($i \in \{1, 2, \ldots, I\}$). The parameters for a task T_i are: d_i, the duration of task T_i, A_i, the arrival date of supplies for task T_i defines the ready-date, and J_i, the due dates of task T_i. Parameter w_{ji} corresponds to the time that must elapse between the beginning of T_j and the beginning of T_i. In fact, w_{ji} is positive and generally equal to the duration d_j of task T_j (thus, the constraint implies that task T_i can start as soon as task T_j finishes) or equal to $-\infty$ (hence, the constraint between T_i and T_j is useless).

Moreover, if task T_i uses one or several external resources that must be reserved, the subcontractor communicates a set of possible time window reservations. This results in a set O_i of n_i unreserved independent time windows in which task T_i can be realized: $O_i = \bigcup_{x\in[1;n_i]}[S_x^i; E_x^i]$. The resources are considered free at infinite times so, $S_1^i = -\infty$ and $E_{n_i}^i = +\infty$. As pre-emption is not allowed, a domain D_i of authorized values is computed for each starting date t_i:

$$\forall t \in [t_i, t_i + d_i] \subset O_t \Leftrightarrow \forall it_i \in \bigcup_{x\in[1;\,n_i]} [S_x^i; E_x^i - d_i] \Leftrightarrow D_i = \bigcup_{x\in[1,n_i]} [S_x^i, E_x^i - d_i]$$
$$= \bigcup_{x\in[1,n_i]} [SD_x^i, ED_x^i].$$

The constraints that link variables comes from this problem definition:

$$t_0 = 0 \tag{0}$$
$$t_i \geq t_0 \qquad (i = 1, \ldots, I) \tag{1}$$
$$\forall i, \;\; t_i \geq A_i \qquad (i = 1, \ldots, I) \tag{2}$$

$$\forall i, j, \quad t_i - t_j \geq w_{ji} \quad (i = 1, \ldots, I, \ j = 1, \ldots, I) \tag{3}$$

$$\forall i, \quad t_i + d_i \leq J_i \quad (i = 1, \ldots, I) \tag{4}$$

$$\forall i, \quad t_i \in D_i \quad (i = 1, \ldots, I) \tag{5}$$

Constraints (0–4) are those typical of project-scheduling problems that do not take resource constraints into consideration: classical PERT (program evaluation and review technique) or CPM (critical path method) models.

The specific nature of the resource constraints results in the establishment of a set of admissible time windows for the starting time of tasks (5). These constraints are a generalization of the constraints of Chen *et al.* (1997). A domain D_i is a calendar associated to a task in the Neumann *et al.* (2003) model. Consequently, unlike RCPSP problems, disjunctive precedence constraints between activities need not be considered.

A graph of the problem is derived from a classical activity on node representation. The nodes depict the tasks. The directed arcs stand for the precedence (1, 3) or ready date constraints (2) between the tasks. Consequently, arc values, denoted by $a_{i,j}$, are positive. The unary constraints (attached to a single node) stand for the due-date constraints (4) or the resource time window constraints (5), which compel each task to use only one admissible time window.

3.1.2. *Extension of the notation*

Prec(j) will denote the set of predecessors to node (j), and Succ(i) the set of successors to node (i). However, a path on such a graph is more complex than a typical path in a PERT since unary constraints must be taken into consideration. Hence, we define a path from node (i) to node (j) denoted $P_\alpha(i,j)$ as follows:

- Chain α of arcs starting at node (i) and ending at node (j) as in a typical PERT graph.
- In addition, the unary constraints that deal with the nodes on the chain (due date, set of time windows...).

Finally, note that if the starting time t_i is fixed for a predecessor (i) of node (j) then

- one can compute a 'reaching date' at node (j) which is: $t_i + a_{i,j}$; and
- if this 'reaching date' does not belong to an admissible time window D_j, the resource time window constraint (5) forces the starting time at node (j) to be delayed to the first starting date SD^j_x greater than the 'reaching date'.

We thus define the minimal delay, called EarliestDelay, between (i) and (j) knowing that (i) starts at t_i. It sums up the constraints (0–3) and (5) between (i) and (j):

$$\text{EarliestDelay}(i, j, t_i) = \max\left[t_i + a_{i,j}, \min_{x \in [1, n_j]} (SD^j_x / ED^j_x \geq t_i + a_{i,j})\right] - t_i \tag{6}$$

Thus, the relation between node (j) and its predecessors becomes

$$t_i \leq \max_{i \in \text{Pred}(j)} (t_i + \text{EarliestDelay}(i, j, t_i)) \tag{7}$$

Note that because of a time delay in the starting date due to resource time windows (5), the EarliestDelay and, consequently, the length of a path depend on the date on which it is analysed.

Reciprocally, we consider that starting time t_j is fixed for a successor (j) of node (i). Then, the resource time window (5), the due date constraint (4) and the precedence constraints (1–3) are summed up into a minimal delay, called LatestDelay, which occurs between (i) and (j):

$$\text{LatestDelay}(i,j,t_j) = t_j - \min\left[(J_i - d_i), (t_j - a_{i,j}), \max_{x \in [1,n_i]} (ED_x^i / SD_x^i \leq t_j - a_{i,j})\right]. \tag{8}$$

Thus, the relation between node (i) and its successors becomes (9):

$$t_i \leq \min_{j \in \text{Succ}(i)} \left(t_j - \text{LatestDelay}(i,j,t_j)\right) \tag{9}$$

Since the length of the paths depend on the date on which they are analysed, Earliest and Latest path length can be introduced such that:

Earliest path length, $EP_\alpha(i,j,t_i)$ is the duration of the earliest project schedule along a path $P_\alpha(i,j)$ knowing that task T_i starts at time t_i. It is defined recursively using the EarliestDelay relation (6):

$$\left.\begin{array}{l} EP_\alpha(i,i,t_i) = 0 \ \text{ and} \\ EP_\alpha(i,j,t_i) = EP_\alpha(i,k,t_i) + \text{EarliestDelay}(k,j,t_i + EP_\alpha(i,k,t_i)) \\ \text{with} \ \ k = \alpha \cap \text{Pred}(j) \end{array}\right\}. \tag{10}$$

Latest path length, $LP_\alpha(i,j,t_i)$ is the duration of the latest project schedule along α knowing that task T_j starts at time t_j. It can also be defined recursively using relation (8):

$$\left.\begin{array}{l} LP_\alpha(j,j,t_j) = 0 \ \text{ and} \\ LP_\alpha(i,j,t_j) = LP_\alpha(k,j,t_j) + \text{LatestDelay}(i,k,t_j - LP_\alpha(k,j,t_j)) \\ \text{with} \ \ k = \alpha \cap \text{Succ}(i) \end{array}\right\}. \tag{11}$$

Relations (7) and (9) show that, if some constraints are relaxed ((4) or (0)), the RRPSP verifies dynamic programming relations. These are necessary for computing earliest and latest starting times using usual forward and backward recursion algorithms (Elmaghraby 1977).

3.2. *Analysis of inconsistency*

3.2.1. *Detection of inconsistency and its causes*

In a *solution* to a constraint satisfaction problem $\text{P} = (V, D, C)$, all variables in V have assigned values in D such that all constraints C are satisfied. If such an assignment is impossible the problem is inconsistent (Verfaillie and Lobjois 1999). The previous section showed that the earliest schedule can be computed if constraint (4) is relaxed. Since no task can begin before its earliest starting date, the RRPSP is inconsistent if and only if this earliest schedule does not fulfil due date constraints (4).

$$\text{RRPSP is inconsistent} \Leftrightarrow \exists i \in [0,I], \quad \underline{t_i} > J_i - d_i. \tag{12}$$

Reciprocally, the latest schedule can be computed relaxing constraint (0). Since there is no schedule that can start after the latest one:

$$\text{RRPSP is inconsistent} \Leftrightarrow \overline{t_0} < 0. \tag{13}$$

By using test (12) or (13), the inconsistency can be detected in a polynomial computation.

For an inconsistent scheduling problem $P=(V, D, P, C)$, a *cause* of the inconsistency of P is a subproblem $P'=(V, D, P, C')$, with C' included in C, such as P' is inconsistent (Jussien 1997).

The problem inconsistency can be identified from the earliest starting dates and due date constraints (12). Earliest starting dates of tasks are equal to the earliest length of their critical path. Consequently, any path from node (0) to a node (i) whose earliest length exceeds the due date on node (i) is an inconsistent sub-RRPSP and therefore a cause of the inconsistency. We call such a path an inconsistent path.

3.2.2. *Identification of the inconsistent paths*

In a forward recursion algorithm (Elmaghraby 1977), although all paths are implicitly explored, only one (the longest) is explicitly stored and taken into account to obtain the value for the earliest starting date of a task.

To make the best use of the algorithm, in each node the starting dates obtained from the other paths can be memorized in addition to the value obtained from the longest path. These dates will be organized into a list associated with each task. This list contains all the earliest starting dates obtained for all the paths leading to a particular task.

As each of these values can be associated with its corresponding predecessor, it is possible to generate all of the paths.

Consider a task T_j. $\overline{t_j}$ denotes its latest starting date. It results from the backward recursion process and is the latest admissible starting time for task T_j that will not violate a due date on a successor of task T_j. Consequently, only starting dates t_j where $t_j > \overline{t_j}$ cause an inconsistency and therefore must be stored. The algorithm in figure 3 computes all the inconsistent paths $P_\alpha(0, i)$ for all the nodes $i \in [0, I]$.

Let L_j be the list of paths arriving at node (j), sorted in a decreasing order of the arriving date. The function AddList(L_j, t', t, i) adds to L_j an earliest path $(i) \rightarrow (j)$ with the values $t_i = t$ and $t_j = t'$.

Figure 4 shows the result of this algorithm for an example.

Since the number of paths $P_\alpha(0, j)$ increases exponentially with the number I of tasks, the length of the list L_I also increases exponentially. Consequently, the algorithm may need exponential computation time and memory when many paths are inconsistent.

However, note that the number of earliest starting dates in a list can be limited since they are ranked in decreasing order. Limiting the length of the lists L_i to a given value L will guarantee that the algorithm finishes in $O(I.I.L)$. However, only the longest paths and hence the most critical causes, will be obtained.

3.3. *Margins and schedule analysis*

3.1.1. *Global slack times*

When the earliest and latest schedules are computed, the starting date t_i of a task T_i belongs to $[\underline{t_i}, \overline{t_i}] \cup D_i$.

0: initialisation. $L_0 = \{0\}$ *and* $L_i = \{\}$ *for* $i > 0$.
1: compute all the latest starting times $\overline{t_j}$
considering that constraint (0) is relaxed.
For all (i) in reverse topological order,

$$\overline{t_i} = MIN\left((J_i - d_i),\ \underset{j \in Succ(i)}{Min}\left[\left(\overline{t_j} - a_{i,j}\right),\ \underset{x \in [1,n_i]}{Max}\left(ED_x^i / SD_x^i \le \overline{t_j} - a_{i,j}\right)\right]\right)$$

2: if $\overline{t_0} > 0$, *STOP, there is no inconsistency*
3: For all (i) in topological order
For all t in L_i
For all (j) in Succ(i)
If $(t + EarliestDelay(i,j,t) > \overline{t_j})$
AddList(Lj, t + EarliestDelay(i,j,t), t,i)
EndFor
EndFor
EndFor

Figure 3. Algorithm for computing inconsistent paths.

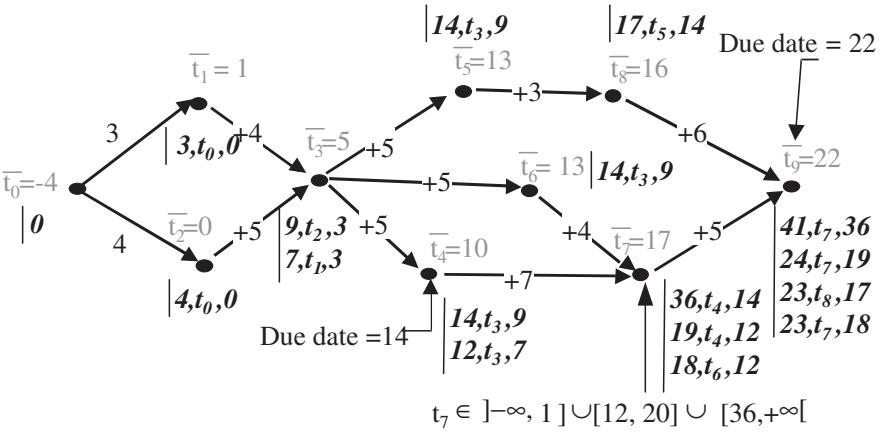


Figure 4. Result of the algorithm.

We propose extending margin indicators to help the decision-maker analyse these schedules.

Total slack time is the difference between the latest starting date and the earliest starting date:

$$M_i = \overline{t_i} - \underline{t_i}. \tag{14}$$

'Dotted' slack time is the slack time taking resource constraints into consideration:

$$m_i = \left(\overline{t_i} - \underline{t_i}\right) - \sum_{x/ED_x^i > \underline{t_i} \text{ and } SD_x^i < \overline{t_i}} \left(\min(ED_x^i, \underline{t_i}) - \max(SD_x^i, \overline{t_i})\right), \tag{15}$$

where $\sum_{x/ED_x^i > t_i \text{ and } SD_x^i < \overline{t_i}} (\min(ED_x^i, \underline{t_i}) - \max(SD_x^i, \overline{t_i}))$ is the loss of time due to unavailability of resources between $\underline{t_i}$ and $\overline{t_i}$. m_i is the real degree of freedom for

managing a task T_i, while M_i is the degree of freedom that exists when other tasks are also taken into consideration. If the ratio m_i/M_i is close to 0, prior tasks being slightly overdue or subsequent ones being early can cause an important reduction in the slack time of task T_i. This ratio helps to identify tasks sensitive to 'environmental' changes.

Pre-task shift (14) expresses the delay on $\underline{t_i}$, which is not due to the predecessors of task T_i but to the resource constraint.

$$Pa_i = \underline{t_i} - \max_{j \in \mathrm{Pred}(i)} \left(\underline{t_j} + d_j \right). \tag{16}$$

Reciprocally, post-task shift (15) expresses the advancement of $\overline{t_i}$ not due to the successors of task T_i but to the resource constraint on task T_i:

$$Pa_i = \operatorname*{Min}_{k \in \mathrm{Succ}(i)} \overline{t_k} - (\overline{t_i} + d_i). \tag{17}$$

Therefore, pre- and post-task shifts express loss of time, with no added value for the project. They are due solely to waiting for resources.

3.3.2. *Slack times on a path*

As shown, the causes of inconsistency are paths that lead from a ready date to a due date. It is therefore necessary to analyse slack time on a path.

Consider a path c that leads from a ready date α_c to a due date β_c. We define $\underline{t_{i,c}}$(respectively $\overline{t_{i,c}}$) as the earliest (respectively latest) starting time of a task T_i computed on the sub-RRPSP, limited to the path c. Such starting times are computed through forward and backward recursion. Consequently, $\underline{t_{i,c}} - \alpha_c$ (respectively $\overline{t_{i,c}} - \beta_c$) is the earliest (respectively latest) length of the portion of path c from α_c (respectively T_i) up to T_i (respectively β_c). Therefore, analysing $\underline{t_{i,c}}$ or $\overline{t_{i,c}}$ results in analysing part of a path.

Let us define the slack indicators on a task T_i that belongs to the path c as:

$$\underline{s_{i,c}} = \underline{t_i} - \underline{t_{i,c}}, \qquad S_{i,c} = \overline{t_i} - \underline{t_{i,c}}, \qquad \overline{S_{i,c}} = \underline{t_i} - \overline{t_{i,c}}, \qquad \overline{s_{i,c}} = \overline{t_i} - \overline{t_{i,c}}.$$

If a consistent RRPSP is considered, $\underline{s_{i,c}}$ (respectively $\overline{s_{i,c}}$) expresses slack time that can be consumed by the tasks of path c that precede (respectively succeed) T_i without consuming the total slack time of task T_i. Moreover, $S_{i,c}$ (respectively $\overline{S_{i,c}}$) expresses the admissible increase of the part of path c before (respectively after) task T_i that can be used without making the RRPSP inconsistent.

Now, if an inconsistent RRPSP is considered, $S_{i,c}$ and $\overline{S_{i,c}}$ express the slack time that must be gained on part of a path in order to make it consistent. Choosing task T_i on which these slack times are the smallest, means choosing the task where the smallest change can make the path consistent. Such information becomes important when repairing an inconsistent RRPSP.

3.3.3. *Distance between two planning procedures*

During the life of a project, perturbing events occur. Some will cause inconsistency in the project schedule. The decision-maker may not be able to identify the disturbing events (These would be too numerous to count between two planning procedures.) Most importantly, he/she needs support in analysing the consequences they may have.

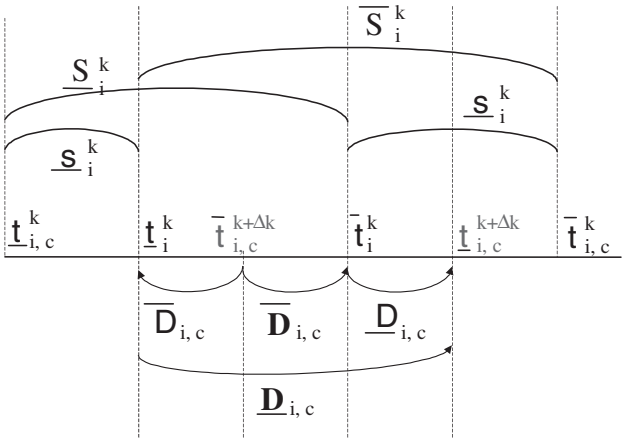


Figure 5. Representation of slacks and distances.

We consider that a consistent project schedule was computed at time k. Now, a new planning procedure is introduced at $k+\Delta k$, which leads to an inconsistency. Only those tasks that are part of the problem at time $k+\Delta k$ are to be considered. The others have begun and have fixed starting times. We also presume that the project manager will keep referring to the initial schedule. Therefore, by analysing the evolution of the incoherent paths between the two successive planning procedures, the portions of paths that increased between k and $k+\Delta k$ can be identified: they explain the emergence of the inconsistency. Two approaches are available to the decision-maker for restoring consistency:

- Controlling, supervising and reducing the length of the portions of paths identified.
- Acting on the tasks up- or downstream from these portions of paths, so that the project schedule withstands the perturbing events.

Consider an inconsistent path c at time $k+\Delta k$ that leads from a ready date α_c to a due date β_c.

Consider that T_i belongs to c. Using exponent indices that express the time at which a starting date is taken into account (either k, or $k+\Delta k$), it can be stated that:

$$\underline{t^k_{i,c}} < \underline{t^k_i} < \overline{t^k_i} < \overline{t^k_{i,c}} \quad (c \text{ is consistent at } k)$$

$$\overline{t^{k+\Delta k}_i} < \overline{t^{k+\Delta k}_{i,c}} < \underline{t^{k+\Delta k}_{i,c}} < \underline{t^{k+\Delta k}_i} \quad (c \text{ is inconsistent at } k+\Delta k).$$

Using such notation, distance indicators (figure 9) between two planning procedures can be expressed. They measure the slack consumption between k and $k+\Delta k$ along a path, and consequently the changes in the length of the incoherent path c:

- $\underline{D_{i,c}} = \underline{t^{k+\Delta k}_{i,c}} - \underline{t^k_i}$. $\underline{D_{i,c}} > 0$ if slack $\underline{s^k_{i,c}}$ available at time k has been consumed at time $k+\Delta k$. This means that the 'earliest' length of path $\alpha_c \rightarrow T_i$ increased between k and $k+\Delta k$ and consumed part of the total slack time.
- $\underline{\mathcal{D}_{i,c}} = \underline{t^{k=\Delta k}_{i,c}} - \underline{t_j{}^k}$. $\mathcal{D}_{i,c} > 0$ if slack $\underline{\mathcal{S}^k_{i,c}}$ has been consumed between time k and $k+\Delta k$. This means that the earliest length of path $\alpha_c \rightarrow T_i$ increased so much between k and $k+\Delta k$ causing an inconsistent path at time k.
- $\overline{D_{i,c}} = \overline{t^{k+\Delta k}_{i,c}} - \overline{t^k_i}$. > 0, if slack $\overline{s^k_{i,c}}$ has been consumed, meaning that the latest length of path $T_i \rightarrow \beta_c$ increased between k and $k+\Delta k$.

- $\overline{\mathcal{D}_{i,c}} = \underline{t_i^k} - \overline{t_{i,c}^{k+\Delta k}} > 0$ if slack $\underline{\mathcal{S}_{i,c}^k}$ has been consumed. Then, the latest length of the path $T_i \rightarrow \beta_c$ increased so much between k and $k + \Delta k$ causing an inconsistent path at time k.

The task's position on the path needs to be considered: as far as production planning is concerned, the earlier the task occurs on the path the more efficient the action will be.

Now, depending on the relative values of $\underline{t_{i,c}^{k+\Delta k}}$, $\underline{t_i^k}$, $\overline{t_{i,c}^{k+\Delta k}}$ and $\overline{t_i^k}$, six configurations can be identified if a path c is inconsistent at time $k + \Delta k$. Figure 6 expresses the positive or negative nature of the distance indicators in these situations.

In configuration 1, the inconsistency is due to a significant increase of the path $\alpha_c \rightarrow T_i$. In fact, the increase of $T_i \rightarrow \beta_c$ does not lead to an over-consumption of slacks $\overline{s_{i,c}^k}$ or $\overline{\mathcal{S}_{i,c}^k}$. The restoration process must lead to $\underline{\mathcal{D}_{i,c}} < 0$. The decision-maker must intervene with restorative action on $\alpha_c \rightarrow T_i$. Reciprocally, in configuration 6, one can conclude that the inconsistency is due to a significant increase in the path between $T_i \rightarrow \beta_c$.

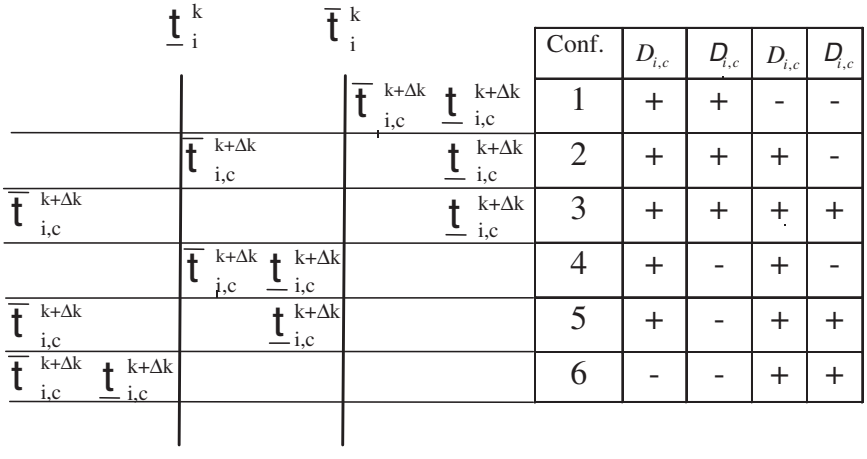
In configuration 2 (respectively 5), the inconsistency is due to an increase in path $\alpha_c \rightarrow T_i$ (respectively $T_i \rightarrow \beta_c$). Nevertheless, this inconsistency will persist even if restoration leads to $\underline{\mathcal{D}_{i,c}} < 0$ (respectively $\overline{\mathcal{D}_{i,c}} < 0$). In configuration 2, for example, $\overline{t_{i,c}^{k+\Delta k}} < \underline{t_{i,c}^{k+\Delta k}} < \overline{t_i^k}$ can be obtained after restoration. Then, $\underline{\mathcal{D}_{i,c}} < 0$. Nevertheless, $\overline{t_{i,c}^{k+\Delta k}} < \underline{t_{i,c}^{k+\Delta k}}$: path c is still inconsistent.

In configuration 3, all the distances are positive. Any part of the path ($\alpha_c \rightarrow T_i$ or $T_i \rightarrow \beta_c$) explains the inconsistency. Consequently, the decision-maker must intervene with restorative action on $\alpha_c \rightarrow T_i$ and on $T_i \rightarrow \beta_c$.

In configuration 4, $\underline{\mathcal{D}_{i,c}} < 0$ and $\overline{\mathcal{D}_{i,c}} < 0$. The growth of one path $\alpha_c \rightarrow T_i$ or $T_i \rightarrow \beta_c$ does not in itself explain the inconsistency. Rather, it results from the simultaneous increase of both paths $\alpha_c \rightarrow T_i$ and $T_i \rightarrow \beta_c$. The restorative action should be applied to anyone of these paths.

3.4. *Decision-maker support for restoring consistency*

Restoring problem consistency means modifying the actual problem to make it consistent, once the incoherent paths have been studied. This will require making changes (reduction of task duration, modification of the sequence) that lead to a



Conf.	$D_{i,c}$	$D_{i,c}$	$D_{i,c}$	$D_{i,c}$
1	+	+	-	-
2	+	+	+	-
3	+	+	+	+
4	+	-	+	-
5	+	-	+	+
6	-	-	+	+

Figure 6. Relationship between the six configurations and the distance indicators.

solution. If the problem itself is poorly defined (impossibility of listing all alternatives for the production process, reallocation of resources, re-reservation of resources, ready dates negotiation with the suppliers), the number of possible restorations is very high. Moreover, we consider that the decision-maker's choices cannot be reduced to an automatic restoration procedure.

The decision-maker will act according to information provided by the system. One key piece of information is related to the notion of a group of paths. Two paths $P_\alpha(i,j)$ and $P_\beta(i,j)$ belong to the same group if a ready date constraint deals with the task i and a due date constraint deals with the task j. The information provided to the user is the intersection between the incoherent paths of a group, or between all paths, the slack time on a path, the sets of tasks belonging to several incoherent paths in the whole graph or in a group of paths.

When the problem is inconsistent, the decision-maker is given the graphs shown in figure 7. The first screen enables the decision-maker to view all the inconsistent paths, including the tasks that require resource reservation, and the incoherent sets of paths. The second screen details a given incoherent set of paths and focuses on the intersection between inconsistent paths and tasks associated with important disturbances: slack time indicators on a path (s and S types) enable selection of these tasks on the path. Analysis of the distance indicators enables identification of that part of incoherent paths whose length has increased and led to the inconsistency.

The aim is to focus on a set of given tasks, where action will readily lead to restoration. For instance, the duration of a task that belongs to all the incoherent paths could be reduced to act on all the paths. A procedure for restoring consistency is proposed for the industrial application under study (see Section 4.2).

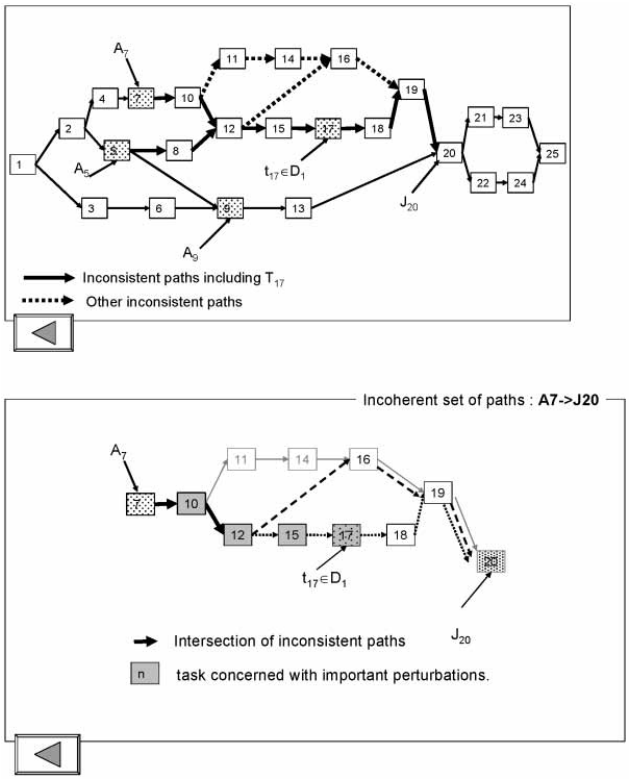


Figure 7. Inconsistent paths and the set of inconsistent paths, the number of inconsistent paths, the number of groups of inconsistent paths and the size of the intersection between all the paths.

4. Industrial application

4.1. *Industrial problem*

The present study led to an initial prototype that was developed and tested on industrial data in collaboration with a company that devises and assembles complex make-to-order products.

The company under study designs and assembles satellites. The chosen site was the assembly division (production), which faces problems specific to the sharing of common resources. These are 'heavy' physical external resources (capacity = 1) such as material-handling facilities or test-beds. Using a predetermined sequence, the project manager schedules tasks according to the availability of external resources. Finding solutions to resource sharing conflicts forces him/her to update plans or schedules frequently, and requires negotiation between decision-makers. To date, communication between projects is mainly on an informal basis. Nevertheless, the need for a re-planning support system for each project is recognized.

This decision tool was tested using several real cases of a problem that occurred during the assembling of a satellite. Integration planning is composed of about 60 tasks plus 100 due date and supply constraints. Two main alternative integration sequences were considered, along with different due date and resource constraints (variation in the number of resources) resulting in 43 RRPSP problems. Computation of earliest starting dates for the 43 problems considered lasted between 1 min 50 s and 3 min 30 s. Where the detection of consistency or of inconsistency was concerned, processing time was less than 1 s. Hence, the belief that the RRPSP problem can be solved through interaction with a human decision-maker.

To evaluate the impact of classical project parameters and of resources (figure 8), two main indicators are used:

- Slack time of the project if the resources are not taken into account.
- 'Tightness of the constraints', which is calculated by totalling the tightness of each resource. A given resource tightness is the ratio between time consumed and the time this resource is available during the project.

One point in figure 8 refers to a RRPSP problem. All the problems have a positive total slack time without resources and therefore are consistent if resource constraints are relaxed. Note that a very low level of constraint tightness (from 0.05 to 0.2) causes the problem inconsistency even if the total slack time without resources

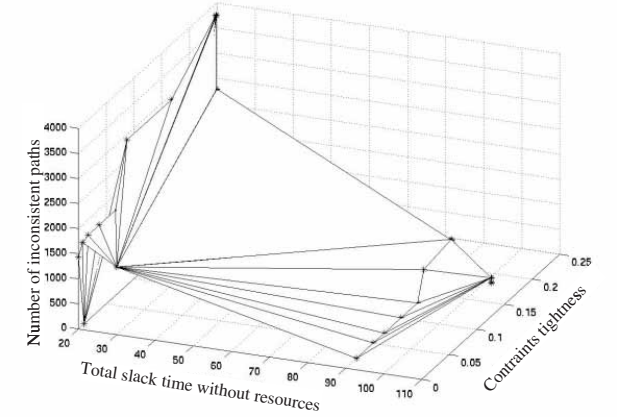


Figure 8. Effect of project parameters and constraint tightness on the number of inconsistent paths.

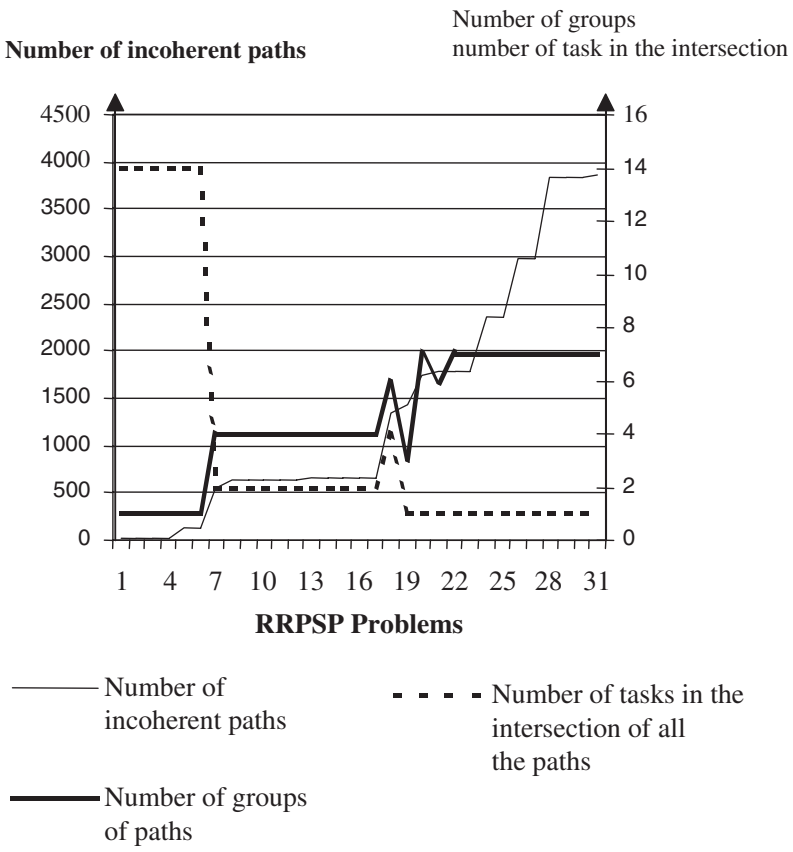


Figure 9. Comparison between the number of inconsistent paths, the number of groups of inconsistent paths and the size of the intersection between all the paths.

is high. This emphasizes the worth of considering availability of the critical resource during medium-term planning.

Because of the high number of inconsistent paths, the causes of the problem inconsistency were also examined and the following parameters were studied: the number of groups of incoherent paths and the size of the intersections of all the paths (figure 9).

One natural decision is to act simultaneously on many incoherent paths by acting on the tasks that belong to the intersection of many paths. Figure 9 shows clearly that the intersection of all the paths is quickly null or very limited (one task). Another possibility is to separate the problem and work on the groups of incoherent paths. Here, figure 9 demonstrates that the number of groups is tractable in the present case study: between four and seven groups. Consequently, a manager can lead an analysis per group of incoherent paths.

4.2. *Procedure for restoring consistency*

As a result of previous findings, the decision-maker adopted a procedure to restore the consistency:

1. When a set of incoherent paths including a task T_i associated with a resource reservation exists:
   - Identify the intersection between the inconsistent paths that led to the inconsistency.
   - Identify the intersection between the portions of inconsistent paths that led to the inconsistency, and the other inconsistent paths.

- Using the slacks, identify by how much the length of the portions of path must be reduced to restore consistency.
- Decide which action must be taken: reduce task duration, bring forward a particular supply, extend a due date or modify the window reservation for the resource.
- Negotiate with the different participants in the project (production, suppliers, customers and the supplier in charge of resource reservation).
- Modify data concerning the problem.

2. For all the other inconsistent paths:
   - Identify the intersection between these inconsistent paths.
   - Using the slacks, identify by how much the length of the portions of path must be reduced to restore consistency.
   - Decide which action must be taken: reduce task duration, bring forward a particular supply, extend a due date or modify the window reservation for the resource.
   - Negotiate with the different participants in the project (production, suppliers, customers and the supplier in charge of resource reservation).
   - Modify data concerning the problem.

5. Conclusion and perspectives

A decision support tool for obtaining earliest and latest schedules for the reserved resources project-scheduling problem (RRPSP) is developed according to the industrial requirement analysis. The earliest and latest schedules of the RRPSP are then used to determine the consistency or inconsistency of the problem. In the first case, a group of possible schedules is defined. The user will then benefit from a graph analysis based on these two schedules in terms of slacks, critical tasks and critical path. When no consistent schedule can be found, the set of inconsistent paths is established. Hence, the project manager can be informed about the inconsistency of the problem using several types of slack indicators applied to the inconsistent paths. This study enables one to emphasize the following:

- Worth of developing the graph-analysis aspect to provide project managers with adequate interface.
- Worth of providing the user with more precise indications on the causes of the inconsistency.

The priority was to help the project manager use this information to restore problem consistency. From this point of view, the worth of such a tool in a distributed management framework was demonstrated in an industrial context. Indeed, a project manager can run a project autonomously or by negotiating with the other participants in the project environment. The tool remains a prototype. Significant interaction between the users and the decision-maker is still necessary to enhance the use of the tool. A better understanding of a decision-maker's actions will enhance the restoration process. Moreover, the RRPSP model has being extended to include uncertain task duration as well as uncertain resources availability using a possibilistic approach (Fargier and Thierry 2002). Finally, the negotiation process involving the external, critical resources subcontractors is also under study. In this context, both customer and subcontractor points of view are analysed from a supply chain perspective.

References

Boctor, F. F., 1996, A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes. *Eur. J. Oper. Res.*, **90**(2), 349–361.

Brucker, P., Drexl, A., Möhring, R., Neumann, K. and Pesch E., 1999, Resource-constrained project scheduling: notation, classification, models, and methods. *Eur. J. Oper. Res.*, **112**(1), 3–41.

Chen, Y.-L, Rinks, D. and Tang, K., 1997, Critical path in an activity network with time constraints. *Eur. J. Oper. Res.*, **100**, 122–133.

Elmaghraby, S. E., 1997, *Activity Network*: *Project Planning and Control by Network Models*, (New York: Wiley).

Fargier, H. and Thierry, C., 2002, Evaluation du risque dans la gestion d'un projet sous contraintes de ressources externes: l'approche probabiliste, in LFA, Rencontres francophones sur la logique floue et ses applications, Montpellier, France.

Golenko-Ginsburg, D. and Blokh, D., 1997, A generalized activity network. *J. Oper. Res. Soc.*, **48**, 399–400.

Jussien, N., 1997, Relaxation de Contraintes pour les problèmes dynamiques. PhD thesis, Université de Rennes I.

Kelley, J. E., 1961, The critical path planning and scheduling: mathematical basis. *Oper. Res.*, **9**, 296–320.

Kolisch, R., 2001, *Make-to-Order Assembly Management* (Berlin: Springer).

Kolisch, R. and Padman, R., 2001, An integrated survey of deterministic project scheduling. *OMEGA*, *Int. J. Manag. Sci.*, **29**(3), 249–272.

Kurtulus, I. and Narula, S., 1985, Multi-project scheduling. Analysis of project performance. *IIE Trans.*, **17**(1), 58–66.

Leus, R., 2003, The generation of stable project plan: complexity and exact algorithms. PhD thesis, Katholieke Universiteit Leuven.

Neumann, K., Schwindt, C. and Zimmermann, J., 2003, *Project Scheduling with Time Windows and Scarce Resources* (Berlin: Springer).

Salewski, F., Schirmer, A. and Drexl, A., 1997, Project scheduling under resource and mode identity constraints: model, complexity, methods, and application. *Eur. J. Oper. Res.*, **102**(1), 88–110.

Speranza, M. G. and Vercellis, C., 1993, Hierarchical models for multi-project planning and scheduling. *Eur. J. Oper. Res.*, **64**, 312–325.

Verfaillie, G. and Lobjois, L., 1999, Problèmes incohérents: expliquer l'incohérence, restaurer la cohérence, in Proceedings of the 5th Journées Nationales sur la Résolution Pratique de Problèmes NP-Complets (JNPC-99) Conference, Lyon, France.

Weglarz, J. (ed), 1999, *Project Scheduling – Recent Models, Algorithms and Applications* (Boston: Kluwer).

Yang, K. and Sum, C., 1997, An evaluation of due date, resource allocation, project release, and activity scheduling rules in a multiproject environment. *Eur. J. Oper. Res.*, **103**, 139–154.