Static et dynamic scheduling of maintenance activities under the constraints of skills.

François Marmier, Christophe Varnier, Noureddine Zerhouni

Static and Dynamic Scheduling of Maintenance Activities Under the Constraints of Skills

François Marmier, Christophe Varnier, Noureddine Zerhouni

FEMTO-ST Institut AS2M Dept. – CNRS/ENSMM/UFC/UTBM
24, rue Alain Savary - 25000 Besançon - FRANCE
email: cvarnier@ens2m.fr

2009

Abstract

Skill management in industry is one of the most important factors required in order to obtain optimal performance of the production system. This is of particular importance in the field of maintenance where the different practical knowledge or skills are the working tools used. We address, in this paper, both the assignment and scheduling problems that may be found in a maintenance service. Each task that has to be performed is characterized by the level of skill required. The problem lies with making the decision of which time is the right time for the assignment and scheduling of the correct resource to do the task.

We introduce both static and dynamic scheduling, applied to the maintenance task assignment. To confer a maximum robustness to the obtained schedule, the approach proposed in this paper is completed by a proactive methodology which takes into account possible variations.

1 Introduction

To remain competitive, companies must decrease their costs as much as possible and optimize their production means operations. In order to confer a better availability of equipment, and through it a better availability of the company, the maintenance service intervenes. It deals with problems before and after breakdowns. This improvement mainly requires better management of the workforce and its skills. The reactivity and the organization of the maintenance service will depend on the importance of the treatment required.

Most information systems are built following the Herbert A. Simon [12] pyramid concept. The latter uses the classical diagram of organization: strategic / tactical / operational. Crespo-Marquez *et al.* proposed a decomposition to align maintenance management in this frame [1].

The highest level concerns non-schedulable tasks. This level concerns the maintenance service policy and maintenance strategies definition. At the tactical level it is possible to schedule decisions. It concerns the implementation of maintenance policies for the assignment and scheduling of tasks and resources. This information is given to the operational level where the corresponding action is carried out. Data concerning the work is recorded in the information system and available for decision-making. The figure, in figure 1, shows how maintenance services work.

If we focus on the tactical level of the maintenance service, we can observe that skills are important to determine the role of the personnel and to take the appropriate decision. Grabot *et al.* carried out a study on nineteen companies to obtain their opinions on the operators' assignment problem [4]. It showed that the management of operators, according to their skills, is important for industry leaders and that there is still no software available which takes this into account. 79% of the companies think that the management of operators is useful or essential in scheduling. While in current software the

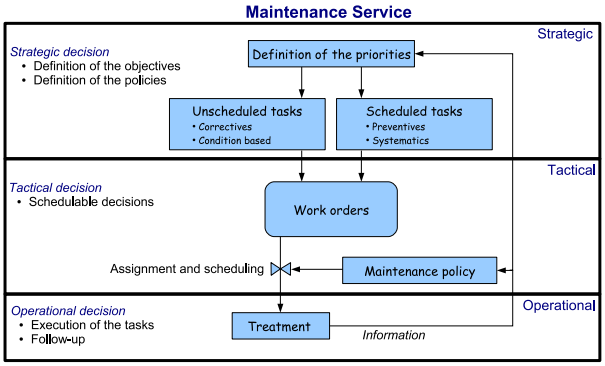

Figure 1: Decomposition of the maintenance services operation.

operational duration is fixed, for the leaders in industry, the consideration of the operators' qualification is a major element to be taken into account when setting up their assignments. For 47% of the companies studied, the qualification level sometimes has an influence on the length of time of the realization of the task while for 27% it always bears an influence. The need for further development appears to link the abilities of human resources and the duration of the operations as in the determination of the potential of the company. However, if the competency levels of each person are known, another problem has to be solved : balancing the workload of resources and trying to reach a compromise between the reactivity and the perturbation due to the modification of the employees planning. In a maintenance service context, Le Quéré *et al.* [**?**] show the difficulty of scheduling due to the different competencies. The different competency types which can be generic and used in various professional situations, or specific to the activity such as an habilitation or a special technique (due to the dominance of mechanics, electricity,...).

There are mainly two types of maintenance activities: preventive maintenance, whose activities are well known and can be planned long term, and corrective maintenance which is related to non-foreseeable breakdowns. Moreover corrective maintenance data used to schedule is estimated and then imprecise. Within the maintenance service, employees have different skills and various levels of qualification. Treatment speed and thus service reactivity will depend on the choice of the employees assigned to the task. The goal of the maintenance management is to assign tasks to the best-known resource. In this article, we detail a method to assign new tasks to resources by distributing the load between them. This approach is composed of a static part to schedule known and foreseeable tasks and a dynamic and proactive part to schedule corrective tasks.

In the next part we precise the context of the problem and the role of the maintenance manager. We then present the model followed by the solutions that we propose. Finally, we implement these solutions and present our results and conclusions on this work.

2 An assignment and scheduling problem

To precise the context of the problem, the three levels (strategic / tactical / operational) are detailed in this parts.

2.1 The context of industrial maintenance

The strategic level is the one of maintenance strategies definition [1]. Equipment which has to be maintained by the maintenance service have availability objectives. The priority of each intervention will depend on the importance of the maintained equipment. Then maintenance policies are defined and applied to determine the proportion of preventive tasks or schedulable intervention and corrective

tasks which are non-schedulable. For this reason, failings in critical equipment are anticipated by a preventive policy.

At the tactical level the maintenance manager receives work requests. These requests come from the information system if tasks were planed like the preventive maintenance ones. They would also come from the production service if tasks were not scheduled such as the corrective maintenance ones. The maintenance policies are carried out by assigning and scheduling tasks to the different resources. After the assignment, work requests are re-injected as work orders in the information system.

Work orders are consulted by human resources at the operational level. Workers have to carry out tasks and to record indicators in the information system. These data permits us to know if the policies and the equipment priorities have to be modified.

2.2 Problems for the decision maker

Human resources are able to deal with the majority of maintenance tasks. Due to this versatility they have differences of efficiency following the type of tasks to achieve [4]. In this work, we suppose that spare parts and tools are available for the intervention. The maintenance manager has an assignment and scheduling problem of maintenance tasks [**?**]. He has to take into consideration the particularity of the maintenance context such as the differences between resources, the differences between tasks (some are preventive others are corrective) and the fact that some tasks are already assigned and scheduled and others are new ones. The manager has to schedule tasks to achieve some objectives like the availability for equipment or the balancing of the load between the various resources.

The manager also has to take care of the different uncertainties of the context. Among uncertainties there is the fact that some tasks are randomly generated by events like breakdowns but also the fact that most data used to make the schedule provide from estimations. Estimations provided from diagnosis. These ones, can be obtained with tools such as Case base Reasoning [**?**]. Therefore the different level of competence of all operators are uncertain.

The issue is then an assignment and scheduling problem of maintenance tasks which is multi-objective and in an uncertain context.

2.3 A parallel machine problem

A maintenance service is an environment composed of m operators working in parallel. All of them are capable of performing each task, but not with the same efficiency. Moreover, the resource which is the most effective for one task, may not necessarily be effective for all tasks. Since the main resource are operators we are faced with a parallel machine problem, but with unrelated machines which is noted R or $R_m|\,\beta|\,\gamma$, where β represents the processing characteristics and constraints and the γ field contains the objective to be minimized [11].

Pfund and *al.* [10] presented a state of the art, unrelated parallel-machine. One part is more precisely devoted to the problem: $R_m \| \, C_{\max}$ in the non-premptive task cases. Among all unrelated parallel-machine scheduling problems, the ones which aim at minimizing the makespan, are the most studied. Some authors have developed approximate methods, which can be executed quickly but have no guarantee of reaching the optimum level. Ibarra and *al.* presented a methodology always used as a basis of comparison for current research in this field [6]. Their heuristic is based on a list algorithm and can lead to the worst case. Many others base their methodologies on a two-phase approach with an assignment problem solved by linear programming followed by a heuristic for the tasks which have not been assigned during the first part. It is in the particular case of Hariri and Potts [5] or Lenstra and *al.* [7] who used amongst other things, the heuristic of Ibarra and *al.* in the second phase. Other authors have developed exact methods which make the obtaining of the optimal solution possible. Mokotoff and Chrétienne [9] presented results obtained using an exact cutting plane algorithm and compared it with the exact algorithms of Van de Velde [2] and Martello [8]. However the minimization

of the $C_{\max}$ does not take into account priorities between tasks. In this way, problems which aim to minimize the weighted tardiness ($R_m \| w_j T_j$) are rarely found in literature.

In this paper we propose to decompose this problem in two sub-problems. The first one is a static problem, mainly to assign and to schedule preventive maintenance tasks. These tasks are considered as being well known. The second sub-problem deals with dynamic arrival of new tasks (mainly corrective maintenance tasks). As most of the data is estimated, we consider in this sub-problem that the information about the tasks present some uncertainties. Task characteristics are modelled in the next part.

3 Model

Our problem is composed of two sub-problems. The first one consists in assigning and scheduling a set of m resources to a set of n known tasks. The second one consists in inserting a new task in a current schedule composed of m resources already assigned to n scheduled tasks. The m resources are supposed to be available during the whole scheduling horizon.

3.1 Precise data

Given the fact that this problem is made with different precise and imprecise data, we model in this first part the precise data.

3.1.1 Tasks

Task characterics are modelled as follows : for each task j,

- p_j : standard duration of task j (this duration is subject to variations depending on the resource assignment),
- r_j : release date of task j,
- d_j : due date of task j (this value is estimated in function of the current availability of the equipment concerned),
- w_j : priority of the task due to the penalties which could be claimed if the treatment of task j is not performed on time.

The maintenance service is composed of m human resources ($i = 1...m$), characterized by a competence profile. Relative speeds do not depend only on the tasks. Each resource has a corresponding qualification level for each task. Operators will perform them more or less rapidly. The duration of the job j, by the human resource i is denoted by p_{ij}. With:

$$p_{ij} = f(p_j, Comp_{i,Cr_j}), \forall i \in \{1, ..., m\} \tag{1}$$

Where $Comp_{i,Cr_j}$ is the skill rate of resource i in the skill which is required to achieve the task j.

3.1.2 Human resources

The skill rates set can be represented with a matrix in which, for each different kind of job, the rate corresponding to the required skill can be found.

$$\begin{array}{c} \\ op_1 \\ \vdots \\ op_m \end{array} \begin{array}{c} \begin{array}{ccc} Comp_1 & \cdots & Comp_{Cr_n} \end{array} \\ \begin{bmatrix} Comp_{1,1} & \cdots & Comp_{1,Cr_n} \\ \vdots & \ddots & \vdots \\ Comp_{m,1} & \cdots & Comp_{m,Cr_n} \end{bmatrix} \end{array}$$

Skill levels are independent from one operation to another. A second factor being that, an operator can be the most efficient operator for one kind of operation and the least efficient for another.

3.2 Imprecise data

Imprecisions mainly concern corrective tasks which are non-foreseeable events. The diagnosis of the event enable us to evaluate the duration of the task. The release date of the task depends mainly on the date of the breakdown but also on the availability or on the delivery date of spare parts. The due date of the task is determined regarding the objectives of the maintenance service.

For these reasons this different data is judged as being imprecise. Moreover, these tasks being uncommon, the resource skill levels can vary. That is why, for the insertion of a new and corrective task, task data is imprecise.

Concerning task duration, we assume that a part of the total treatment time is sensitive to variations. The duration is then composed of an incompressible part and a variable part. Contrary to Esswein *et al.* who used a probability law in order to determine the presence of variations, we think that a schedule is made up of task duration previsions [3]. The totality of the tasks is subject to variations. The p_{ij}, r_j or d_j real values are obtained by using a normal law on their variable part. Each time we generate disturbances, we do it a hundred times and conserve the average.

3.3 Variables

The variables of our problem are the following ones for each task j:

- $t_j \quad (j = 1...n)$: starting of task j,
- $x_{ij} \quad (j = 1...n \; and \; i = 1...m)$: 0-1 value representing the tasks' assignment. $x_{ij} = 1$ if task j is assigned to a resource i, else $x_{ij} = 0$,
- $T_j \quad (j = 1...n)$: lateness of task j,
- $mod_j \quad (j = 1...n)$: represents the number of modifications made to the employees timetable. mod_j is incremented each time the assignment j is modified,
- $PL_i \quad (i = 1...m)$: potential load of the human resource i. It corresponds to the sum of the tasks durations assigned to i.

With :

$$PL_i = \sum_{j=1}^{n} x_{ij} * p_{ij} \tag{2}$$

3.4 Constraints

Constraints considered in this problem are as follows:

Each task has to be assigned only once to only one resource:

$$\sum_{j=1}^{n} x_{ij} = 1, \forall i \in \{1, ..., m\} \tag{3}$$

Task j cannot be planned before the equipment i is available:

$$t_j \geqslant r_j, \forall j \in \{1, ..., n\} \tag{4}$$

Resources are disjunctive. That means that resource can only be used by a task at the same time. Any couple of tasks (j_1, j_2) using the same resource is associated to the pair of disjunction $(j_1 \prec j_2) \vee (j_2 \prec j_1)$ [**?**]. Tasks using the same resource are then totally sequenced and there is a disjunction between the two inequalities of potential :

$$(t_{j_1} - t_{j_2} \geq p_{j_2}) \vee (t_{j_2} - t_{j_1} \geq p_{j_1}) \tag{5}$$

then :

$$\forall t, \forall i, \sum_{j=1}^{n} a_{ij}(t) \leq 1 \tag{6}$$

3.5 Objective

Any equipment, which may be subject to maintenance activities, is framed with a contract. In those contracts, we find specifications regarding consequences of the availability losses for the maintenance service provider. These consequences are modelled with the weight w_j. The penalties depend on the equipment contract, and they may be of a different form from one piece of equipment to another.

Tasks which are finished late, decreasing the equipment availability ratio, have an impact which is proportional to their weight. It implies that we have to minimize the total weighted tardiness.

$$\min \sum_{j=1}^{n} w_j T_j, \tag{7}$$

With T_j the tardiness of the task j.

3.6 Optimization problem

Our problem can then be summarized as follow:

$\min \sum_{j=1}^{n} w_j T_j,$

So :

$$\sum_{j=1}^{n} x_{ij} = 1, \forall i \in \{1, ..., m\},$$
$$t_j \geqslant r_j, \forall j \in \{1, ..., n\},$$
$$\forall t, \forall i, \sum_{j=1}^{n} a_{ij}(t) \leq 1,$$

with $T_j = max(O, C_j - d_j)$ and $C_j = t_j + p_{ij}$,

with $p_{ij} = \sum_{j=1}^{n} x_{ij} * f(p_j, comp_{i,cr_j})$.

4 Resolution approaches

The maintenance manager has to assign and schedule a set of known tasks to a set of resources. These tasks are mainly preventive ones which have t be scheduled in an horizon of medium term. This can be done for each horizon period with a static scheduling approach. However regularly, in the short-term horizon, events (such as breakdowns) lead to the insertion of new tasks in the current schedule. Then, a new schedule must be generated with a dynamic scheduling approach. Both approaches are presented in this part.

4.1 A static scheduling approach

In this part we detail the heuristic which has been developed in order to solve the static problem. This will be applied to the preventive maintenance task assignment. The developed heuristic is a two-phase heuristic. The fact that tasks are assigned to human resources and the comparison of our problem with the problem of the parallel machines justifies that we balance and reduce the load. The first phase consists in minimizing the makespan in order to balance the load between resources. To minimize the makespan we introduce a lower bound. The second phase of the heuristic consists in taking into account due dates and release dates.

4.1.1 Lower Bound

We used the lower bound which is the simplest limit for a problem like $R_m \| C_{\max}$ and that we find in particular in the work of Ibarra and *al.* or of Mokotoff and Chrétienne. It consists of taking for each of the n tasks, the most powerful of the m resource and deducing the shortest duration p_{ij} for each task. Thus we obtain for the Lower Bound:

$$LB(C_{\max}) = \max \left\{ \left\lceil \frac{1}{m} \sum_{i=1}^{n} p_i^{\min} \right\rceil ; \max_{i \in \{1,\ldots,m\}} p_i^{\min} \right\} \tag{8}$$

with:

$$p_j^{\min} = \min p_{ij}, j \in \{1, ..., n\} \tag{9}$$

4.1.2 Assignment algorithm

This algorithm corresponds to the first heuristic phase and permits the makespan to be minimized. In a first stage, tasks are sorted according to the longest duration in a decreasing order and assigned, one by one, to the most efficient corresponding resource. If the assignment of a task to a resource implicates that the resource workload should be higher than the lower Bound, we try to assign the task to the next most efficient resource which would have a lower workload than LB. This resource must not be the worst one for that type of task. In the letter case we will search the resource, which would finish the task treatment first. If this resource is the worst one, we would use the exception algorithm. We propose two approaches in reply to these two problems.

- **The main algorithm :**

L = { tasks order by decreasing longest duration p_{ij} } ;
$\bar{\text{L}} = \emptyset$;
While (L $\neq \emptyset$) **Then**
  k $\leftarrow$ first task of L;
  i $\leftarrow$ fastest resource for process task k;
  If $\sum_{j \in \bar{L}} p_{ij} x_{ij} + p_{ik} \leqslant$ LB **Then**
    $x_{ik} \leftarrow 1$;
    $x_{ak} \leftarrow 0$, **for** $a = 1 \ldots n$ and $a \neq i$;
    $\bar{\text{L}} \leftarrow \bar{\text{L}} + k$;
    L $\leftarrow$ L $- k$;
  Else try to assign task k to the fastest resource l,
    With $l = 1 \ldots n$ and $l \neq$ worst case
    that respect $\sum_{j \in \bar{L}} p_{ij} x_{ij} + p_{ik} \leqslant$ LB;
    If l not found
      find resource l so that:

$\min_{l=1\ldots n} \sum_{j\in\bar{L}} p_{lj}x_{lj} + p_{lk}$
If l = worst case
Exception algorithm;
End If
End If
$x_{lk} \leftarrow 1$;
$x_{ak} \leftarrow 0$, for $a = 1 \ldots n$ and $a \neq l$;
$\bar{\mathrm{L}} \leftarrow \bar{\mathrm{L}} + k$;
$\mathrm{L} \leftarrow \mathrm{L} - k$;
End If
End While

The exception algorithm evoked in the main algorithm has for objective to reduce as much as possible the number of tasks assigned to the least efficient resource. The first task which would not be treated by the worst resource is moved to the top of the list. If it is impossible to find one task which would not be assigned to another one, it is assigned to the resource which would complete the treatment of the first.

• **The exception algorithm :**

Insert the first task on the list which would not be treated
by the worst resource at the top of the list
If all tasks $\in$ L check $p_j^{\max} = \max p_{ij}$ **Then**
Then assign the tasks without being worried by the fact
that it could be to the worst resource
End

4.1.3 Tardiness penalties

Tardiness is the respect of the tasks due-dates. There exist various methods for taking tardiness into account. In the previous algorithm we worked on minimizing the maximal completion time (C_{max}). We compared results already obtained, reorganized with an Earliest Due Date (EDD) post-treatment within each resource solution, with two other possibilities.

To consider tardiness, we replaced the pre-treatment with fixed tasks by decreasing longest duration, by another with sorted tasks by the due-date d_j in increasing order (EDD). In order to take into account the potential penalties of each late task, we also tried a Weighted Shortest Processing Time first (WSPT) pre-treatment which sort tasks by their decreasing w_i/p_i.

After the assignment phase, the sorting of the tasks can generate a high number of late tasks. To avoid this effect, we placed an EDD post-treatment sort for each resource assignment. Results of this adaptation are presented in table 2, where H-EDD is our heuristic followed by an EDD post-treatment, EDD-H-EDD means that the Longest Processing Time (LPT) pre-treatment has replaced an EDD and finally WSPT-H-EDD means that the pre-treatment is a WSPT one.

5 A dynamic scheduling approach

As new tasks are mainly maintenance corrective tasks, their characteristics are stochastic as long as a diagnosis has not been made. When a new task has to be inserted and when their is not any obvious solution, two ways are possible. The first way consists in generating a completely new static scheduling. This methodology does not take into account any potential disturbance for employees who

have a new planning. The second one consists in searching new scheduling with a few modifications to the current schedule. This kind of approach enables to disturbed as little as possible the existing planning and the employees' organization.

5.1 A partial re-scheduling methodology

The proposed method is inspired from a neighbourhood search. This method is principally based on local descent and on the kangaroo methodology in order to avoid local blocking.

The algorithm uses the following notation:

- S denotes the current solution,
- S' a neighbour solution of S,
- $Initialization(S)$: find the initial solution. This is found by trying all the various possibilities of insertion of the new task in the current scheduling. These solutions (or schedulings) are then compared and the best one is kept.
- $neighbour(S)$: find a neighbour of S by exchanging two tasks which have been chosen randomly,
- $jump(S)$: find a new solution after consecutively using three times the neighbourhood operator on different tasks which have been chosen randomly.

The parameters of the algorithm are :

- nb_search_max : Destined to limit the number of passes in the algorithm,
- $nb\_descent\_max$: Destined to limit number of local search
- nb_jump_max : Destined to limit the number of jump with to go out of a local optimum,.

We use the dominance relation between two solutions X_1 and X_2. This relation is noted by $X_1 \prec X_2$ where X_1 dominates X_2. The fact of proceeding stochastically to task exchanges rather than to a stochastic displacement permits to conserve a certain balancing of the load. The balancing of the load is usually made with the total duration of tasks assigned through the number of tasks.

- **The algorithm :**

Initialization (S) ; **While** $(nb\_search \leq nb\_search\_max)$ **Then**
  $nb\_search++$;
  $nb\_descent \leftarrow 0$;
  While $nb\_descent \leq nb\_descent\_max$ **Then**
    $nb\_descent++$;
    $S' \leftarrow neighbour(S)$;
    If $S \prec S'$
      $nb\_descent \leftarrow 0$;
      $S \leftarrow S'$;
    End If
  End While
  While $(nb\_jump \leq nb\_jump\_max)$ **Or** $(find == false)$ **Then**
    Nb_jump++ ;
    $S' \leftarrow jump(S)$;
    If $S' \prec S$ **Then**
    $nb\_jump \leftarrow 0$;

$S \leftarrow S'$;
$find \leftarrow true$;
End If
End While
End While

5.2 Evaluation of solutions : modelling with graphs

We consider a current schedule (already computerized) which integrates n tasks that have already been assigned to m human resources. The current schedule can be modelled as a graph (figure 2). The graph is decomposed in chains, each one representing a human resource schedule. They are composed of nodes which represent tasks and arcs which are the potential constraints between two tasks (precedence). The valuation of arcs is the duration of the original task. Tasks are placed between a fictive beginning task B and a fictive end task E. There is no link between branches, because resources work independently.

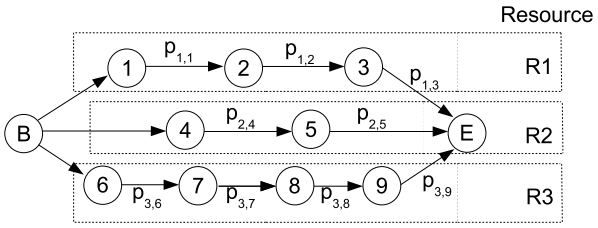

Figure 2: Current planning

To introduce the release dates constraints all nodes are linked to the B node and are valuated by r_i. Bellmann long path algorithm can be performed to find the earliest starting dates of each task (graph "Earliest Starting Date Graph" on figure 3), denoted ES_j for task j. ES_j value is either to the release date constraint or to the task sequence in the schedule.

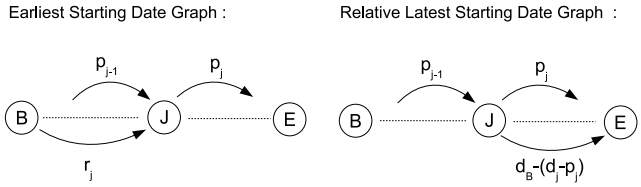

Figure 3: Graphs use to model the scheduling problem

To find the latest starting time LS_j, it is necessary to consider the due date of each task. We propose to construct a second graph, in which we keep the same node and the original potential constraints. However we add arcs between each task j and the fictive end of task E. These new links are valued, for each task j, by the difference between the branch last task due-date d_B and $d_j - p_{ij}$ (where i is the resource assigned to task j). Then, an adaptation of Bellmann algorithm allows the relative latest starting time LS_j^E to be found (in reverse order). We call LS_j^E, the relative latest starting time, since, for resource i, it represents the duration between the end of the last task assigned to i and starting time of task j.

Computation of ES_j and LS_j can lead us to observe a delay in the treatment of the task j.

The real latest starting date of a task j, is thus obtained as follows:

$$LS_j^E > ES_j \rightarrow LS_j = LS_j^E \tag{10}$$

$$ES_j > LS_j^E \rightarrow LS_j = ES_j \tag{11}$$

In order to study the different place within the schedule where a new task could be inserted, we have to check time windows between tasks.

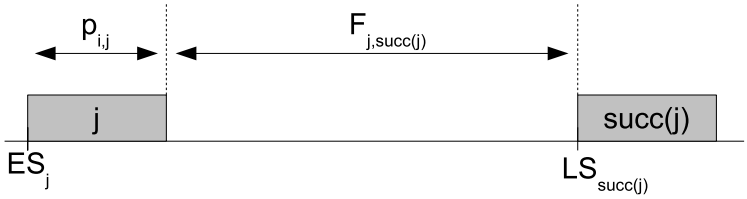

Figure 4: Task insertion window

The computation of the earliest and latest time for each task allows us to evaluate the available time windows within a current schedule. The case, presented in figure 4, describes a time window. We observe there the classical case, where new tasks will be inserted.

As explained earlier, this dynamic approach can mainly be used to integrate corrective maintenance tasks in a current schedule. Their data is principally estimated and the schedule approach has to anticipate the uncertainties.

5.3 Scheduling approach under uncertainties

A new task will have different characteristics which will allow us to evaluate the best place to insert it in the schedule. We have to search windows which are large enough within time windows which are localized between the release-date and the due-date of the new task.

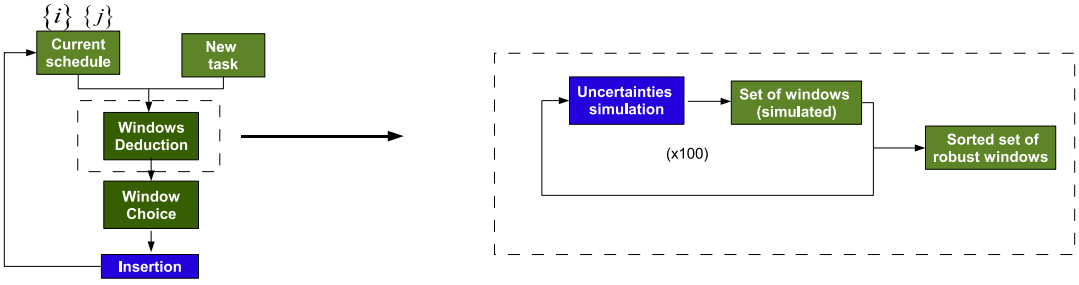

Figure 5: Task insertion methode

Windows will be sorted by their level of robustness. If there is no window large enough exists within the totally robust windows, we will extend our research through the most robust windows we can find. As described in figure 5, the deduction of the adequate insertion window will be done by variation simulation on the scheduling. The current planning and its possible variations will be simulated a hundred times in order to obtain data concerning the windows' robustness. During the hundred simulations the size of windows will vary. At each simulation, the procedure will check if the task could be inserted in the different windows of the schedule. During the hundred simulations, the number of times where it will be possible to insert the task in each window will be counted.

Figure 6 presents an example of insertion simulation of a new task (N) between two tasks (1 and 2). The first picture does not use variations. The second and the third ones show that variations may have an impact on the possibilities of insertion.

For each window we obtain a percentage (the number of time when the insertion was possible) which represents its robustness level. Windows are then sorted by there robustness level and grouped

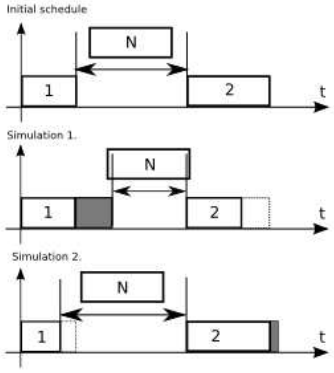

Figure 6: Insertion after variations

into various classes of equivalent level of robustness as follow : $[100\%]$, $]100\% - 90\%]$, $]90\% - 80\%]$, ..., $]20\% - 10\%]$, $]10\% - 0\%]$. Within the highest windows' set, of equivalent level of robustness, we choose the window which balances the load between resources and minimizes the total working duration. It corresponds to the window which, after insertion, will minimize the workload standard deviation.

In the following simulations, we compare two cases. The aim of the first one is to find the best location place to insert a new task as the maintenance manager would do. The choice will be made through the minimization of the standard deviation workload on the total of the proposed windows. The second one has the same objective but it will use our methodology by simulating disturbances on the existing planning. That means that we will have to find the most robust insertion windows for new tasks. The minimization of the standard deviation workload will be applied only on the set of the most robust windows.

After each task insertion the current planning is modified. Then, in case of a further insertion task, the newly obtained planning will be used in the same way as the current planning and the final inserted task will also present variations.

The fact that we take into account variations by anticipating the various disturbances and propose solutions signifies that our scheduling approach is proactive.

6 Results

In this section we present simulation results in order to describe the different methodologies previously presented.

6.1 The static problem

6.1.1 Data generation

We chose to use an algorithm like the Ibarra one (previously described), and to improve it for our problem[6]. This algorithm is called ECT: Earliest Completion Time. Results obtained by our algorithm and by the Ibarra one are presented in table 1. These averages have been calculated from 20 different problems. The number of tasks (n) and maintenance operators (m) were chosen to be representative of the reality. In table 1 the C_{max} columns contain the makespan obtained with both algorithms (in Time Unit); SD columns are Standard Deviation between the duration of the assignments of the different operators in each solution. The last column shows the average time per solution. We carried out a computational experiment on a Pentium IV 3.00GHz considering tests obtained by generating randomly the p_{ij} values. p_{ij} values are principally obtained by the combination of the basic tasks' duration which is an integer from the uniform distribution [1 , 16]. This duration is multiplied by the level of skill from the resource in the corresponding skill. For each task, a corresponding skill is

determined by an integer from the uniform distribution [1 , 3]. For each resource the level of competence in each domain is a real value generated from a uniform distribution [1.01 , 2.00]. This data is determined before the simulation. Considering the resources and the number of tasks , the complexity is then $O(n*m)$. Penalties are determined as integers from the uniform distribution [1 , 100]. They are assigned if the task treatment is finished after its due-date, which is also obtained following uniform distribution.
We used the algorithm in three cases: in the cases of low, medium and high load. These conditions are determined by the generation of the due-dates. For the same task and resource number, we create task due-dates for a near future. For a same task and resource number, we create task due-dates in a nearer future. In order to ensure that each task could be finished in time (depending on the scheduling), their due-date cannot be fixed before $t = now$ and $t_2 = now + 2 * p_j$ ("now" being the program launching date, in seconds). To regulate the load we modified the maximal limit value t_3 and then we obtained due-dates as reals from the uniform distribution $[t_2 , t_3]$. For the low load case $t_3 = t_2 + 720$ $t.u.$, in the medium load case: $t_3 = t_2 + 540$ $t.u.$ and in the high load case: $t_3 = t_2 + 360$ $t.u.$.

6.1.2 Computational results

- **Assignment algorithm :**

Table 1: Results of our method and of the ECT algorithm

		Our algorithm			ECT		
m	n	C_{max} *(t.u.)*	SD	time *(ms)*	C_{max} *(t.u.)*	SD	time *(ms)*
2	20	122	1.02	12.55	129.32	0.51	0.80
	30	185.87	0.57	11.65	190.79	0.45	1.50
	50	303.96	0.4	19.45	316.2	0.43	1.55
	100	595.9	0.36	43.85	625.9	0.4	7.80
	200	1224.31	0.33	138.35	1280.24	0.22	32.20
5	20	45.83	1.2	8.65	47.1	1.23	1.55
	30	67.75	0.96	10.15	70.15	0.72	2.35
	50	115.24	0.67	18.80	120.67	0.54	2.35
	100	225.87	0.51	34.60	244.11	0.47	12.50
	200	445.91	0.5	83.60	486.9	0.42	36.75
8	20	28	1.18	9.50	28.21	1.07	2.35
	30	41.91	1.15	10.20	42.74	0.87	0.80
	50	69.57	0.69	13.25	71.72	0.6	5.35
	100	139.56	0.56	36.05	147.65	0.5	14.85
	200	268.77	0.42	79.85	291.87	0.45	43.70

The standard deviation (SD) permits us to know, for the same set of data, if the load of each resource is close to C_{max}. In the case of an identical C_{max} for two different simulations: the bigger SD is, the more free time the operators (not concerned by the C_{max} duration) have for eventual new tasks. This problem is not a search for the optimal solution but for a good (figure 7) and fast answer. This solution will be re-organized using task time constraints which will also need calculation time. Then, we directed our research toward an heuristic. The lower bound (LB) used in the algorithm is not the best lower bound that could be used, because it is only reachable in certain rare and particular cases. A better lower bound would be globally higher and during the assignment heuristic, would allow more tasks to be assigned to the most efficient resource. However the maximal variation between LB and our solution varies only from 5% for a two-resource and twenty-task problem to 12% for an eight-resource and two hundred-task problem. This heuristic also presents an improvement of 8% compared to ECT

for the eight-resource and two hundred-task problem, which is a large-sized problem. This is logical because of the added treatment of this algorithm. That treatment time is slightly increased with our algorithm, but this is not perceptible for the program user, as it does not represent a problem.

- **Tardiness management :**

Table 2: Tardiness consideration

		Low load	Medium Load	High Load
H-EDD	$\sum U_i$	3	22	148
	$\sum w_i T_i$	173	1141	7301
	C_{max}	448	464	461
EDD-H-EDD	$\sum U_i$	33	71	136
	$\sum w_i T_i$	1656	3487	6841
	C_{max}	456	473	468
WSPT-H-EDD	$\sum U_i$	3	21	143
	$\sum w_i T_i$	162	1027	7142
	C_{max}	454	469	466

In the low and medium load cases, the WSPT-H-EDD heuristic presents the best results concerning the number of late tasks and the total of penalties, whereas, with a high load, the best results are given by the EDD-H-EDD.

6.2 The dynamic problem with imprecisions

6.2.1 Data generation

In this part, p_{ij} values are principally obtained by the combination of the basic tasks' duration (in time unit) which is an integer from the uniform distribution [1 , 7200]. This duration is multiplied by the level of skill of the resource of the corresponding skill. For each task, a corresponding skill is determined by an integer from the uniform distribution [1 , 3]. It refers for each resource to a level, which is a real from the uniform distribution [1.01 , 2.00], in this competence. Penalties are determined as integers from the uniform distribution [1 , 100]. They are assigned if the task treatment is finished after its due-date, which is also obtained following a uniform distribution.The release-dates r_j are obtained as reals from the uniform distribution [Now , 86400$u.t.$] (Now being the simulation launching time) and the due-dates d_j are obtained as reals from the uniform distribution [r_j+2*p_j , $r_j+2*p_j+86400$ $u.t.$].

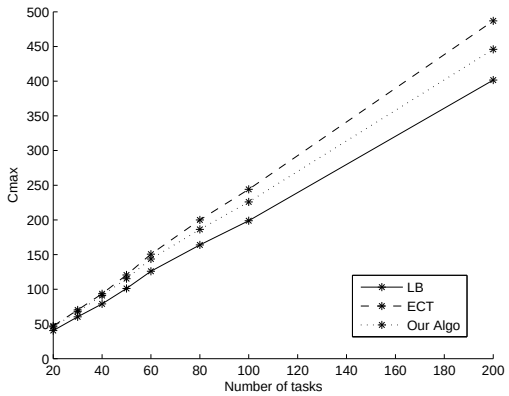

Figure 7: Evolution of the C_{max} on 5 human resources

6.2.2 Computational results

A classical schedule, which does not take into account the possible disturbance, will search the different windows and will obtain a certain number of positions. Our procedure considers the different disturbances as explained above which is why we will compare insertion propositions on the same problem instances with and without uncertainties.

6.2.3 Insertion of one task

On the first graph in figure 8, we compared the number of insertion windows which are proposed, in order to insert task k, randomly generated. Perturbations are generated on 35% of each task duration with a standard deviation of 30%. The number of tasks (n) and human resources (hr) were chosen to be representative of the reality. Results were obtained on a hundred computed instances.

Our objective was to dynamically insert tasks in a current schedule. However, we had a large set of possible choices. The real sizes of a window can be bigger than foreseen which is fine, but they could also be smaller which could cause a problem. If the manager had chosen an insertion place which is, in reality, smaller than imagined, one task or more would be late. That is why it is really necessary to consider possible disturbances when we have to dynamically insert a new task. In the observed case with five human resources and twenty-five tasks, on the first graph in figure 8, only 45% of the windows could be considered as robust.

On the second graph in figure 8 we observe total weighted tardiness obtained from three different case studies. Here we show the efficiency of the method with an increasing existent load in a schedule with five human resources. We based our study on three different load levels: fifty, sixty and seventy tasks already assigned in their schedule and generated within the same period. Results obtained are the averages of ten simulations of insertions in every kind of schedule. The interest of the method which has been shown here will now be completed with the case of successive insertions.

6.2.4 Insertion of ten different tasks

In a second time we dynamically inserted ten new tasks in an existing planning composed of five human resources and fifty tasks. Table 3 shows the results of five different current plannings. The comparison is the result of the average of a hundred simulations of disturbances on the final planning. We can observe that the total weighted tardiness is far less important with our methodology.

Table 3: Total weighted tardiness after the insertion of 10 new tasks.

Instance	1	2	3	4	5
Classical solution	97454	293580	355950	121168	152795
Robust solution	32065	68659	27958	47189	45220

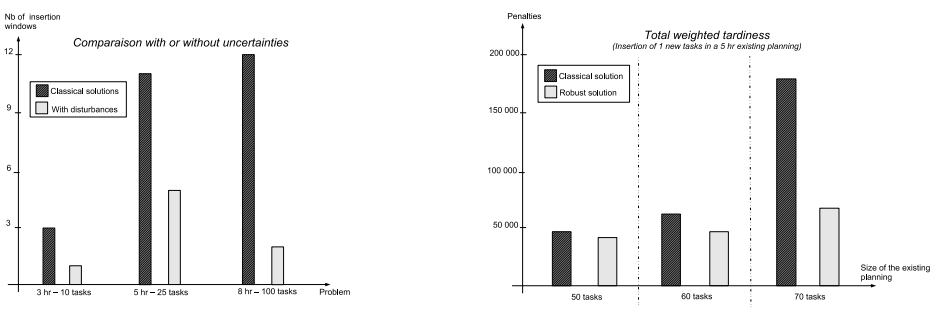

Figure 8: Robustness and flexibility results

We also checked the evolution of the total weighted tardiness, through the second example of table 3, presented in figure 9. When inserting the second task, results show that the location was not really effective. It comes from the fact that the simulation reflects the reality and even if we are in a proactive reasoning, the results could be different in reality. However, globally the results are better and after the insertion of the tenth task, results are nearly four times better with our method.

As regards the partial rescheduling methodology, we compared results obtained after ten dynamic new tasks were inserted. Weighted tardiness where a decrease of 69% compared to a complete re-scheduling of the initial scheduling (with the static scheduling method). The number of tasks which had a new assignment was reduced by 25%. This methodology has then to be privileged during the dynamic insertion phase to a task for which there is no evident solution to obtain a robust schedule. However if there is a large number of new tasks which has to be assigned (superior to the already assigned one) or if there is a new scheduling to be created from a list, the static methodology will remain more.

7 Conclusion

As already mentioned, this work has permitted to assign tasks to maintenance operators under skill constraint. Each task that has to be performed is characterized by a required competence. The answer to this assignment and scheduling problem leads to finding the right resource and the corresponding time to do the task.
Firstly, we developed an approach to assign tasks to minimize the makespan. It was realized for the tasks which are in the medium-term horizon before each shift of the horizon. A good maintenance workforce plan considers each operator and his skill in order to determine the strategy to put into place to optimize the resources workload. We have presented here some of the numerical results obtained. Due to the method the results were rapidly obtained and close to the optimum. We have also considered the number of late tasks and the tardiness penalties by using different list algorithms. Secondly, we observed the effect of uncertainties on existing schedules in an unrelated parallel machine context. In order to dynamically insert new tasks in a current schedule, we worked on the proactivity to find the set of robust places. We showed, through examples that the consequences of a bad choice (a non robust window) for dynamic insertions could, in case of variations, induce lateness. By inserting consecutive tasks in a current planning, we confirmed that correct results previously obtained on one dynamic insertion were valid and necessary in the cases of multiple insertions, which is closer to the reality. The fact of choosing insertion windows by considering uncertainties, is a contribution in order to anticipate and to minimize possible lateness. By minimizing the workload standard deviation between resources, we developed an approach which permits to balance the load between resources but also to minimize the total working duration.
We completed this dynamic task insertion methodology with an approach which partially modifies the

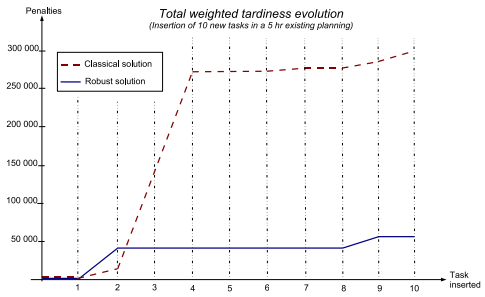

Figure 9: Total weighted tardiness evolution

existing scheduling. It enables to improve the solution concerning weighted lateness by moving some tasks which have been compared to the static heuristic and shows that it gives comparable results. It has mainly for interest to reduce the employees disturbances in the case of too frequent changes of schedule.

References

[1] Adolfo Crespo-Marquez and Jatinder N.D. Gupta. Contemporary maintenance management: process, framework and supporting pillars. *Omega*, 36:313–326, 2006.

[2] S.L. Van de Velde. Duality-based algorithms for scheduling unrelated parallel machines. *ORSA Journal of Computing*, 5:192–205, 1993.

[3] Carl Esswein, Jean-Charles Billaut, and V.-A. Strusevich. Two-machine shop scheduling: compromise between flexibility and makespan value. In *European Journal of Operational Research*, 2005.

[4] B. Grabot and A. Letouzey. Short-term manpower management in manufacturing systems : new requirement and dss prototyping. *Computers in Industry*, 43:11–29, 2000.

[5] A. Hariri and C. N. Potts. Heuristics for scheduling unrelated parallel machines. *Computers & Operations Research*, 18(3):323–331, 1991.

[6] O. Ibarra and C. Kim. Heuristic algorithms for scheduling independent tasks of non identical processors. *Journal of the Association for Computing Machinery*, 24:280–289, 1977.

[7] J. K. Lenstra, D. B. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated machines. *Mathematical programming*, 46:259–271, 1990.

[8] Silvano Martello, François Soumis, and Paolo Toth. Exact and approximation algorithms for makespan minimisation on unrelated parallel machines. *Discrete Applied Mathematics*, 75:169–188, 1997.

[9] E. Mokotoff and P.Chrétienne. A cutting plane algorithm for the unrelated parallel machine scheduling problem. *European Journal of Operational Research*, 141:515–525, 2002.

[10] Michele Pfund, John W. Fowler, and Jatinder N. D. Gupta. A survey of algorithms for single and multi-objective unrelated machine deterministic scheduling problems. *Journal of the Chinese Institute of Industrial Engineers*, 21(3):230–241, 2004.

[11] Mickael Pinedo. *Scheduling, Theory, Algorithms and Systems.* Prentice, 1995.

[12] H.A. Simon. *Administrative behaviour: a study of Decision Making Processes in Administrative Organizations.* Mac Millan, New York, 1947.