J Intell Manuf (2012) 23:1207–1224
DOI 10.1007/s10845-010-0425-0

# Earliness–tardiness minimization on a single machine to schedule preventive maintenance tasks: metaheuristic and exact methods

**Maher Rebai · Imed Kacem · Kondo H. Adjallah**

Received: 24 April 2010 / Accepted: 17 June 2010 / Published online: 13 July 2010


**Abstract** In this paper, we consider the problem of scheduling a set of $M$ preventive maintenance tasks to be performed on $M$ machines. The machines are assigned to execute production tasks. We aim to minimize the total preventive maintenance cost such that the maintenance tasks have to continuously be run during the schedule horizon. Such a constraint holds when the maintenance resources are not sufficient. We solve the problem by two exact methods and meta-heuristic algorithms. As exact procedures we used linear programming and branch and bound methods. As meta-heuristics, we propose a local search approach as well as a genetic algorithm. Computational experiments are performed on randomly generated instances to show that the proposed methods produce appropriate solutions for the problem. The computational results show that the deviation of the meta-heuristics solutions to the optimal one is very small, which confirms the effectiveness of meta-heuristics as new approaches for solving hard scheduling problems.

**Keywords** Scheduling · Maintenance · Linear programming · Branch and bound · Genetic algorithm (GA) · Local search

M. Rebai
ICD-LOSI CNRS FRE 2848, University of Technology of Troyes, Troyes, France
e-mail: maher.rebai@utt.fr

I. Kacem (✉)
LITA, University Paul Verlaine Metz, Metz Cedex, France
e-mail: kacem@univ-metz.fr

K. H. Adjallah
LGIPM, Ecole Nationale d'Ingénieurs de Metz, Metz, France
e-mail: adjallah@univ-metz.fr

## Introduction

In numerous manufacturing and service industries situations, the used equipments are sometimes blocked for reasons of preventive maintenance. As example of preventive maintenance activity, we can cite the exchange of a car or machine motor oil. Certainly, this kind of preventive maintenance operation is important to keep the equipment in well working conditions. Moreover, it should be carried out after an optimistic date and should not exceed a pessimistic date. A situation may frequently occur that is, several preventive maintenance deadlines of many equipments arrives, and at the same time it is not possible to do all the preventive maintenance activities due to the expensive cost of having the preventive maintenance resources available. The equipments may continue to work without undergoing the preventive maintenance activities. However, delaying the preventive maintenance may cause damages to the equipment's components. The minimization of the total damage cost of the maintained equipments can be solved by applying the Operations Research techniques. Indeed, this problem can be considered as a scheduling problem on a single machine: the machine represents the preventive maintenance resource that cannot handle more than one task at a time; the tasks correspond to the preventive maintenance operations and finally the considered criterion should be the minimization of the total earliness–tardiness cost of the preventive maintenance tasks. Here, the tardiness cost clearly represents the damage cost caused by tardy execution of preventive maintenance operations. Similarly, the earliness cost is the consequence of starting early the preventive maintenance operations. Hence, the earliness cost value corresponds to the part of the system that is not efficiently used. In the cited example, if the oil is changed before the optimistic deadline then, the system (in particular, the oil) is considered not efficiently used because

it would be better to continue to work with the old resource until an instant between the optimistic and the pessimistic deadlines.

This studied problem differs from the classical earliness–tardiness scheduling problem on a single machine in a major fact. The fact is that each preventive maintenance job has two due dates instead of one: an *optimistic date* and a *pessimistic date*. Each instant from the time period between the two due dates is a good starting time for the preventive maintenance task. In the literature, numerous studies have been published on the single machine scheduling problem involving both job earliness and tardiness and considering only one due date per job. Given the aim of this paper, we recall some references related to our work.

The first related work was presented by Held and Karp (1962). They proposed a dynamic programming approach. The authors considered the subset of jobs as the state-space and the recursion consists in adding a job in passing from a state to another.

Azizoglu et al. (1991) proposed a branch and bound algorithm for the weighted earliness–tardiness minimization problem. The lower bound is obtained by relaxing the completion time variable in the objective function and the upper bound is determined by a local search technique. Their results showed that the upper bound slightly exceeds the optimal solution computed by the branch and bound algorithm.

Abdul-Razaq and Potts (1988) developed a branch and bound algorithm with lower bounds obtained by relaxing the state-space of a dynamic programming technique and performing the recursion on the new obtained state-space.

Li (1997) used a branch and bound algorithm to solve short sequences (<50 jobs). His exact algorithm is based on a neighbourhood search heuristic and a lower bound determined by decomposing the main problem into two sub-problems, relaxing a constraint in each sub-problem and solving their Lagrange duals. Indeed, the main problem was decomposed into two sub problems ($1\,|d_i|\sum w_i T_i$) and ($1\,|d_i|\sum h_i E_i$). Each of these sub-problems was formulated by a linear model for which a constraint is relaxed. The resolution of the two Lagrange duals problems by tow efficient adjustment multiplier approaches gives two lower bounds. The lower bound of the main problem is obtained by summing the two obtained lower bounds. Computational results indicate that the branch and bound algorithm can easily produce optimal solutions for small problems. The results also show that local search heuristic is not only effective, but also strong.

Liaw (1999) also proposed a branch and bound algorithm. The upper bound of the algorithm is obtained by computing in a first step an initial solution using the priority rule of Ow and Morton (1989). After that, the initial solution will be improved by an insertion procedure followed by a permutation procedure. The lower bound is based on a Lagrangian relaxation that decomposes the relaxed problem into tow sub-problems. The obtained results show that the branch and bound algorithm is very effective even for problems of 50 jobs.

M'Halla (2006) solved the single machine earliness–tardiness scheduling problem by hybridizing Genetic Algorithm with Hill Climbing and Simulated Annealing using two levels of hybridization: the low level and the high level. In the low level hybridization, she augmented the ability of the Genetic Algorithm to perform a Local Search by replacing its classical mutation operator by Hill Climbing. In the high level relay hybridization, she acted both on the initial population and on the best individuals of each generation. She used three greedy heuristics to generate the initial population, and refined the best solutions of each generation using Simulated Annealing.

Sourd et al. (2008) proposed recently a branch and bound based on a Lagrange relaxation of the resources constraints in a time-indexed formulation. They also added new dominance properties for solving problems of 50-jobs-size.

The reader is referred to Baker and Scudder (1990) for other studies on the single machine earliness and tardiness scheduling problem and its extensions.

Other scheduling problems joining the production and maintenance activities were also studied in the literature. Following some works of them treating the stochastic case (a breakdown) or the deterministic case (preventive maintenance).

Lee (1996) considered the deterministic case of the two-machine flowshop scheduling problem with an availability constraint. He developed a pseudo-polynomial dynamic programming algorithm to minimize the makespan. He also proposed an $O(n \log n)$ time heuristic algorithm to solve the problem with an availability constraint imposed on machine 1, and another heuristic algorithm with the same time complexity to solve the problem with an availability constraint imposed on machine 2. The worst case error bound for the first algorithm is 1/2 and the second one is 1/3.

The same problem was considered by Kubiak et al. (2002). He proposed a branch and bound algorithm based on important properties of the optimal schedule.

Lee and Chen (2000) considered in their work the problem of scheduling a set of production tasks that must be carried out on a park of machines where every machine must be maintained once during the production horizon. The aim is to determine a schedule for the production tasks and the maintenance activities so that the total weighted completion times of the jobs is minimized. Two cases are studied. In the first case, many machines can undertake maintenance operations at the same time due to the sufficient resources. In the second case, only one machine can be maintained. Both cases were shown to be NP-hard and solved by a branch and bound algorithm based on a column generation approach. Results showed that the branch and bound algorithms are capable to

optimally solve medium sized problems within a reasonable computation time.

Graves and Lee (1999) studied the problem of scheduling a set of jobs on a single machine where the machine must be under maintenance during certain intervals implying a non-availability of the machine during these periods. When a job is not completely handled before the machine is turned off for maintenance, a setup time is necessary when the production is taken back. Two objectives were considered: the minimization of the total weighted jobs completion time and the minimization of the maximum delay. The problem is solved by the dynamic programming approach.

Aghezzaf et al. (2007) were interested in the batch production problem. The production system is subject to random failures and at each maintenance intervention the production system is not available which decreases its production capacity. The purpose of their study was to determine a plan in which the cost of production and maintenance is minimal. The problem is solved by a linear model.

The remainder of this paper is organized as follows. In the section "Problem description and notations", a description of the problem and necessary notations are presented. Then, three linear formulations are proposed in "Mixed integer linear models". Some valid additional constraints to improve the linear relaxation of the developed models are also described. In the section "Lower bounds procedure", we develop an assignment based lower bound and a Lagrangian based lower bound. In this same section, we also adapt the lower bound of Li (1997) to our problem. Section "The branch and bound algorithm" briefly describes the implementation of the branch and bound algorithm. The local search technique and the genetic algorithm are respectively presented in the sections "Local search procedure" and "Genetic algorithm (GA)". Computational results are discussed in the section "Computational results". Finally, we conclude by summarizing the main proposals presented in this article and the obtained results.

## Problem description and notations

The considered problem in this paper is to schedule $M$ preventive maintenance tasks that have to take place on $M$ machines charged to process production tasks (exactly one preventive maintenance task per machine). We aim to determine a sequence composed of the $M$ preventive maintenance tasks such as each preventive maintenance task $i$ has a processing time $p_i$, an optimistic deadline $d_{i1}$, a pessimistic deadline $d_{i2}$ greater or equal to $d_{i1}$, a tardy weight $w_i$, an early weight $h_i$ and a minimal preventive maintenance cost $C_{i0}$. The execution of the preventive maintenance tasks on the machines should be continuous (i.e., non-premptive) during the schedule horizon (i.e., the interval $\left[0, \sum_{i=1}^{M} p_i\right]$). This constraint holds when the cost of having the maintenance resource is very expensive. If a preventive maintenance task $i$ starts before its optimistic deadline $d_{i1}$ the preventive maintenance cost of the considered task exceeds $C_{i0}$ and will be equal to $h_i\,(d_{i1} - t_i) + C_{i0}$ where $t_i$ is the starting time of the task $i$. Similarly, when a preventive maintenance task $i$ starts after its pessimistic deadline $d_{i2}$, the preventive maintenance cost is greater than $C_{i0}$. This cost is equal to $w_i\,(t_i - d_{i2}) + C_{i0}$ Finally, if the preventive maintenance task $i$ starts at an instant from the $[d_{i1}\, d_{i2}]$ interval, the preventive maintenance cost is equal to $C_{i0}$ (see Fig. 1 for illustration). Our objective in this paper is to provide a schedule composed of the $M$ preventive maintenance tasks with a minimal preventive maintenance cost. We remind that the preemption of the tasks is not allowed and no idle time is permitted, providing so a continuously execution of the tasks during the maintenance schedule horizon. We remind also that this problem can be treated as a problem of scheduling a given $M$ jobs (or preventive maintenance tasks) on a single machine (the preventive maintenance resource) with the objective of minimizing the sum of total weighted earliness and tardiness of jobs. The described problem can be noted as $1\,|d_{1i}\,d_{2i}|\,\sum w_i T_i + h_i E_i$ according to the standard scheduling notation, where $T_i$ and $E_i$ are respectively the tardiness and the earliness of task $i$.

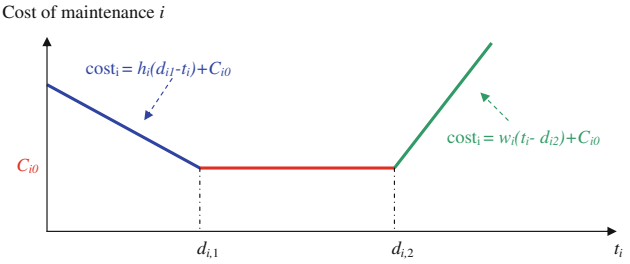


**Fig. 1** Cost of preventive maintenance $i$

## Mixed integer linear models

In this section we first present three linear formulations for the problem described above. Then, we propose some valid constraints that can be added to the developed linear models. The impact of adding these constraints is discussed in the computational experiments section.

First formulation (MILP1)

In this first formulation, we use the binary incidence variables on precedence between jobs (maintenance tasks) as decision variables. We note that the precedence between the jobs is not necessary immediate. Hence, our decision variables are defined as follows:

$$x_{ij} = \begin{cases} 1 & \text{if the job } i \text{ precedes the job } j. \\ 0 & \text{otherwise} \end{cases}$$

Let us also define $d_i^+$ the positive deviation of the job $i$ from its pessimistic deadline $d_{i,2}$ or the tardiness that can be happen. Similarly, we define $d_i^-$ as the negative deviation of the job $i$ from its optimistic dead line $d_{i1}$ or the earliness that may occur. Our first proposed linear formulation for the problem is the following.

$$Min \sum_{i=1}^{M} w_i d_i^+ + h_i d_i^- + C_{i0}$$

$$d_i^- \geq d_{i1} - t_i \quad \forall i = 1 \ldots M \tag{1}$$
$$d_i^+ \geq t_i - d_{i2} \quad \forall i = 1 \ldots M \tag{2}$$
$$x_{ij} + x_{ji} = 1 \quad \forall i \neq j; i = 1 \ldots M; j = 1 \ldots M \tag{3}$$
$$x_{ii} = 0 \quad \forall i = 1 \ldots M \tag{4}$$
$$t_j \geq t_i + p_i - P(1 - x_{ij}) \quad \forall i \neq j; i = 1 \ldots M; j = 1 \ldots M \tag{5}$$
$$t_i \geq 0 \quad \forall i = 1 \ldots M \tag{6}$$
$$t_i \leq P - p_i \quad \forall i = 1 \ldots M \tag{7}$$
$$d_i^+ \geq 0, d_i^- \geq 0 \quad \forall i = 1 \ldots M, \tag{8}$$
$$t_i \geq 0 \quad \forall i = 1 \ldots M \tag{9}$$
$$x_{ij} \in \{0, 1\} \tag{10}$$

where $P = \sum_{i=1}^{M} p_i$.

Inequalities 1 and 2 decide for each job to start early, tardy or on time. The Eqs. 3 and 4 decide for each job $i$ to be before or after another job $j\,(i \neq j)$.

The inequalities 5 compute for every job $i \neq j$ the optimal starting time when job $i$ precedes job $j$ in the optimal sequence. Otherwise, the constraint is redundant.

Equations 6 and 7 indicate the limits of the starting time of a given job. The remaining constraints are integer and non negative constraints.

We note that the MIP solver of ILOG CPLEX 10.1 is able to solve instances with $M = 17$ in less than 1 h. Also due to the use of the big parameter $P$, the linear relaxation obtained by dropping the integrity constraints provides a weak lower bound.

Second formulation (MILP2)

The second proposed formulation is a time-indexed formulation. The main idea on which this formulation is based consists in decomposing the schedule horizon into time slots where each time slot starts at time $t$ and ends at time $t+1$ ($t \in [0, P-1]$). According to Valente and Shaller (2010), the major advantage of using this time-index formulation is that the linear relaxation obtained by dropping the constraint of variables integrities provides in general 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 time consuming especially when the planning horizon is big. For our problem, let $x_{it}$ be a binary variable equal to 1 if job $i$ starts at time $t$ and to 0, otherwise. By conserving the same variables $d_i^+$ as the positive deviation of the job $i$ from its pessimistic date $d_{i,2}$, $d_i^-$ as the negative deviation of the job $i$ from its optimistic date $d_{i,1}$ and $t_i$ the optimal starting time of the task $i$ in the optimal sequence, the time-indexed linear formulation can be described as follows.

$$Min \sum_{i=1}^{M} w_i d_i^+ + h_i d_i^- + C_{i0}$$

$$\sum_{t=0}^{P-pi} x_{it} = 1 \quad \forall i = 1 \ldots N \tag{1}$$
$$\sum_{i=1}^{M} \sum_{s=\max\{t-p_i,0\}}^{t} x_{is} \leq 1 \quad \forall t = 0 \ldots P-1 \tag{2}$$
$$\sum_{t=0}^{P-pi} t x_{it} + d_i^- \geq d_{i,1} \quad \forall i = 1 \ldots M \tag{3}$$
$$\sum_{t=0}^{P-pi} t x_{it} - d_i^+ \leq d_{i,2} \quad \forall i = 1 \ldots M \tag{4}$$
$$x_{it} \in \{0, 1\} \tag{5}$$

In this model, the first equations ensure that each job is processed once. The second inequalities are resources constraints indicating that at most one job can be handled at a time slot. The third and the fourth inequalities decide for each task to start early, tardy or on time. Finally, the fifth constraints are integrity constraints.

For this formulation, the MIP solver of ILOG Cplex 10.1 is able to solve all our small instances with a size less or equal to 15 jobs in a time period less than 1 h.

Third formulation (MILP3)

This third formulation is based on the binary assignment variables of the jobs to the positions in the optimal sequence. Indeed, the optimal sequence is composed of $M$ jobs. Each job $i\,(i = 1 \ldots M)$ is assigned to a position $j\,(j = 1 \ldots M)$ in the sequence. Hence, our decision variables in this formulation can be defined as follows:

$$x_{ij} = \begin{cases} 1 & \text{if the job } i \text{ is assigned to position } j. \\ 0 & \text{otherwise.} \end{cases}$$

Let us also consider $t_i$ the starting time of job $i$ in the optimal sequence. We use the same variables $d_i^+$ as the positive deviation of job $i$ from its pessimistic date $d_{i,2}$ and $d_i^-$ as

the negative deviation of job $i$ from its optimistic date $d_{i,1}$ and by considering $t_i$ as the optimal starting time of job $i$ in the optimal sequence. The third linear formulation of the problem is described as follows.

$$Min \sum_{i=1}^{M} w_i d_i^+ + h_i d_i^- + C_{i0}$$

*S/C*

$$t_i - d_i^+ + d_i^- \geq d_{i1} \quad \forall i = 1 \ldots M \tag{1}$$

$$t_i - d_i^+ + d_i^- \leq d_{i2} \quad \forall i = 1 \ldots M \tag{2}$$

$$\sum_{i=1}^{M} x_{ij} = 1 \quad \forall i = 1 \ldots M \tag{3}$$

$$\sum_{j=1}^{M} x_{ij} = 1 \quad \forall i = 1 \ldots M \tag{4}$$

$$t_i - P(1 - x_{ij}) \leq \sum_{k=1}^{j} \sum_{t=1}^{M} p_t x_{tk} - p_i \quad \forall i, j = 1 \ldots M \tag{5}$$

$$t_i + P(1 - x_{ij}) \geq \sum_{k=1}^{j} \sum_{t=1}^{M} p_t x_{tk} - p_i \quad \forall i, j = 1 \ldots M \tag{6}$$

$$x_{it} \in \{0, 1\} \tag{7}$$

Equations 1 and 2 decide for each job to start early, tardy or on time. Equalities 3 and 4 are assignment constraints. They indicate that each job has to take just one position and each position must contain just one job. Inequalities 5 and 6 compute the optimal starting time of each job $i$ if it is assigned to position $j$ in the optimal sequence. Otherwise, the constraints are redundant. The remaining constraints are integrity constraints.

For this formulation, the MIP solver of ILOG Cplex 10.1 is able to solve all our small instances with a size $M \leq 13$ in a time period less than 1 h. Because of using the big parameter $P$, the linear relaxation of this formulation provides a weak lower bound.

Improvement of the linear relaxations

In this subsection, we propose some valid constraints that can be added to the developed models. We remind that the main objective of adding the following constraints is to improve the linear relaxation of the models.

*First type constraints (C1)*

This first type of valid inequalities is based on the WSPT rule developed by Smith (1956) and the WLPT rule.

**Definition 1** Let $\alpha = (\alpha_1, \ldots, \alpha_M)$ and $\beta = (\beta_1, \ldots, \beta_M)$ be tow vectors of positive numbers and $p = (p_1, \ldots, p_M)$ be the vector of processing times. Let $WF_1(p, \alpha)$ denotes the minimal weighted flow-time obtained by applying the $WSPT$ rule (proposed by Smith (1956)) to the corresponding problem $1 \,||\, \sum \alpha_i C_i$ and $WF_2(p, \beta)$ denotes the maximal weighted flow-time obtained by applying the $WLPT$ rule.

*Property 1* Kacem (2007). Let $\alpha = (\alpha_1, \ldots, \alpha_M)$ be a vector of positive numbers and $t_j$ $(1 \leq j \leq M)$ be the set of starting times of jobs in a feasible schedule. Therefore, the following inequality holds:

$$\sum_{i=1}^{M} \alpha_i t_i \geq WF_1(p, \alpha) - \sum_{i=1}^{M} \alpha_i p_i$$

$$\sum_{i=1}^{M} \alpha_i t_i \leq \sum_{i=1}^{M} \alpha_i (P - p_i) - WF_1(p, \alpha) - \sum_{i=1}^{M} \alpha_i p_i^{WSPT}$$

*Property 2* Let $\beta = (\beta_1, \ldots, \beta_M)$ be a vector of positive numbers and $(t_i)$ $(1 \leq i \leq M)$ be the set of starting times of jobs in a feasible schedule. Therefore, the following inequality holds:

$$\sum_{i=1}^{M} \beta_i t_i \leq WF_2(p, \beta) - \sum_{i=1}^{M} \beta_i p_i$$

$$\sum_{i=1}^{M} \beta_i t_i \geq \sum_{i=1}^{M} \beta_i (P - p_i) - WF_2(p, \beta) - \sum_{i=1}^{M} \beta_i p_i^{WLPT}$$

*Second type constraint (C2)*

*Property 3* Let $p = (p_1, \ldots, p_M)$ be the vector of the job's processing times. Based on the tow properties presented above, the following equality holds:

$$\sum_{i=1}^{M} p_i t_i = \sum_{i=1}^{M} p_i C_i^H - \sum_{i=1}^{M} p_i^2$$

where $C_i^H$ is the completion time of job $i$ in an arbitrary solution (heuristic solution).

*Remark* This equality has a great impact on the improvement of the linear relaxation of the first and last formulations.

*Third type constraints (C3)*

These added constraints are reserved to the last formulation. The main idea of this third type of constraints is to limit the possible value of the decision variable $t_i$ in each position. In other words, we try to determine for each variable $t_i$ in each position a lower bound and an upper bound in order to not let this variable take impossible values which leads to the weakness of the linear relaxation.

**Definition 2** Let $\underline{t}_{ij}$ be a lower bound on the starting time of job $i$ assigned to position $j$ in the optimal sequence calculated according the following formula:

$$\underline{t}_{ij} = \begin{cases} \underline{t}^1_{ij} = \sum_{k=1}^{j} p_{[k]} - p_i & \text{if } i \in \{[1],[2],\ldots,[j]\} \\ \underline{t}^2_{ij} = \sum_{k=1}^{j-1} p_{[k]} & \text{if } i \notin \{[1],[2],\ldots,[j]\} \end{cases}$$

$[k]$ is the $k^{\text{th}}$ job according to the *SPT* order (Shortest Processing Time).

Therefore, the following inequality holds:

$$t_i \geq \sum_{j=1}^{M} \underline{t}_{ij} x_{ij} \quad \forall i = 1 \ldots M$$

**Definition 3** Let us consider $\bar{t}_{ij}$ an upper bound on the starting time of job $i$ assigned to position $j$ in the optimal sequence calculated according the following formula:

$$\bar{t}_{ij} = \begin{cases} \bar{t}^1_{ij} = \sum_{k=1}^{j} p_{[M-k+1]} - p_i & \text{if } i \in \{[M],[M-1], \\ & \ldots,[M-j+1]\} \\ \bar{t}^2_{ij} = \sum_{k=1}^{j-1} p_{[M-k+1]} & \text{if } i \notin \{[M],[M-1], \\ & \ldots,[M-j+1]\} \end{cases}$$

Therefore, the following inequality holds:

$$t_i \leq \sum_{j=1}^{M} \bar{t}_{ij} x_{ij} \quad \forall i = 1 \ldots M$$

*Fourth constraints (C4)*

The following equality is particular to the first proposed model:

$$\sum_{i=1}^{M} p_i x_{ij} = t_j \qquad \forall j = 1 \ldots M$$

Indeed, if we sum all the processing times of the jobs that precede job $j$ then, we can easily obtain the starting time of job $j$.

## Lower bounds procedure

In this section we propose three lower bounds procedures. The first one is based on the Lagrangian relaxation of the constraints (1) and (2) of the first developed model to which we add only the last added constraints (C4) presented above. The second one is an assignment based lower bound obtained by the sum of $M$ computed costs corresponding to the $M$ jobs. The third one is an adaptation of the lower bound proposed by Li (1997) for the problem of earliness–tardiness minimization with one due date for every job.

### Lagrangian based lower bound

We consider in this subsection a lower bound based on the Lagrangian relaxation of the first and the second constraints of the first developed model (MILP1) to which the forth type of valid constraints (cuts) is added. Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_i, \ldots, \lambda_M)$ be the vector of nonnegative multipliers corresponding to the $M$ first relaxed constraints and let $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_i, \ldots, \gamma_M)$ be the vector of nonnegative multipliers corresponding to the $M$ second relaxed constraints. Therefore, the objective function of the relaxed model can be written as follows:

$$\begin{aligned} \text{(LR) } Min\, Z(\lambda,\gamma) = \sum_{i=1}^{N} & (w_i - \lambda_i) d_i^+ + (h_i - \gamma_i) d_i^- + C_{i0} \\ & + \lambda_i d_{1i} - \gamma_i d_{2i} + (\gamma_i - \lambda_i) t_i \end{aligned}$$

For any sequence, the minimization of the first and the second terms in $Z(\lambda,\gamma)$ are achieved by setting $d_i^+$ to 0 if $w_i \geq \lambda_i$ and $d_i^-$ to 0 if $h_i \geq \gamma_i$ for all $i = 1, \ldots, M$, or by setting $d_i^+ = -\infty$ if $w_i < \lambda_i$ and $d_i^- = -\infty$ for all $i$ if $h_i < \gamma_i$. Since the latter case tends to lead to $Z(\lambda,\gamma) = -\infty$, a situation where the lower bound is not useful, we consider only the first case in the following discussion. The third, the forth and the fifth terms are constant. For the last remaining term, we impose the following constraint $\lambda_i < \gamma_i$ to obtain always a non negative lower bound.

To determine the value of the Lagrangian multipliers and obtain a lower bound value, we use the sub-gradient method. The sub-gradient method is a simple iterative algorithm used to minimize a non differentiable convex function. In this algorithm we start by initializing each $\lambda_i$ multiplier by $w_i$ and each $\gamma_i$ multiplier by $h_i$. Then, we solve the LR problem and we obtain the $t_i$ values. The sub-gradient related to the $\lambda_i$ multiplier is then computed by subtracting from $d_{1i}$ the $t_i$ value and the sub-gradient related to the $\gamma_i$ multiplier is determined by subtracting from $t_i$ the value of $d_{2i}$. The update of the Lagrangian multipliers from the $k^{\text{th}}$ iteration to the $(k+1)^{\text{th}}$ iteration is made by adding to the Lagrangian multipliers values computed in the $k^{\text{th}}$ iteration the value of the sub-gradient multiplied by the coefficient $\theta = \alpha(bs - \max(bi, 0))/Norm$ where $\alpha$ is a parameter that takes 2 at the beginning and it is divided by 2 at each iteration; $bs$ is the upper bound of the B&B algorithm (it will be described later); $bi$ is the lower bound obtained at the $k^{\text{th}}$ iteration and $Norm$ is equal to the sum of the squared sub-gradients. When the value of $\alpha$ is inferior to 0.0001, then it takes 2 as new value. After computing the new values of the Lagrangian multipliers we verify the imposed conditions. If the value of $\lambda_i$ exceeds $w_i$ it takes $w_i$ as new multiplier. The same rule is applied to $\gamma_i$ parameter: if it exceeds $h_i$ value then it takes as new value $h_i$. When $\lambda_i$ or $\gamma_i$ takes a negative value after updating, they are replaced by 0. We note that

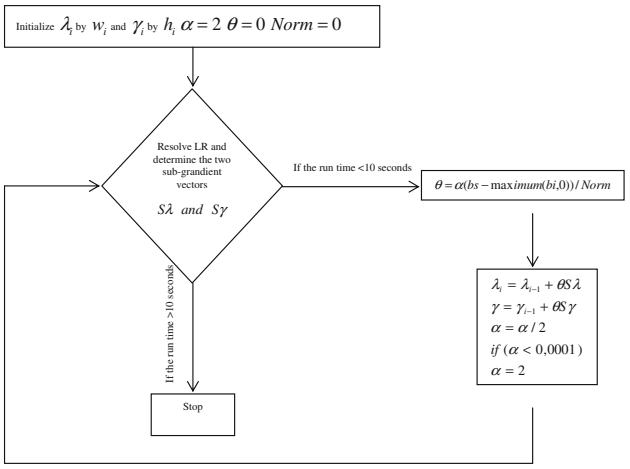


**Fig. 2** Computation process of the Lagrangian lower bound

the time period allowed for computing the lower bound is equal to 10 s. The following diagram in Fig. 2 illustrates the computation process:

Assignment based lower bound

We consider in this subsection a new lower bound obtained by the sum of $M$ costs assigned to the $M$ jobs. We first start by determining a minimal cost for every job in every position. The whole costs are gathered in a cost's matrix noted $M^c$ in which every element $a_{ij}$ represents the minimal cost of assigning job $i$ to position $j$ ($i = 1, \ldots, M$; $j = 1, \ldots, M$). After computing the cost's matrix, we determine the lower bound's value by summing $M$ costs, assigned to the $M$ jobs. The objective of the assignment problem is to minimize the total cost assignment.

Let $\hat{C}_{ij}(S)$ be the tardiness cost of job $i$ assigned to position $j$:

$$\hat{C}_{ij}(S) = \begin{cases} w_i(\underline{t}_{ij} - d_{i2}) & \text{if } \underline{t}_{ij} - d_{i2} > 0 \\ 0 & \text{otherwise} \end{cases}$$

We note $\hat{C}_{ij}(L)$the earliness cost of job $i$ assigned to position $j$:

$$\hat{C}_{ij}(L) = \begin{cases} h_i(d_{i1} - \bar{t}_{ij}) & \text{if } d_{i1} - \bar{t}_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

If we compute for every job in each position the cost $\hat{C}_{ij} = \hat{C}_{ij}(L) + \hat{C}_{ij}(S)$ then, the value obtained by solving the assignment problem defined by the assignment of just one job in each position with minimizing the total cost assignment, determines a lower bound for the total earliness and tardiness problem.

In our implementation, the lower bound is computed by the Hungarian algorithm with a time complexity of $O(M^3)$ (Table 1).

**Table 1** Single machine earliness/tardiness scheduling problem with 5 jobs

| Job $i$ | $p_i$ | $d_{i1}$ | $d_{i2}$ | $w_i$ | $h_i$ |
|---|---|---|---|---|---|
| 1 | 62 | 75 | 109 | 3 | 9 |
| 2 | 89 | 51 | 69 | 3 | 7 |
| 3 | 25 | 62 | 80 | 3 | 5 |
| 4 | 69 | 49 | 61 | 2 | 3 |
| 5 | 23 | 63 | 111 | 10 | 7 |

*Example* The optimal sequence is: 4 3 5 1 2 with a total cost of 501. For the given example, the cost's matrix is the following:

$$M^c = \begin{pmatrix} 675 & 0 & 0 & \boxed{24} & 291 \\ 357 & 0 & 0 & 123 & \boxed{330} \\ 310 & \boxed{0} & 15 & 222 & 489 \\ \boxed{147} & 0 & 0 & 98 & 276 \\ 441 & 0 & \boxed{0} & 450 & 1340 \end{pmatrix}$$

The minimal cost obtained for the assignment problem is equal to 501 with job 4 assigned to position 1, job 3 to position 2, job 5 to position 3, job 1 to position 4 and finally job 2 to position 5. In this example, we observe that the lower bound is equal to the optimal solution.

Adaptation of Li's [12] lower bound

The main idea of this lower bound procedure consists in formulating the problem as a simple linear model and decomposing it into two sub-problems with simpler structure. By solving the Lagrange duals of the obtained sub-problems using the multiplier adjustment method developed by Potts and Van Wassenhove (1984), the lower bound of the main problem is derived by summing the two obtained lower bounds of the sub-problems. Here, the sub-problems are obtained by a simple decomposition of the objective function into two parts: the earliness part and the tardiness part. For each part, the technique proposed by Potts and van Wassenhove can be applied to derive an efficient lower bound. The two Lagrange duals sub-problems were solved by using the efficient adjustment multiplier approach in $O(M\log(M))$ time.

## The branch and bound algorithm

In this section, we briefly define the three main procedures on which the branch and bound algorithm is based: the initialization, the branching and the bounding.

Initialization

At this step, an initial complete solution is used until a better new solution is obtained from the search tree. Our proposed initial solution is based on efficient meta-heuristic solutions that will be described later in the next sections.

Branching

Branching is a procedure used to develop the search tree. In this paper, we adopt the depth first strategy with a backward and forward sequencing branching rule. In the backward sequencing branching rule, a node at a level $k$ of the search tree corresponds to a sequence with $k$ jobs fixed in the $k$ last positions. However, in the forward sequencing branching rule the $k$ fixed jobs are sequenced in the first $k$ positions.

Bounding

Bounding means determining a lower bound for a partial sequence $S_p$ in a node of the tree search. Two bounding procedures are tested. In the first one, we used only two lower bounds: The adapted Li lower bound (LB_Li) and the assignment lower bound (LB_Aff). In the second one, in addition of these lower bounds, the linear-relaxation lower bound (denoted later as $LB_{14}$) of the first model to which we add the fourth type of valid constraints (cuts) is used (this choice has been motivated by the results of the numerical experiments we will show later in the section "Computational results"). To more improve the performance of the tree search, we used a dominance property that consists in comparing tow partial schedules: The first partial schedule is composed of the fixed jobs in the node and the second is obtained by exchanging the tow last jobs in the first sequence. In each node of the tree search we first verify the dominance property: If the evaluation of the first partial schedule is grater than the evaluation of the second partial schedule we stop the branching procedure on the node. Otherwise, we move to compute the lower bounds.

In the first bounding procedure, we start by computing LB_Li on the problem composed of the remaining non fixed jobs. If the evaluation of the partial fixed sequence to which we add the value of LB_Li is greater than the upper bound value we remove the current node. Otherwise, we move to compute the next lower bound LB_Aff. If the evaluation of the partial fixed sequence to which we add the value of the second computed lower bound is greater than the current upper bound value we stop the separation. Otherwise, we continue to explore new branches from the current node.

In the second bounding procedure, instead of exploring new branches after testing LB_Aff, we compute $LB_{14}$ If the evaluation of the partial fixed sequence to which we add the $LB_{14}$ value is greater than the current upper bound value then, we stop the separation. Otherwise, we explore the new branches starting from the current node.

The difference between the two bounding procedures will be shown in the computational results section.

## Local search procedure

A local search procedure is a non exact method that allows us to obtain a solution that may be optimal or near to the optimal. The main elements that constitute a local search algorithm are respectively: an initial solution, a movement and a stop condition. The local search algorithms may differ in the choice of the starting solution, the movements by which the neighbourhood is formed or by the stop condition.

In our local search algorithm, we have chosen as initial solution the one proposed by Ow and Morton (1989) (described in the "Appendix"). This solution is a good heuristic for the problem of minimizing the weighted earliness–tardiness jobs on a single machine with just one due date for every job. Because we have two due dates for our problem in stead of one, then we ignore the first due date and only the second due date $d_{i2}$ has been considered in the construction of the initial solution. Many movements to form the neighbourhood are used. The first one consists in the permutation of two adjacent jobs. The second movement consists in the permutation of two jobs having just one job between them. The $k^{th}$ movement consists in exchanging two jobs having $k-1$ jobs between them. We have used $M$ movements to create the neighbourhood. Once ending the generation of the neighbourhood, the solution with the best evaluation is then taken and compared to the upper bound value. If the value of the upper bound is greater than the evaluation of the best neighbourhood solution we update the upper bound value and we repeat the procedure to the new solution. Otherwise, in case we have 5 successive non improvements of the upper bound value we move to generate another random solution. The procedure is repeated until having generating 10 random solutions. The stop condition of our local search algorithm is having 15 successive times non improvements of the final solution.

The proposed local search algorithm works as follows:

1- Generate the initial solution by applying Ow and Morton (1989) rule to the tasks and assign its evaluation to the local search solution value.
2- Repeat the following until having 15 successive times of non improvement of the final solution value:
   (a) Create the neighbourhood by the $M$ movements.
   (b) Evaluate each neighbourhood solution and select the one with the best evaluation.

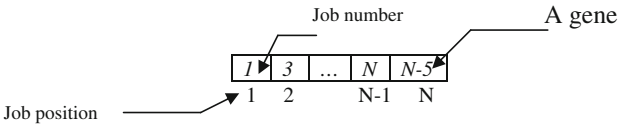


**Fig. 3** Chromosome representation

(c) Update the final solution value when the evaluation of the best found solution is better than the current final solution.

## Genetic algorithm (GA)

GA is an efficient search algorithm used to find excellent solution for hard problems in reasonable time. Its main idea is inspired from the natural reproduction of human beings: the crossing of two individuals give new children. The main elements that formed a genetic algorithm are respectively the initial population of size $N_p$, the cross-over operator, the mutation operator and the replacement strategy.

### The initial population

The initial population is a set of $N_p$ initial solutions called chromosomes or individuals. The choice of the number $N_p$ differs from a genetic algorithm to another. In general, a big number $N_p$ has as consequence the increase the time resolution and a small $N_p$ may influence the quality of solution (due to a partial exploration of the search space). The chromosomes can be randomly generated. In our proposed genetic algorithm, we create an initial population with size $N_p$ of 100 individuals. Its first five chromosomes are obtained respectively by applying *SPT, LPT, WSPT, WLPT* and Ow and Morton (1989) heuristics. The remaining chromosomes are randomly generated. We note that a chromosome is encoded according to the representation of Fig. 3.

### The crossover operator

In the cross-over step, two parents are selected to produce at least one child. For the proposed genetic algorithm, only 80% of the best individuals of the population are taken to undergo the crossover operation and produce two children. The selection of the parents is organized by the following rule: The first and the second parents are first selected. Then, the third and the fourth follow. Then, the other are selected until the parents number seventy nine and eighty. In our implementation, we have used two crossover operators: "one opts" and "tow opts". In the "one opts" operator, we randomly generate an integer $k$ from the uniform $[1, M]$. Then the first child $C_1$ receives the subsequence $[1, \ldots, k]$ of its genes from *the first parent and* the second child $C_2$ receives the subsequence $[1, \ldots, k]$ of its genes from *the second parent*. Then, we fill the remaining empty genes of *the child* $C_1$ according to their order of appearance in the second parent: if a gene is already in $C_2$, reject it; else position it in the first empty gene in $C_1$. Finally we fill the remaining empty genes of *the child* $C_2$ according to their order of appearance in the first parent: if a gene is already in $C_1$ reject it; else position it in the first empty gene in $C_2$

In the two "opts operator", we randomly generate two integers $k_1$ and $k_2$, greater than $k_1$ from the uniform $[1, M]$. Then, the first child $C_1$ receives the two sub-sequences $[1, \ldots, k_1]$ and $[k_2, \ldots, M]$ of its genes from *the first parent* and the second child $C_2$ receives the two sub-sequences $[1, \ldots, k_1]$ and $[k_2, \ldots, M]$ of its genes from *the second parent*. Then, we fill the remaining empty genes of *the child* $C_1$ according to their order of appearance in the second parent and the remaining empty genes of $thechild\, C_2$ according to their order of appearance in the first parent. We note that the choice of the crossover operator is randomly chosen.

### The mutation operator

In the mutation operation, an individual is randomly chosen from the population to undergo a small modification for improving its quality. In our problem, we apply the mutation on ten individuals randomly chosen from the population. The modification consists in the permutation of two genes that are randomly selected.

### The replacement strategy

Keeping the population size constant from a generation to another is very important for different reasons such as consuming less time to obtain a solution and also less memory. Hence, the idea consists in replacing the individuals (solutions) in the population with worst evaluations by the new children with better evaluations. In our implementation, we have imposed a condition to the new children with best evaluation to enter the population. The condition consists in having at least 3 genes that are different compared to those of the best ten solutions of the population of the last iteration. In other words, if a child with good evaluation does not differ from all the best first ten solutions in at least three genes then, it cannot enter the population and undergo crossover or mutation operations. This condition has a good impact on the diversification of the population because it eliminates the appearance of the same solutions more than once in the population.

To more improve the GA solution, we create for the first and the second best found solutions the neighbourhood described before. The final solution is the one with the best evaluation among the first and the second solutions and their neighbourhood solutions.

**Table 2** The impact of the added constraints (C1) (C2) and (C4) to the first model according to the $T$ factor

| | LB1 | LB11 | LB14 | LB12 | LB124 | LB1124 |
|---|---|---|---|---|---|---|
| $N=10$ | | | | | | |
| $T=0.2$ | 0 | 1666.062 | 1771.1755 | 447.637 | 1771.1755 | 1771.1755 |
| $T=0.3$ | 0 | 1069.125 | 1098.572 | 131.52 | 1098.572 | 1098.572 |
| $T=0.4$ | 0 | 239.94 | 242.748 | 4.196 | 242.748 | 242.748 |
| $T=0.5$ | 0.75 | 672.812 | 683.1785 | 37.6705 | 683.1785 | 683.1785 |
| $T=0.6$ | 66.95 | 2190.7075 | 2289.5675 | 482.858 | 2289.5675 | 2289.5675 |
| $T=0.7$ | 337.3 | 4372.2895 | 4666.3795 | 2093.7025 | 4666.3795 | 4666.3795 |
| $T=0.8$ | 1083.4 | 6357.949 | 7152.732 | 4335.2615 | 7152.732 | 7152.732 |
| $N=20$ | | | | | | |
| $T=0.2$ | 0 | 9357.962 | 10409.936 | 2227.337 | 10409.936 | 10409.9355 |
| $T=0.3$ | 0 | 6103.5975 | 6510.8925 | 372.3205 | 6510.8925 | 6510.8925 |
| $T=0.4$ | 0 | 1531.302 | 1568.3635 | 0.8545 | 1568.3635 | 1568.3635 |
| $T=0.5$ | 0 | 1069.2655 | 1082.503 | 0 | 1082.503 | 1082.503 |
| $T=0.6$ | 45.45 | 5831.059 | 6292.9575 | 338.0455 | 6292.9575 | 6292.9575 |
| $T=0.7$ | 605.35 | 12184.801 | 13668.734 | 3093.0805 | 13668.734 | 13668.734 |
| $T=0.8$ | 2189 | 22247.607 | 25403.0815 | 10917.4995 | 25403.0815 | 25403.0815 |
| $N=30$ | | | | | | |
| $T=0.2$ | 0 | 22319.236 | 25496.883 | 4418.0355 | 25496.883 | 25496.883 |
| $T=0.3$ | 0 | 13888.825 | 15138.868 | 348.39 | 15138.868 | 15138.8675 |
| $T=0.4$ | 0 | 4374.4955 | 4555.272 | 0 | 4555.272 | 4555.272 |
| $T=0.5$ | 0 | 1968.022 | 2018.338 | 0 | 2018.338 | 2018.338 |
| $T=0.6$ | 43.4 | 11289.938 | 12207.749 | 181.776 | 12207.749 | 12207.749 |
| $T=0.7$ | 971.2 | 25935.945 | 29490.268 | 4666.7945 | 29490.268 | 29490.2675 |
| $T=0.8$ | 4339.5 | 45464.2115 | 53250.2775 | 19142.344 | 53250.2775 | 53250.2775 |

The main steps of the proposed genetic algorithm can be summarized as follows:

1. Generate the initial population composed of $N_p$ individuals.
2. Evaluate each individual of the population.
3. Determine the sub-population undergoing the cross-over (80% of the best chromosomes).
4. Repeat $n_g$ times.
   (a) Select two by two parents from the current sub-population.
   (b) Apply crossover to the two parents to produce two children.
   (c) By the replacement strategy, decide if introduce the new children.
   (d) Randomly select ten individuals from the whole population.
   (e) Apply mutation to the selected individuals.
   (f) By using the replacement strategy, decide if introduce the new mutated individuals.
5. Create for the first and the second best found solutions a neighbourhood.
6. Find the best solution

## Computational results

To generate an instance, three parameters are needed: the number of jobs $M$, the range factor $R$ and the tardiness factor $T$. For each job $i$ the processing time is randomly determined from the discrete uniform distribution $[1 \ldots 99]$. The optimistic deadlines $d_{i1}$ are then generated from the discrete uniform distribution $[d_{min} \cdots d_{min} + R.P]$ where $d_{min} = max\{0;\ P\ (T - R/2)\}$. The pessimistic deadlines are generated from the discrete uniform distribution $[d_{i1}; \cdots ; d_{i1} + P/M]$. Finally, the earliness and the tardiness penalties are drawn from $[1 \ldots 10]$.

To show the impact of the adding constraints presented in the section "Mixed integer linear models", we have tested the relaxed developed models on problems with size of 10, 20 and 30 jobs. 20 instances have been generated for each combination of the following parameters: $T \in \{0.2, 0.4, 0.6, 0.8\}$, $R \in \{0.2, 0.4, 0.6, 0.8\}$; and $M \in \{10, 20, 30\}$. We note that the processing times are determined from the discrete uniform distribution $[1 \ldots 30]$ for these tests.

The branch-and-bound algorithm is tested on problems with 15, 20, 25, 30 and 40 jobs. 5 instances have been generated for each combination of $T$ and $R$. To prevent

**Table 3** The impact of the added constraints (C1) (C2) and (C4) to the first model according to the $R$ factor

| | LB1 | LB11 | LB12 | LB14 | LB124 | LB1124 |
|---|---|---|---|---|---|---|
| $N=10$ | | | | | | |
| $R=0.2$ | 4.4 | 2780.6205 | 3000.889 | 1272.7075 | 3000.889 | 3000.889 |
| $R=0.3$ | 43.7 | 2333.068 | 2520.8325 | 915.2965 | 2520.8325 | 2520.8325 |
| $R=0.4$ | 85.65 | 2597.925 | 2771.614 | 1064.8805 | 2771.614 | 2771.614 |
| $R=0.5$ | 149.05 | 2127.0155 | 2304.037 | 892.2265 | 2304.037 | 2304.037 |
| $R=0.6$ | 303.65 | 2336.5135 | 2538.4145 | 1121.065 | 2538.4145 | 2538.4145 |
| $R=0.7$ | 356.05 | 2062.473 | 2204.2295 | 906.764 | 2204.2295 | 2204.2295 |
| $R=0.8$ | 520.65 | 2189.312 | 2382.318 | 1219.393 | 2382.318 | 2382.318 |
| $N=20$ | | | | | | |
| $R=0.2$ | 0 | 8858.0345 | 9810.1465 | 2421.909 | 9810.1465 | 9810.1465 |
| $R=0.3$ | 13.65 | 8639.726 | 9645.627 | 2510.4165 | 9645.627 | 9645.627 |
| $R=0.4$ | 91 | 8845.509 | 9862.67 | 2254.7025 | 9862.67 | 9862.67 |
| $R=0.5$ | 239.05 | 8511.1415 | 9528.535 | 2146.0695 | 9528.535 | 9528.535 |
| $R=0.6$ | 528.55 | 8565.548 | 9585.317 | 2511.085 | 9585.317 | 9585.317 |
| $R=0.7$ | 834.15 | 8182.78 | 9020.643 | 2546.111 | 9020.643 | 9020.643 |
| $R=0.8$ | 1082.4 | 6872.607 | 7677.265 | 2497.2305 | 7677.265 | 7677.265 |
| $N=30$ | | | | | | |
| $R=0.2$ | 0 | 18761.619 | 21396.741 | 3786.0245 | 21396.741 | 21396.741 |
| $R=0.3$ | 0.6 | 19264.175 | 21896.635 | 4050.201 | 21896.635 | 21896.635 |
| $R=0.4$ | 99.8 | 19409.383 | 22190.691 | 4264.609 | 22190.691 | 22190.691 |
| $R=0.5$ | 331.5 | 17912.041 | 20358.591 | 3348.1185 | 20358.591 | 20358.591 |
| $R=0.6$ | 715.65 | 17483.431 | 19638.163 | 3790.2845 | 19638.163 | 19638.163 |
| $R=0.7$ | 1656.6 | 16768.038 | 18829.491 | 4545.31 | 18829.491 | 18829.491 |
| $R=0.8$ | 2494.25 | 15501.32 | 17681.401 | 4867.675 | 17681.401 | 17681.401 |

excessive computation time, whenever a problem is not solved within the time limit of *3,600 s (1h)*, the computation is stopped for that problem. For the local search and the genetic algorithm, 25 instances have been generated for each combination of the $T$ and $R$ parameters. The problems solved by these non exact algorithms are of size of *75, 100, 150, 200* and *250*.

We note that the algorithms are coded in the *C* language using concert technology technique with Cplex 10.1 and implemented on a Pentium *IV-500* personal computer.

The remaining of this section is organized as follows. First, we illustrate the impact of adding the constraints to the developed models. Then, we analyze the performance of the branch and bound algorithm. Finally, we present the results of the GA and the local search heuristic.

## The impact of the added constraints to the developed models

In Tables 2–7, the first number after LB is the number of the model used to compute the lower bounds. The remaining numbers after the model number represents the corresponding added valid constraints (cuts). For example LB1, LB2 and LB3 represent the lower bounds mean values obtained respectively from the first, the second and the third model without any added constraint. LB11, LB21, LB31 are the lower bounds mean values obtained respectively from the first, the second and the third model with the first type of constraints.

### *The impact of the added constraints (C1) (C2) and (C4) to the first model*

Table 2 shows the impact of adding the constraints (C1) (C2) and (C4) to the first model according to the mean tardy factor parameter $T$. Table 3 shows the impact of adding these constraints according to the R parameter. From these tables, we can conclude that each added constraint cited above improves the lower bounds mean value. The improvement's contribution of the mean lower bound value depends on the type of added constraints. It appears to be more important for the forth type of valid constraints and less important for the first and second types.

We also observe that the same mean lower bounds value is repeated in LB14 and LB1124 columns. Hence, we can

**Table 4** The impact of the added constraints (C1) and (C2) to the second model according to the $T$ factor

| | LB2 | LB21 | LB22 | LB212 |
|---|---|---|---|---|
| $N = 10$ | | | | |
| $T = 0.2$ | 1771.1755 | 1771.1755 | 1771.1755 | 1771.1755 |
| $T = 0.3$ | 1098.572 | 1098.572 | 1098.572 | 1098.572 |
| $T = 0.4$ | 242.748 | 242.748 | 242.748 | 242.748 |
| $T = 0.5$ | 683.1785 | 683.1785 | 683.1785 | 683.1785 |
| $T = 0.6$ | 2289.5675 | 2289.5675 | 2289.5675 | 2289.5675 |
| $T = 0.7$ | 4666.3795 | 4666.3795 | 4666.3795 | 4666.3795 |
| $T = 0.8$ | 7152.732 | 7152.732 | 7152.732 | 7152.732 |
| $N = 20$ | | | | |
| $T = 0.2$ | 10409.936 | 10409.936 | 10409.936 | 10409.936 |
| $T = 0.3$ | 6510.8925 | 6510.8925 | 6510.8925 | 6510.8925 |
| $T = 0.4$ | 1568.3635 | 1568.3635 | 1568.3635 | 1568.3635 |
| $T = 0.5$ | 1082.503 | 1082.503 | 1082.503 | 1082.503 |
| $T = 0.6$ | 6292.9575 | 6292.9575 | 6292.9575 | 6292.9575 |
| $T = 0.7$ | 13668.734 | 13668.734 | 13668.734 | 13668.734 |
| $T = 0.8$ | 25403.0815 | 25403.0815 | 25403.0815 | 25403.0815 |
| $N = 30$ | | | | |
| $T = 0.2$ | 25496.883 | 25496.883 | 25496.883 | 25496.883 |
| $T = 0.3$ | 15138.868 | 15138.868 | 15138.868 | 15138.868 |
| $T = 0.4$ | 4555.272 | 4555.272 | 4555.272 | 4555.272 |
| $T = 0.5$ | 2018.338 | 2018.338 | 2018.338 | 2018.338 |
| $T = 0.6$ | 12207.749 | 12207.749 | 12207.749 | 12207.749 |
| $T = 0.7$ | 29490.268 | 29490.268 | 29490.268 | 29490.268 |
| $T = 0.8$ | 53250.2775 | 53250.2775 | 53250.2775 | 53250.2775 |

confirm that the forth type of valid constraints dominates the other types.

*The impact of the added constraints (C1) and (C2) to the second model*

Table 4 shows the impact of adding the constraints (C1) and (C2) to the second model according to the mean tardy factor parameter $T$. Table 5 shows the impact of adding the constraints according to the R parameter. Table 6 represents the computation time spent with each linear relaxation. From these tables, we can make the main two following remarks:

1. Neither the first type of valid constraints nor the second one improves the average value of the lower bounds.
2. The second type of constraints has as impact in the reduction of the computation time needed to obtain the lower bound value.

*The impact of the added constraints (C1) (C2) and (C3) to the third model*

Table 7 shows the impact of adding the constraints (C1), (C2) and (C3) to the third model according the mean tardy factor parameter $T$. Table 8 shows the impact of adding the constraints according to the R parameter. From these tables, we can observe that each added constraint improves the average values of the lower bounds. Such an improvement depends on the type of the added constraints. It appears to be more important when we add all the proposed types of constraints.

In the two tables, we can observe that the greatest average values of the lower bounds are found in the LB3123 column. Hence, we can confirm that there is no specific type of constraints that dominates the others and that the best linear relaxation is obtained by adding the all constraints.

*The best linear lower bound*

From LB14 column in Tables 2 and 3, LB21 column in Tables 4 and 5 and LB3123 column in Tables 7 and 8, we can examine that the best average values of the lower bounds appear for LB14 and LB21. To decide which lower bound is the best, we have to compare their computation time. From Fig. 4, we observe that the average computation time of LB14 is smaller than the one of LB21. Hence, as a conclusion, we can say that from the developed models with the added constraints, the linear relaxation provided by the first model to

**Table 5** The impact of the added constraints (C1) and (C2) to the second model according the $R$ factor

| | LB2 | LB21 | LB22 | LB212 |
|---|---|---|---|---|
| $N=10$ | | | | |
| $R=0.2$ | 3000.889 | 3000.889 | 3000.889 | 3000.889 |
| $R=0.3$ | 2520.8325 | 2520.8325 | 2520.8325 | 2520.8325 |
| $R=0.4$ | 2771.614 | 2771.614 | 2771.614 | 2771.614 |
| $R=0.5$ | 2304.037 | 2304.037 | 2304.037 | 2304.037 |
| $R=0.6$ | 2538.4145 | 2538.4145 | 2538.4145 | 2538.4145 |
| $R=0.7$ | 2204.2295 | 2204.2295 | 2204.2295 | 2204.2295 |
| $R=0.8$ | 2382.318 | 2382.318 | 2382.318 | 2382.318 |
| $N=20$ | | | | |
| $R=0.2$ | 9810.1465 | 9810.1465 | 9810.1465 | 9810.1465 |
| $R=0.3$ | 9645.627 | 9645.627 | 9645.627 | 9645.627 |
| $R=0.4$ | 9862.67 | 9862.67 | 9862.67 | 9862.67 |
| $R=0.5$ | 9528.535 | 9528.535 | 9528.535 | 9528.535 |
| $R=0.6$ | 9585.317 | 9585.317 | 9585.317 | 9585.317 |
| $R=0.7$ | 9020.643 | 9020.643 | 9020.643 | 9020.643 |
| $R=0.8$ | 7677.265 | 7677.265 | 7677.265 | 7677.265 |
| $N=30$ | | | | |
| $R=0.2$ | 21396.741 | 21396.741 | 21396.741 | 21396.741 |
| $R=0.3$ | 21896.635 | 21896.635 | 21896.635 | 21896.635 |
| $R=0.4$ | 22190.691 | 22190.691 | 22190.691 | 22190.691 |
| $R=0.5$ | 20358.591 | 20358.591 | 20358.591 | 20358.591 |
| $R=0.6$ | 19638.163 | 19638.163 | 19638.163 | 19638.163 |
| $R=0.7$ | 18829.491 | 18829.491 | 18829.491 | 18829.491 |
| $R=0.8$ | 17681.401 | 17681.401 | 17681.401 | 17681.401 |

**Table 6** The computation time for the second model's linear relaxation

| | LB2 Time(s) | LB21 Time(s) | LB22 Time(s) | LB212 Time(s) |
|---|---|---|---|---|
| $N=10$ | 0.153983 | 0.0827869 | 0.2710325 | 0.1421068 |
| $N=20$ | 1.9909029 | 0.9509903 | 6.3818835 | 2.74579612 |
| $N=30$ | 12.370827 | 7.4303709 | 70.715084 | 34.3139898 |

which we add the forth type of valid constraints dominates all the other linear relaxations in terms of the average value and the computation time.

### The best lower bounds among all the proposed lower bounds

According to Fig. 5, the assignment-based lower bound (denoted as LB_Aff) has the best average value when the average tardiness factor $T$ is equal to 0,4 or 0,5. Hence, it dominates the linear lower bound $LB_{14}$ and the adapted lower bound of Li (denoted as LB_Li). For the remaining value of T, the linear lower bound $LB_{14}$ is more efficient than the other proposed lower bounds. From the same figure, we also conclude that the Lagrangian lower bound value (LB_LAG) converges to the linear lower bound value $LB_{14}$ and it does not exceed it. These observations have motivated our choice of such lower bounds as it is mentioned in the section "The branch and bound algorithm".

### The branch and bound algorithm results

Computational results concerning the average percentage deviation of the upper bound from the optimum (UBD), the average percentage deviation of the linear lower bound from the optimum (LIND), the average percentage deviation of the assignement-based lower bound from the optimum (AFFD) are reported in Table 9. In the same table, we find also the average percentage deviation of LB_Li from the optimum (LID).

In Table 10, the average time required to find the optimal solution in seconds ($AT_1$) and the average number of the

**Table 7** The impact of the added constraints (C1), (C2) and (C3) to the third model according to the $T$ factor

| | LB3 | LB31 | LB33 | LB32 | LB313 | LB3123 |
|---|---|---|---|---|---|---|
| $N=10$ | | | | | | |
| $T=0.2$ | 0 | 430.072 | 528.7665 | 1463.14 | 588.6165 | 1680.841 |
| $T=0.3$ | 0 | 129.0085 | 149.333 | 955.537 | 201.724 | 1073.5405 |
| $T=0.4$ | 0 | 4.196 | 1.8615 | 225.4945 | 4.386 | 240.0455 |
| $T=0.5$ | 0 | 37.185 | 20.177 | 672.374 | 41.6595 | 673.87 |
| $T=0.6$ | 0.4925 | 436.256 | 438.7275 | 2166.108 | 591.863 | 2203.41 |
| $T=0.7$ | 41.231 | 1989.4595 | 2183.0435 | 4297.439 | 2437.6645 | 4430.169 |
| $T=0.8$ | 272.0325 | 4053.044 | 4710.1475 | 6054.224 | 4971.8675 | 6597.8915 |
| $N=20$ | | | | | | |
| $T=0.2$ | 0 | 2188.3155 | 2952.2595 | 8704.4385 | 3127.283 | 9465.999 |
| $T=0.3$ | 0 | 365.4795 | 738.896 | 5714.693 | 795.2835 | 6141.213 |
| $T=0.4$ | 0 | 0.8545 | 6.3815 | 1462.066 | 7.236 | 1533.14 |
| $T=0.5$ | 0 | 0 | 0 | 1068.701 | 0 | 1070.2395 |
| $T=0.6$ | 0 | 304.281 | 429.6975 | 5807.7335 | 512.258 | 5868.694 |
| $T=0.7$ | 105.1225 | 2785.396 | 3928.467 | 11975.02 | 4140.972 | 12334.751 |
| $T=0.8$ | 703.2345 | 10239.5105 | 12981.863 | 21656.049 | 13366.7605 | 22755.755 |
| $N=30$ | | | | | | |
| $T=0.2$ | 0 | 4354.5195 | 7266.47 | 21124.645 | 7320.048 | 22588.2345 |
| $T=0.3$ | 0 | 344.545 | 1408.6365 | 13241.268 | 1426.923 | 13974.6 |
| $T=0.4$ | 0 | 0 | 3.458 | 4164.0345 | 3.458 | 4384.561 |
| $T=0.5$ | 0 | 0 | 0 | 1963.0965 | 0 | 1969.3795 |
| $T=0.6$ | 0 | 143.271 | 354.573 | 11271.556 | 396.4785 | 11357.969 |
| $T=0.7$ | 255.94 | 4172.279 | 7347.665 | 25602.08 | 7530.0185 | 26267.1005 |
| $T$=0.8 | 1909.0205 | 17894.723 | 25449.2295 | 44398.1835 | 25711.6345 | 46565.586 |

visited nodes ($AN_1$) using the first bounding procedure in the search tree are reported.

In Table 11, we report the average time required to find the optimal solution in seconds ($AT_2$) and the average number of the visited nodes ($AN_2$) using the second bounding procedure in the search tree.

As it can be seen in Table 9, results show that the upper bound represents an efficient initial solution for the problem with a maximum average percentage deviation of 3.76% from the optimum solution. Thus, the solutions from the heuristic are typically optimal or near optimal.

Results show also that the proposed linear lower bound is greater than the adapted lower bound produced by Li (1997) for all T values. However, when the T parameter takes a value in {0.2,0.3, 0.4, 0.5} the linear lower bound is dominated by the assignment-based lower bound. We note that the improvement of this lower bound may be possible by finding and adding other valid inequalities.

From Tables 10 and 11, we can see that the first bounding procedure based on the LB_Li and LB_Aff lower bounds is faster than the second bounding procedure (that utilizes with these two lower bounds the linear lower bound $LB_{14}$). However, the number of visited nodes in the second bounding procedure is less than the number of visited nodes in the first bounding procedure. This result confirms what we have already observed: the dominance of the linear lower bound but, its higher computation time compared to the computation time of the two other lower bounds.

We note that for the branch and bound implementation using the first bounding procedure, 77% of problems with size of 30 jobs are solved in a time period less than 20 s and 20% of problems with size of 40 are not solved in 1 hour.

## Results of the GA and the local search heuristic

In Table 12, the first column is the mean computation time for obtaining a solution by the local search technique for every $M \in \{75, 100, 150, 200, 250\}$. The second column represents the average computation time necessary to obtain a solution by the genetic algorithm. The third column is the percentage of the genetic algorithm solution's evaluations less than local search solution's evaluations. The last column represents the percentage of genetic algorithm solutions that are equal to the best found solution after resolving the problem 20 successive times by the same algorithm.

**Table 8** The impact of the added constraints (C1), (C2) and (C3) to the third model according to the $R$ factor

| | LB3 | LB31 | LB33 | LB32 | LB313 | LB3123 |
|---|---|---|---|---|---|---|
| $N=10$ | | | | | | |
| $R=0.2$ | 0 | 1269.1765 | 1480.281 | 2722.5025 | 1590.011 | 2845.2145 |
| $R=0.3$ | 2.036 | 903.105 | 986.2645 | 2262.1535 | 1089.564 | 2373.672 |
| $R=0.4$ | 11.2365 | 1042.6 | 1108.4925 | 2513.057 | 1231.3795 | 2632.4085 |
| $R=0.5$ | 10.353 | 849.122 | 976.8255 | 2036.818 | 1074.932 | 2168.9905 |
| $R=0.6$ | 84.4325 | 1024.825 | 1177.6705 | 2224.376 | 1301.814 | 2390.8385 |
| $R=0.7$ | 79.098 | 784.8155 | 917.0395 | 1906.59 | 1023.8265 | 2096.5545 |
| $R=0.8$ | 124.8485 | 1070.086 | 1241.284 | 2032.8725 | 1363.901 | 2237.1665 |
| $N=20$ | | | | | | |
| $R=0.2$ | 0 | 2414.941 | 3083.632 | 8618.046 | 3254.2735 | 8958.2715 |
| $R=0.3$ | 0 | 2493.3525 | 3239.623 | 8434.2975 | 3373.512 | 8759.845 |
| $R=0.4$ | 0.3325 | 2201.642 | 3003.279 | 8616.972 | 3089.6545 | 8967.5115 |
| $R=0.5$ | 25.0825 | 2040.9105 | 2828.35 | 8246.9715 | 2920.724 | 8641.4685 |
| $R=0.6$ | 125.1655 | 2326.0535 | 3210.292 | 8276.0775 | 3299.1185 | 8715.1605 |
| $R=0.7$ | 246.529 | 2247.9415 | 2917.112 | 7840.5315 | 3074.8225 | 8293.7535 |
| $R=0.8$ | 411.124 | 2125.0615 | 2803.4005 | 6527.362 | 2928.4345 | 6988.068 |
| $N=30$ | | | | | | |
| $R=0.2$ | 0 | 3772.46 | 5991.2945 | 18368.1 | 6056.9815 | 19020.161 |
| $R=0.3$ | 0 | 4031.247 | 6345.472 | 18892.085 | 6393.7805 | 19486.526 |
| $R=0.4$ | 0 | 4205.4895 | 6836.374 | 18975.977 | 6879.429 | 19725.349 |
| $R=0.5$ | 38.078 | 3174.627 | 5214.811 | 17452.213 | 5285.6575 | 18120.09 |
| $R=0.6$ | 199.82 | 3545.3655 | 5505.1455 | 17041.855 | 5584.601 | 17735.601 |
| $R=0.7$ | 686.897 | 4029.648 | 5766.2745 | 16126.104 | 5844.832 | 17027.406 |
| $R=0.8$ | 1240.1655 | 4080.962 | 6046.7615 | 14789.711 | 6210.9775 | 15855.414 |

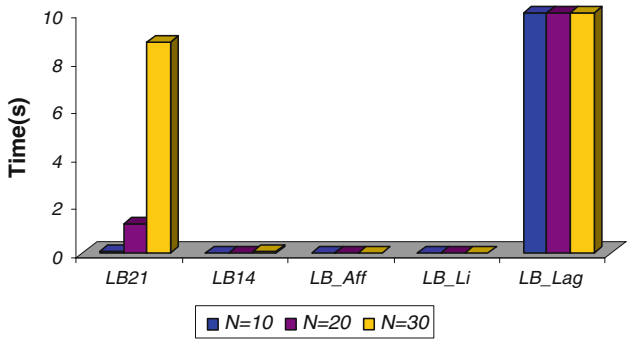


**Fig. 4** Computation time for all the proposed lower bounds

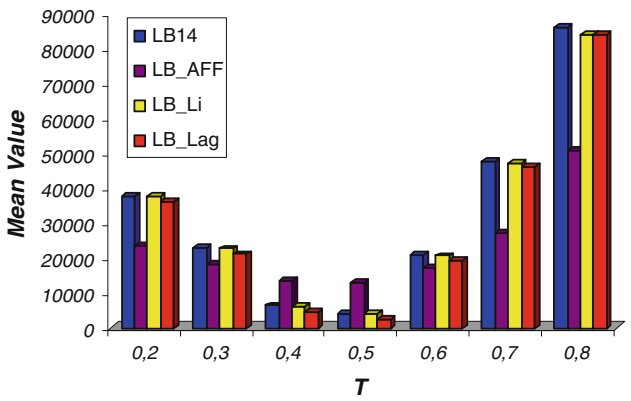


**Fig. 5** Lower bounds mean value according the average tardiness $T$

From the first and the second columns, we can conclude that the time consumed by the local search algorithm is less than the time consumed by the genetic algorithm for all the size of tested problems. From the third column, we observe that a great percentage of genetic algorithm solutions dominate the local search solutions. Moreover, this percentage increases when the size of the problem increases.

We finally note that all the genetic algorithm solutions were improved by applying the neighbourhood search to the first and the second best solutions of the genetic algorithm.

## Conclusion

In this paper we have studied the problem of scheduling a set of $M$ preventive maintenance tasks that have to be performed on $M$ machines with the aim of minimizing the total preventive maintenance cost. The execution of the maintenance tasks should be continuous during the maintenance horizon due to the expensive cost of having the maintenance

**Table 9** Computational results for the lower bounds, the branch and bound algorithm and the heuristic

| $T$ | UBD(%) | LID(%) | AFFD(%) | LIND(%) |
|---|---|---|---|---|
| $N=15$ | | | | |
| $T=0.2$ | 0.69334437 | 55.9078661 | 41.78957282 | 55.9078661 |
| $T=0.3$ | 2.04868392 | 79.95328043 | 25.97353323 | 79.95328043 |
| $T=0.4$ | 0.32660577 | 86.82745902 | 45.43229333 | 86.82745902 |
| $T=0.5$ | 0.30659282 | 66.05080917 | 31.3062986 | 66.05080917 |
| $T=0.6$ | 0.36385558 | 42.74148655 | 49.59459664 | 38.72670675 |
| $T=0.7$ | 0.89991454 | 9.353132851 | 40.44237897 | 8.206773146 |
| $T=0.8$ | 0.00422726 | 4.732189943 | 27.53484995 | 3.457753062 |
| $N=20$ | | | | |
| $T=0.2$ | 2.65154318 | 41.39008119 | 42.22655129 | 41.37799719 |
| $T=0.3$ | 0.22844428 | 62.16990829 | 37.91614887 | 62.16990829 |
| $T=0.4$ | 1.22374666 | 78.06528441 | 39.03413759 | 78.06528441 |
| $T=0.5$ | 1.84766282 | 82.49830643 | 49.73108524 | 82.49830643 |
| $T=0.6$ | 0.6000173 | 30.72410923 | 47.13489527 | 29.84256978 |
| $T=0.7$ | 0.86264869 | 15.33139645 | 48.5149824 | 14.28020528 |
| $T=0.8$ | 0.42488469 | 4.60686716 | 34.09921222 | 4.23741595 |
| $N=25$ | | | | |
| $T=0.2$ | 2.99649821 | 34.25231787 | 44.66018654 | 34.21086403 |
| $T=0.3$ | 3.33144157 | 54.15381441 | 42.51072995 | 54.14261839 |
| $T=0.4$ | 2.26135131 | 76.95075784 | 44.32517211 | 76.65997782 |
| $T=0.5$ | 1.7039384 | 74.34213305 | 46.85498085 | 74.26417236 |
| $T=0.6$ | 1.1547265 | 34.78071424 | 43.17718875 | 34.45561451 |
| $T=0.7$ | 1.05066288 | 16.46295619 | 40.99716246 | 13.80305491 |
| $T=0.8$ | 0.03244333 | 1.79712566 | 28.69710264 | 1.516577753 |
| $N=30$ | | | | |
| $T=0.2$ | 3.06594235 | 32.44566132 | 43.95661935 | 32.44205348 |
| $T=0.3$ | 2.51450407 | 47.56713754 | 46.39228906 | 47.56299762 |
| $T=0.4$ | 3.75965152 | 76.58969326 | 46.65531128 | 76.58969138 |
| $T=0.5$ | 3.01158618 | 72.80074453 | 47.9674387 | 72.66139295 |
| $T=0.6$ | 2.00986075 | 38.88462246 | 49.10509764 | 37.90673624 |
| $T=0.7$ | 0.4914925 | 14.07047262 | 44.21369813 | 12.12378265 |
| $T=0.8$ | 0.27651437 | 4.792682431 | 38.65960027 | 2.871811619 |
| $N=40$ | | | | |
| $T=0.2$ | 1.02485155 | 22.39681992 | 43.92310484 | 22.3463491 |
| $T=0.3$ | 1.17857444 | 24.47327152 | 36.79289482 | 24.36048405 |
| $T=0.4$ | 1.96960248 | 35.52037303 | 26.77511025 | 35.52037626 |
| $T=0.5$ | 1.33706698 | 35.3400558 | 25.29347647 | 34.91414251 |
| $T=0.6$ | 1.74472489 | 32.12656382 | 39.22079343 | 31.41004556 |
| $T=0.7$ | 0.4810698 | 12.49696954 | 42.2769649 | 10.89915266 |
| $T=0.8$ | 0.18419151 | 4.100042706 | 37.29018575 | 2.289487149 |

resources available. The problem was solved by three linear programs, a branch and bound algorithm, a local search approach and a genetic algorithm. For the linear programs, the best developed model is able to solve problems of size of 17 tasks in less than 1 h without dropping the constraints of the variables integrities. The same model gives the best linear relaxation among the developed models after adding a valid constraint to it. The obtained linear relaxation is equal to that provided by the time-indexed formulation that is considered, as a tight formulation. Our linear lower bound has the advantage of the computation time shortness. For the B&B algorithm, we have developed two lower bounds and an upper bound. The first lower bound is an adaptation of the lower bound proposed by Li (1997) for the problem of earliness–tardiness minimization. The second lower bound is an assignment-based lower bound. The upper bound is based on

**Table 10** Computation time and average number of the visited nodes using the first bounding procedure

| *N* | AT(s) | AN |
|---|---|---|
| 15 | 0.01 | 159 |
| 20 | 0.101 | 719 |
| 25 | 1.886 | 6.955 |
| 30 | 26.93 | 53.051 |
| 40 | 483.299 | 359.198 |

**Table 11** Computation time and average number of the visited nodes using the second bounding procedure

| *N* | AT(s) | AN |
|---|---|---|
| 15 | 1.32 | 146 |
| 20 | 8.41 | 646 |
| 25 | 95.53 | 5732 |
| 30 | Stopped | Stopped |
| 40 | Stopped | Stopped |

**Table 12** Computational results for the local search and the genetic algorithm

| *N* | LS_Time(s) | GA_Time(s) | (%)GA_Sol < LS_Sol | (%)GA_Sol= Best_GA |
|---|---|---|---|---|
| 75 | 0.888 | 1.478 | 72.8 | 88 |
| 100 | 2.811 | 4.535 | 74.4 | 84 |
| 150 | 14.702 | 23.698 | 80.7 | 82 |
| 200 | 47.546 | 77.791 | 83.5 | 80 |
| 250 | 121.832 | 198.75 | 86.7 | 77.5 |

an iterated local search heuristic. The B&B algorithm allows us to solve problems with 40 tasks in a period less than 1 h. For the Local search and the genetic algorithm, a great percentage of the solutions obtained by the genetic algorithm dominate those obtained by the local search approach. The computational results show that the deviation of the meta-heuristics solutions to the optimal one is very small, which confirms the effectiveness of meta-heuristics as new approaches for solving hard scheduling problems.

## Appendix

### *The earliness/tardiness dispatch priority rule of* Ow and Morton (1989)

$$\beta(j) = \begin{cases} w_j/p_j & \text{if} \quad s_j < 0 \\ -h_j/p_j(h_j/p_j + w_j/p_j) & \\ \quad \exp(-s_j/k\overline{p}) & \text{if} \quad 0 \le s_j \le 0 \\ -h_j/p_j & \text{if} \quad s_j > k\overline{p} \end{cases}$$

where $s_j = d_j - t_j - p_j$ is the slack of job $j$, $p_j$ is the processing time of j, $\bar{p}$ is the average processing time of the jobs, $k$ is an empirical integer, and $\beta$ (j) is the local priority value assigned to job $j$. In this dispatch priority rule, whenever the machine is available, and there are jobs waiting to be processed, the dispatch rule is used to select the next job. However, the schedule produced by this earliness/tardiness rule, as it is shown in their computational study Ow and Morton (1989), is far from the optimal. Therefore, we proposed a new dispatch priority rule based on the earliness/tardiness dispatch priority rule to select the next unscheduled job.

**Acknowledgments** This work has been funded by "Conseil Régional Champagne Ardenne" (OCIDI Project) when the second author was with Charles Delaunay Institute (LOSI team) at UTT.

## References

Abdul-Razaq, T., & Potts, C. N. (1988). Dynamic state space relaxation for single machine scheduling. *Journal of Operations Research Society, 39*(2), 141–152.

Aghezzaf, E. H., Jamali, M. A., & Ait-Kadi, D. (2007). An integrated production and preventive maintenance planning model. *European Journal of Operational Research, 181*, 679–685.

Aghezzaf, E. H., & Najid, N. M. (2008). Integrated production planning and preventive maintenance in deteriorating production systems. *Information Sciences, 178*, 3382–3392.

Azizoglu, M., Kondakci, S., & Kirca, O. (1991). Bicriteria scheduling problem involving total tardiness and total earliness penalties. *International Journal of Production Economics, 23*, 17–24.

Baker, K. R., & Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: a review. *Operations Research, 38*, 22–36.

Graves, H. G., & Lee, C.-Y. (1999). Scheduling maintenance and semiresumable jobs on a single machine. *Naval Research Logistics, 46*, 845–863.

Held, M., & Karp, R. M. (1962). A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics, 10*, 196–210.

Kacem, I. (2007). Lower bounds for tardiness minimization on a single machine with family setup times. *International Journal of Operations Research, 4*, 18–31.

Kubiak, W., Blazewicz, J., Formanowicz, P., Breit, J., & Schmidt, G. (2002). Two-machine flow shops with limited machine availability. *European Journal of Operational Research, 136*, 528–540.

Lee, C.-Y. (1996). Minimising the makespan in the two machine scheduling scheduling problem with an availability constraint. *Operational Research Letters, 20*, 129–139.

Lee, C.-Y., & Chen, Z.-L. (2000). Scheduling jobs and maintenance activities on parallel machines. *Naval Research Logistics, 47*, 145–165.

Li, G. (1997). Single machine earliness and tardiness scheduling. *European Journal of Operational Research, 26*, 546–558.

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

M'Halla, R. (2006). Minimising total earliness and tardiness on a single machine using a hybrid heuristic. *Computers & Operation Research, 34*, 3126–3142.

Ow, P. S., & Morton, E. T. (1989). The single machine early/tardy problem. *Management Science, 35*, 331–342.

Potts, C. N., & Van Wassenhove, L. N. (1984). A branch and bound algorithm for the total weighted tardiness problem. *Operations Research, 33*, 363–377.

Sakib, A. M., & Anup, K. S. (2001). Single machine weighted earliness/tardiness penalty problem with a common due date. *Computers & Operation Research, 28*, 649–669.

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

Smith, W. E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly, 3*, 59–66.

Van den Akker, M., Hoogeveen, J. A., & van deVelde, S. (2002). Combining column generation and lagrangean relaxation to solve a single-machine common due date problem. *INFORMS Journal on Computing, 14*, 37–51.

Valente, J. M. S., & Shaller, J. E. (2010). Improved heuristics for the single machine scheduling problem with linear early and quadratic tardy penalties. *European Journal of Industrial Engineering, 4*, 99–129.

Valente, J. M. S. (2010). Heuristics for the single machine scheduling problem with early and quadratic tardy penalties. *European Journal Industrial Engineering, 4*, 431–448.