# Scheduling Jobs and Preventive Maintenance Activities on Parallel Machines

Maher Rebai
University of Technology of Troyes
Department of Industrial Systems
12 rue Marie Curie, 10000 Troyes
France
maher.rebai@utt.fr

Imed Kacem
University Paul Verlaine Metz
Department of Informatique
Ile du Saulcy 57000, Metz
France
kacem@univ-metz.fr

Kondo H.Adjallah
Ecole Nationale d'Ingnieurs de Metz
Electrical & Computer Engineering Dpt
Ile du Saulcy 57045, Metz
France
adjallah@enim.fr

*Abstract:* We propose in this study a hierarchical method for the problem of scheduling *N* jobs on *M* parallel machines where each machine should be maintained once during the planning horizon. The maintenance tasks should be continuously executed because the maintenance resources are not sufficient. Our objective is to find a schedule composed of the jobs and the maintenance tasks in which the total sum of the job's weighted completion times and the preventive maintenance cost are simultaneously minimized.
The proposed hierarchical method is essentially based on a linear model and an Evolutionary algorithm. Computational experiments are performed on randomly generated instances. The results show that the proposed method is able to produce appropriate solutions for the problem.

*Key–Words:* Linear programming, lower bound, Evolutionary algorithm.

## 1 Introduction

In most manufacturing and industries shops, the preventive maintenance operations are necessary to keep processing equipments in well working conditions. According to our industry experience, the preventive maintenance cost of a maintenance task depends on the start execution time of the maintenance task. It is considered minimal only if the task is performed at an instant between two deadlines defined for the task: the optimistic deadline and the pessimistic deadline. When ever the maintenance task begins before the optimistic deadline or after the pessimistic deadline, the preventive maintenance cost associated to the task increases. Sometimes, the resources ensuring the maintenance are expensive. Hence, it is not possible to find more than one example of maintenance resource ensuring the maintenance of the entire machines of the shop. In such a situation, when several preventive maintenance deadlines arrive, just one machine should be taken for preventive maintenance and the remaining machines should continue working for different time periods without undergoing the preventive maintenance activities. Generally, this behavior provides additional maintenance costs related to the delayed preventive maintenance tasks. Indeed, when a preventive maintenance task begins after its pessimistic deadline, two important facts frequently occur: The first one is related to the damages that may affect the machine's components. As consequence, the machine performance may decrease and the failure risk increases. The second fact is related to the defect products'rate that may quickly rise. The total damage cost of the maintained machines may be optimized by establishing a maintenance plan for which we define the starting time of each maintenance task. Because many maintenance plans may exist, it is associated to each one a specific machines availabilities plan. Hence, to optimize the global solution of scheduling maintenance tasks and jobs, it is recommended to jointly schedule the maintenance activities and the jobs. In the literature, few researchers were interested with the problem of jointly schedule jobs and maintenance activities. Lee et al.[9] treated two cases for this problem. In the first case, they consider sufficient maintenance resources. In the second case, the maintenance resources are not sufficient so that just one machine may be maintained at a time. Both cases are shown NP-hard and solved by a branch and bound algorithm based on column generation approach. Results show that the branch and bound algorithm is capable to solve optimally medium sized problems within a reasonable computational time. Graves et al [5] solved the problem of scheduling a set of jobs on a single machine using a dynamic programming approach. The machine should be maintained in certain intervals and when a job is not completely handled before the machine is turned off for maintenance, a set up time is needed before restart the treatment. The

authors optimize two criteria: the total weighted jobs completion times and the maximum delay. Aghezzaf et al [1] are interested with the batch production type problem. Random failures may affect the production system and at each maintenance intervention the production system is turned off. In their study, they aim to determine a plan for which the cost of production and maintenance is minimized. The problem was solved using a linear programming approach.
In this article we aim to find a schedule composed of *N* jobs and *M* maintenance tasks for which the sum of the weighted completion times of the jobs and the preventive maintenance tasks' cost are minimized. The preventive maintenance cost corresponds to the total early-tardy cost of the preventive maintenance tasks. The tardy cost is the damage cost caused by tardy execution of a preventive maintenance operation. Similarly, the early cost is the consequence of early execution of a preventive maintenance activity. In other words, the early cost corresponds to the part of the process equipment that is not efficiently used. The remainder of this paper is organized as follows: in section 2, a description of the problem and necessary notations are presented. In section 3, we describe the hierarchical method that is essentially based on a linear model and an evolutionary algorithm. In section 4, computational results will be discussed. Finally we conclude by summarizing the main proposals presented in this paper.

# 2 Problem description and notations

The considered problem consists of scheduling a set of *N* jobs on *M* parallel machines where each machine should be maintained once during the planning horizon. A job *i* has a processing time $p_i^p$ and a weight $w_i^p$. For each machine *j*, corresponds a preventive maintenance task *j* characterized by a processing time $p_j^m$, an optimistic deadline $d_{j1}$, a pessimistic deadline $d_{j2}$, greater or equal to $d_{j1}$, a tardy weight $w_j^m$, an early weight $h_j^m$ and a minimal preventive maintenance cost $C_{j0}^m$. For the reason that the preventive maintenance resources are not sufficient, the running of the preventive maintenance tasks on the machines should be continuous during the planning horizon. In fact, when a preventive maintenance task *j* begins at an instant from the $[d_{j1}\ d_{j2}]$ interval, the preventive maintenance cost is minimal and equal to $C_{j0}^m$. Otherwise, if the preventive maintenance task *j* starts before its optimistic deadline $d_{j1}$, the resulting preventive maintenance cost increases and will be equal to $h_j^m(d_{j1} - t_j) + C_{j0}^m$ where $t_j$ is the starting time of the task *j*. Similarly, when the preventive maintenance task *j* begins after its pessimistic deadline $d_{j2}$, the preventive maintenance cost will be equal to $w_j^m(t_j - d_{j2}) + C_{j0}^m$. Our objective in this study is to provide a schedule composed of the *N* jobs and the *M* preventive maintenance tasks for which the total weighted completion times of the *N* jobs and the preventive maintenance cost are simultaneously minimized.

# 3 The hierarchical method

In this section we describe our proposed hierarchical method used to produce an upper bound for the problem described above. The method consists of decomposing the original problem into two sub-problems. The first sub-problem concerns the maintenance plan for which the preventive maintenance cost should be minimized by solving optimally the early-tardy cost minimization problem on a single machine $(1|d_{j1}d_{j2}|\sum h_iE_i + w_iT_i)$. The elaborated solution for this sub-problem should provide all the starting times of the maintenance tasks allowing a minimal preventive maintenance cost. In other words, by solving this sub-problem, the availabilities of the machines on the entire planning horizon are obtained. Given the availabilities of the machines, we move to solve the second sub-problem of scheduling the *N* jobs on *M* parallel machines with availabilities constraints $(P_m, h_{j1}||\sum w_iC_i)$. The sum of the two sub-problem solutions gives an upper bound solution for the original problem.

## 3.1 Linear model for $1|d_{j1}d_{j2}|\sum h_iE_i + w_iT_i$

In this sub-section we present a linear formulation for the problem of $1|d_{j1}d_{j2}|\sum w_iE_i + h_iT_i$. We remain that the main role of this linear model is to determine the starting times of the maintenance tasks allowing a minimal preventive maintenance cost. The proposed formulation is a time-index formulation. The main idea on which based this formulation consists of decomposing the planning horizon, noted *T*, in time slots where each time slot starts at time t and ends at time it t+1. According to J.M. van den Akker [21], the major advantage of using this time-index formulation is that the linear relaxation obtained by dropping the constraint of variables integrity provides generally a strong lower bound which dominates the linear relaxations of other mixed integer programming formulations based on other decision variables. A main disadvantage of this formulation is that its linear relaxation is sometimes time consuming especially when the planning horizon is big. For our problem, let $x_{it}$ be a binary variable taking *1* if the job *i* starts at time *t* and 0 Otherwise. Let us define $t_i$ the starting time of the preventive maintenance task *i*, $T_i$ the tardiness that can be happen and $E_i$ the earliness that may happen . The proposed linear formulation for the early-tardy cost minimization on a single machine problem is:

$$Min \sum_{i=1}^{M} h_i E_i + w_i T_i$$

Subject to

$$\sum_{t=1}^{P-p_i+1} x_{it} = 1 \qquad \forall i = 1 \dots M \quad (1)$$

$$\sum_{i=1}^{M} \sum_{s=Max(t-p_i+1,1)}^{t} x_{is} \leq 1 \qquad \forall t = 1 \dots P \quad (2)$$

$$\sum_{t=1}^{P-p_i+1} t x_{it} + E_i \geq d_{j1} + 1 \qquad \forall i = 1 \dots M \quad (3)$$

$$\sum_{t=1}^{P-p_i+1} t x_{it} - T_i \leq d_{j2} + 1 \qquad \forall i = 1 \dots M \quad (4)$$

$$P = \sum_{i=1}^{N} p_i^p + \sum_{j=1}^{M} p_j^m;\ x_{it} \in \{0,1\}\ \forall i = 1 \dots M \quad (5)$$

In this model, the equations (1) ensure that each job is processed once. The second inequalities are resources constraints indicating that at most just one job can be handled at a time slot. The third and the fourth inequalities decide for each maintenance task to start early, tardy or on time. Finally, the fifth constraints are integrity constraints. For this formulation, the MIP solver ILOG Cplex 10.1 is able to solve all our small instances with a size less or equal to 5 maintenance tasks in a time period less than 3 seconds.

## 3.2 An evolutionary algorithm for $P_m, h_{j1} || \sum w_i C_i$

Evolutionary algorithms are probabilistic algorithms used to provide excellent solutions in reasonable time for hard and time complex problems. In a genetic algorithm, many new solutions are created at each iteration (called also generation). A new obtained solution (called also a child solution) emanates from the crossing of two solutions from the previous generation (called parents solutions). Sometimes, the child solution may undergo a mutation operation before it passes for evaluation (fitness). Generally, a genetic algorithm is defined by the correspondingly elements: Initial population, cross-over operation, mutation operation and the replacement strategy.

*The initial population*

The initial population consists of a set of $N_p$ initial solutions called individuals or chromosomes. The chromosomes of a population are usually randomly generated. Sometimes they are determined by specific rules. Generally, the population's size should not be large to not increase the computational time of obtaining the final solution. On the other hand, it should not be small to not affect the quality of the final solution. For our proposed genetic algorithm, we generate an initial population of size of *100* chromosomes *(Np=100)*. Each chromosome is randomly generated and represented by a table of size of *N* genes. A gene corresponds to a machine on which the job having as number the index of the gene in the table is executed. The following figure shows the used chromosome's representation:

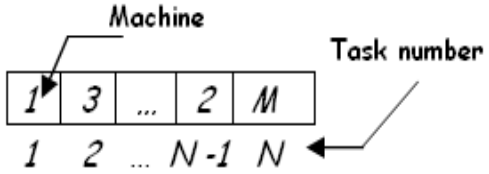


**Fig1.** *Chromosome representation*

This chromosome's representation is translated as follows in a solution for the problem : We first establish the *WSPT* sequence. After that, we test if the first job of the *WSPT* sequence *(job 1)* may be assigned before the maintenance task of the machine determined by the first gene in the table. If it is possible, we assign it. If it is not possible we assign it on the same machine after the maintenance task. We repeat the procedure until assigning the task *N* of the *WSPT* sequence.

*The crossover operation:*

In the cross-over operation, two parents should be selected from the population of the current generation to generate at least one child. In our genetic algorithm, only 90% of the best chromosomes participate in the crossover operation. At each cross-over operation, two children are produced. The principle of parent's selection is as follows: First the $1_{st}$ parent and the $45_{th}$ parent are selected. After that, the second and the $46^{th}$ parents, etc... until the $44^{th}$ parent and the $90^{th}$ parent . After the selection of each two parents, we randomly generate an integer *k* from the uniform [*1, N*].The first child $C_1$ inherits the subsequence [*1,... , k*] of its genes from the first parent and its remaining empty genes [*k+1, ... , N*] from the second parent. The second child $C_2$ inherits the subsequence [*1, ... , k*] of its genes from the second parent. After that, we fill its remaining empty genes [*k+1,... , N*] from the first parent. The cross-over operator used to generate the children in our genetic algorithm is called one opts cross-over operator.

*The mutation operation:*

In the mutation operation, an individual is randomly chosen from the population to undergo a small modification. In our algorithm, we apply the mutation on ten individuals randomly chosen from the population. The slight modification consists of permutation of two genes randomly selected.

*The replacement strategy:*

In a genetic algorithm, the number of new solutions increases from a generation to another. Keeping all generated solutions of the iterations should amplify the population size. This fact may increase the computational time for obtaining the final solution and also the algorithm may be memory consuming. A replacement strategy that consists of replacing the individuals (solutions) in the population

with worst fitness by the new children with better evaluations may be a good solution for a better performance of the algorithm. In our implementation, we have taken at each generation all the best *100* chromosomes.

### 3.3 Lower bound for $P_m, h_{j1}||\sum w_iC_i$

The proposed lower bound is based on a Lagrangian relaxation technique where Lagrangian multipliers are identified by a multiplier adjustment algorithm. The main principle is that the maintenance tasks will be considered as new jobs for which the weights should be computed and the processing times are the maintenance tasks' durations. By computing a good quality lower bound for this new problem and subtracting the sum of the weighted completion times of the maintenance tasks computed in the first sub-problem, we obtain a lower bound value for $P_m, h_{j1}||\sum w_iC_i$

**Definition 1** *Let consider $P_1$ the problem of scheduling N jobs on M parallel machines. Each job i (i = 1...N) has a processing time $p_i^p$ and a weight $w_i^p$ while each machine j is not available during the time period [$T_{j1}T_{j2}$] for the reason of preventive maintenance.*

**Definition 2** *Let $P_2$ denotes the problem of scheduling N+M tasks on M parallel machines available all the time. The N tasks correspond to the N jobs of $P_1$ while the M tasks (N+1,...,N+M) are the maintenance tasks with processing times equal to the unavailability periods of the machines ($p^m_{N+j} = T_{j2} - T_{j1}$) and with weights $w^m_{N+j}$ (j=1,...,M) to be determined. Let W be the weights vector of the new M jobs* ($W = (w^m_{N+1}, \ldots, w^m_{N+M})$).

**Lemma 3** *If $\gamma(P_2(W))$ be a lower bound for the problem $P_2$ ($P_m||\sum w_iC_i$), therefore the following expression is a lower bound for $P_1$: LB(W)=$\gamma(P_2(W))$-$\sum_{j=1}^{M} w_jT_{j2}$.*

This bound has been introduced for the single machine problem by Kacem and Chu[6]. Many lower bounds exist in the literature for the problem $P_2$. In this article we will use the lower bound proposed by Eastman et al. [3]. It is computed by the following expression: $\gamma(P_2(W)) = \frac{1}{M}(\sum_{i=1}^{N} w_i^p C_i^p(WSPT) + \sum_{j=1}^{M} w^m_{N+j} C^m_{N+j}(WSPT)) + \frac{M-1}{2M}(\sum_{i=1}^{N} w_i^p p_i^p(P_2) + \sum_{j=1}^{M} w^m_{N+j} p^m_{N+j}(P_2))$ where $C_j^p(WSPT)$ is the completion time of the job *j* in the *WSPT* sequence and $C_j^m(WSPT)$is the completion time of the maintenance task *j* in the same *WSPT* sequence. According to Lower bound formula, we first need to identify the Lagrangian multiplier vector *W*. After that, we should determine the *WSPT* order of the *N+M* jobs. Finally, we compute the $\gamma(P_2(W))$ value from which we subtract $\sum_{j=1}^{M} w_jT_{j2}$ to obtain the lower bound value of our second sub-problem . Many algorithms exist in the literature used to compute the best value of *W* allowing a good quality lower bound for $P_2$. The most known methods are the sub-gradient method and the multiplier adjustment method. In our implementation, we have used a multiplier adjustment method.

*Identification of Lagrangian multipliers:*
The identification of Lagrangian multipliers is an essential step in a lower bound's computation process. To obtain a satisfactory quality lower bound, it is important to identify interesting values for the Lagrangian multipliers vector *W* which maximize the lower bound function. Therefore, to maximize *LB(W)*, we need a sequence composed of the *(M+N)* jobs for which the *W* vector provides a $\gamma(P_2(W))$ value as maximum as possible. The computed Lagrangian multipliers should also satisfy, for each maintenance task *i*, the conditions of the *WSPT* order. In other words, the ratio $\frac{p_i^m}{w_i^m}$ of a maintenance tasks *i* should not exceed the ratio $\frac{p_k^p}{w_k^p}$ of the first job *k* after the maintenance task *i* and should not be less than the ratio $\frac{p_{k'}^p}{w_{k'}^p}$ of the first job that precedes the maintenance task *i*. By these restrictions, the Dual Lagrangian problem of the relaxed problem can be written as follows:

$$MaxZ = \frac{1}{M}\sum_{i=1}^{M} w_i^m C_j^m(WSPT) + \frac{M-1}{2M}\sum_{i=1}^{M} w_i^m p_j^m$$

*Subject to*

$$w_i^m \geq \frac{p_i^m w_k^p}{p_k^p} \qquad \forall i = 1 \ldots M \qquad (6)$$

$$w_i^m \leq \frac{p_i^m w_{k'}^p}{p_{k'}^p} \qquad \forall i = 1 \ldots M \qquad (7)$$

$$w_i^m \geq 0 \qquad \forall i = 1 \ldots M \qquad (8)$$

$C_i^m$ corresponds to the completion time of the maintenance task *i* in the *WSPT* sequence.

# 4 Computational results

To define a problem, we first need to identify the number of jobs *N* and the number of machines *M* that corresponds to the number of maintenance tasks. In our implementation, the number of jobs $N \in \{50, 100, 150, 200, 300\}$ while the number of machines $M \in \{1, 2, 3, 4\}$. For the jobs, the weights are randomly generated from the discrete uniform distribution [*1,...,10*] and the processing times are randomly chosen from the discrete uniform distribution [*1,...,50*]. To generate a maintenance task *i*, two parameters are needed: the range factor *R* and the tardiness factor *T*. By using these parameters, the optimistic deadlines $d_{i1}$ are generated from the uniform distribution [$d_{min}$,...,$d_{min}$ + $P_{mean}$], where $d_{min}$ = $max\{0, x(T - \frac{R}{2})\}$ and $P^m_{mean} = \frac{\sum_{i=1}^{M} p_i^m}{M}$. The pessimistic deadlines $d_{i1}$ are generated from the uniform distribution [$d_{i1}, \ldots, d_{i1} + P_{mean}$]. The processing times for each maintenance task *i*, are randomly determined from the discrete uniform distribution [$0.5P^p_{mean}$,...,$2P^p_{mean}$ ] where $P^p_{mean}$ corresponds to mean of the job's processing times ($\frac{\sum_{i=1}^{N} p_i^p + \sum_{i=1}^{M} p_i^m}{N}$).

Finally the earliness and the tardiness penalties are drawn from the uniform distribution [*1,...,10*]. 5 instances have been generated for each combination of *N, M, T and R*. The algorithms were coded in *C* language and implemented on a Pentium IV-500 personal computer using concert technology technique with Cplex 10.1 to solve the Dual Lagrangian model of the proposed lower bound.

Table 1. Mean gaps value between the Genetic Algorithm and the lower bound

| M | N | Mean Gap (%) | Max Gap(%) | Min Gap(%) |
|---|---|---|---|---|
| 2 | 50 | 0.4158 | 1,97 | 0,08 |
| | 100 | 0,1257 | 0,47 | 0,03 |
| | 200 | 0,0401 | 0,18 | 0,01 |
| 3 | 50 | 0,9977 | 2,22 | 0,40 |
| | 100 | 0,3508 | 1,05 | 0,15 |
| | 200 | 0,1266 | 0,29 | 0,06 |
| 4 | 50 | 1,7388 | 3,33 | 0,81 |
| | 100 | 0,6775 | 1,41 | 0,33 |
| | 200 | 0,2647 | 0,50 | 0,10 |
| 5 | 50 | 2,7841 | 5,24 | 1,36 |
| | 100 | 1,0929 | 1,78 | 0,59 |
| | 200 | 0,4460 | 0,88 | 0,23 |

The first column of table 1 represents the machine number *M*. The second column represents the jobs number *N*. The third, the fourth and the fifth columns are respectively the average gap between the solution of the genetic algorithm and the based Lagrangian lower bound, the maximum found gap and the minimum found gap.

From column 3, we can observe that for all produced upper bounds, the average gap decreases when the job number increases. From column 4 and 5, the maximum observable gap is equal to 5,24% while the minimum gap is equal to 0,01%. Hence, we can conclude that both proposed genetic algorithm and lower bound produce good quality upper and lower bounds values for the second sub-problem for all instances.

Table 2. Mean computational time for obtaining a Genetic Algorithm solution

| M | N | Time (s) |
|---|---|---|
| 2 | 50 | 0,174 |
| | 100 | 0,288 |
| | 200 | 0,592 |
| 3 | 50 | 0,202 |
| | 100 | 0,363 |
| | 200 | 0,707 |
| 4 | 50 | 0,218 |
| | 100 | 0,392 |
| | 200 | 0,782 |
| 5 | 50 | 0,236 |
| | 100 | 0,427 |
| | 200 | 0,826 |

According to this table, we observe that for all generated instances, the maximum mean computational time does not exceed 1 second. Hence, we can confirm that the proposed genetic algorithm is efficient in terms of quality solution and computational time.

# 5 Conclusion

In this paper we have proposed an upper bound for the problem of scheduling *N* jobs on *M* parallel machines where each machine should be maintained once during the planning horizon. We have simultaneously minimized the total sum of the job's weighted completion times and the preventive maintenance cost. The proposed upper bound solution is obtained by a hierarchical method that consists of decomposing the problem into two sub-problems: The first sub-problem is called earliness-tardiness minimization problem with a particular framework costs. The second sub-problem is the problem of scheduling jobs on parallel machines. For the first sub-problem, we have proposed a linear model that allows the availabilities of the machines during the planning horizon with a minimal maintenance cost. For the second sub-problem, a Genetic algorithm and lower bound were proposed. Computational results show that the genetic algorithm and the lower bound for the second subproblem of the hierarchical method produce excellent initial solutions that may be used to perform the B&B algorithm.

**Acknowledgements:** This work has been funded by Conseil Régional Champagne Ardenne.

*References:*

[1] EH. Aghezzaf, MA. Jamali and D. Ait-Kadi, An integrated production and preventive maintenance planning model, *European Journal of Operational Research.* 181, 2007, pp. 679–685.

[2] EH. Aghezzaf and NM. Najid, Integrated production planning and preventive maintenance in deteriorating production systems,*Information Sciences.* 178, 2008, pp. 3382-3392.

[3] WL. Eastman, S. Even and IM. Isaacs, Bounds for optimal scheduling of n jobs on m processors,*Management science* 11, 1964,pp. 268-279.

[4] R. Mellouli, S. Cherif, C. Chou and I. Kacem, Identical parallel-machine scheduling under availability constraints to minimize the sum of completion times,*European Journal of Operational Research* 197, 2009, pp. 1150-1167.

[5] HG. Graves, C-Y. Lee, Scheduling Maintenance and Semiresumable Jobs on a Single Machine,*Naval Research Logistics* 46, 1999, pp. 845-863.

[6] I. Kacem, C. Chengbin, S. Ahmed, Single-machine scheduling with an availability constraint to minimize the weighted sum of the completion times,*Computers & Operations Research* 35, 2008,pp. 827- 844.

[7] W. Kubiak, J. Blazewicz, P. Formanowicz, J. Breit and G. Schmidt, Two-machine flow shops with limited machine availability,*European Journal of Operational Research* 136, 2002, pp. 528-540.

[8] C-Y. Lee, Minimising the makespan in the two machine scheduling scheduling problem with an availability constraint,*Operational Research Letters* 20, 2000, pp. 129-139.

[9] C-Y. Lee, Z-L. Chen, Scheduling Jobs and Maintenance Activities on Parallel Machines,*Naval Research Logistics* 47, 2000, pp. 145-165.

[10] G. Li, Single machine earliness and tardiness scheduling,*European Journal of Operational Research* 26, 1997, pp. 546-558.

[11] C-F. Liaw, A branch and bound algorithm for the single machine earliness and tardiness scheduling problem, *Computers & Operations Research* 26, 1999, pp. 679-693.

[12] F. Sourd, S. Keded-Sidhoum and Y. Rio Solis, Lower bounds for the earliness-tardiness scheduling problem on parallel machines with distinct due dates, *European Journal of Operational Research* 189, 2008, pp. 1305-1319.

[13] W.E. Smith, Various optimizers for single-stage production, *Naval Research Logistics Quarterly* 3,1956, pp. 59-66.