Lower Bounds for Parallel Machine Scheduling Problems

Philippe Baptiste[1], **Antoine Jouglet**[2], **David Savourey**[2]

[1] Ecole Polytechnique, UMR 7161 CNRS LIX, 91128 Palaiseau, France

philippe.baptiste@polytechnique.fr

[2] Heudiasyc, UMR CNRS 6599, Université de Technologie de Compiègne

BP 20529, 60205 Compiègne, France

{antoine.jouglet, david.savourey}@hds.utc.fr

February 8, 2009

Abstract

We study the parallel machine scheduling problem with release dates and we consider several "min-sum" objective functions including total weighted tardiness, total tardiness, total weighted completion time and total completion time. We describe several lower bounds for these problems, most of them being original ones. We provide experimental results to compare these lower bounds according to their quality and of their computational time requirement.

Key words: Scheduling, Identical Parallel Machines, Total Cost, Lower Bounds.

1 Introduction

In this paper we consider the situation where a set of n jobs $N = \{1, \ldots, n\}$ has to be processed on m identical parallel machines and where the objective is to minimize a sum objective function. Several criteria are studied: total (weighted) tardiness and total (weighted) completion time. Associated with each job i is a release date r_i, a processing time p_i, a due date d_i and a weight w_i. A job cannot start before its release date and preemption is not allowed. No more than m jobs can be scheduled simultaneously. The tardiness of job i is defined as $T_i = \max(0, C_i - d_i)$, where C_i is the completion time of job i. The problem is to find a feasible schedule with minimum total (weighted) tardiness $\sum(w_i)T_i$ or with minimum total (weighted) completion time $\sum(w_i)C_i$. Note that all criteria are special cases of the total weighted tardiness criterion. These problems are denoted as $Pm|r_i|\sum w_iT_i$, $Pm|r_i|\sum T_i$, $Pm|r_i|\sum w_iC_i$ and $Pm|r_i|\sum C_i$. As these problems are strongly NP-Hard [13], it is essential to have good and fast lower bounds. In this paper, we propose a brief survey of existing bounds and we present several new ones.

1.1 Related Work

The $Pm|r_i|\sum C_i$ problem has been studied in [16]. The authors describe two lower bounds. The first one consists in relaxing all release dates to the earliest one, and solving exactly the $Pm||\sum C_i$ problem (this can be done in polynomial time [8]). The second lower bound uses "Job Splitting". Preemption and simultaneous execution of parts of a job are allowed. The optimal value of the relaxed problem (solvable in polynomial time), is a valid lower bound of the initial problem.

The earliness tardiness problem, denoted $Pm|r_i|\sum \alpha_i E_i + \beta_i T_i$ has also been studied, mainly in [12]. The authors use a time indexed formulation, and propose several relaxations to compute lower bounds for the problem. The special case where $m = 1$ (single machine problem) has also been extensively studied.

In [10, 11], lower bounds are presented for the criteria studied in this paper. In [5], Rivreau describes a lower bound based on a Lagrangian relaxation of a time indexed formulation of the single machine problem $1|r_i|F_i$, where the cost function F must comply with some specific constraint.

1.2 Outline of the Paper

Lower bounds relying on similar relaxation techniques are described in the same section. In Section 2, we relax release dates. In Section 3, reduce the problems to flow problems. In Section 4, we focus on lower bounds relying on i-th minimal completion times. In Section 5, we describe lower bounds based on different relaxations of a time indexed mixed-integer formulation. Finally, experimental results and comparisons between all lower bounds are provided in Section 6.

All examples presented in this paper are based upon the instance of the total weighted tardiness problem, described in Table 1 with $m = 2$ and $n = 5$. The same instance will also be used for other "easier" (for total completion time, we consider only release dates and processing times, *etc.*).

i	1	2	3	4	5
r_i	0	1	3	4	5
p_i	6	4	3	1	4
d_i	7	5	8	7	12
w_i	1	4	1	2	3

Table 1: The Instance Used over the Paper.

2 Relaxing Release Dates

The two following lower bounds (Sections 2.1 and 2.2) deal only with total completion time. However, they can be used for more complex criteria (Section 2.3).

2.1 Relaxing all Release Dates to the Earliest Release Date

This lower bound has been proposed by Yalaoui and Chu in [16] for the total completion time criterion. All release dates are relaxed to the minimal release date among all release dates. The relaxed problem reduces to $Pm||\sum C_j$ which is polynomially solvable [8]. All jobs are scheduled according to the Shortest Processing Time (SPT) Rule on the earliest available machine. That is to say that the job with the shortest processing time is scheduled on the machine which is available the earliest. This lower bound can be then computed in $O(n \log n)$. From now on we shall refer to this lower bound as $lb_{no-release}$ (*Release dates are relaxed to the smallest one*).

The optimal solution for the total completion time problem with relaxed release dates is provided in Figure 1. The value of the lower bound is $lb_{no-release} = 27$.

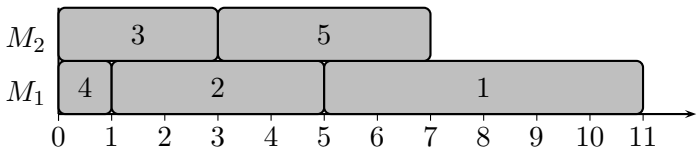

Figure 1: Optimal solution when all release dates are relaxed to the earliest release date

2.2 Splitting Problem into Relaxed Sub-Problems

In this section, we describe a lower bound in which several subsets of jobs are build according to the value of their release dates. The main idea is to relax the release dates of jobs according to the minimal release date among the jobs of the same subset. We then build the optimal solutions of these subsets independently using the SPT rule described in the previous section. The sum of the costs of each of these schedules is then a lower bound of the original problem. In the following, we only use a separation in two subsets. Nevertheless note that the mechanism can be generalized to a greater number of subsets.

Let t be a given date. We define $N_0 = \{i \in N / r_i < t\}$ and $N_t = \{i \in N / r_i \geq t\}$. The initial problem is now split into two relaxed sub-problems: (1) schedule optimally jobs of N_0 from the earliest release date and (2) schedule optimally jobs of N_t from date t. These two sub-problems can be solved in polynomial time, and the sum of the optimal values of these sub-problems is denoted $\Gamma(t)$. The value $\Gamma(t)$ is a lower bound of the initial problem. Every time point t can be chosen to compute a lower bound. We want to pick one which maximizes $\Gamma(t)$. In fact, it is sufficient to consider only the dates corresponding to the release dates of jobs to find the best value Γ. From now on we shall refer to this lower bound as $lb_{no-release-subsets}$ (*Decomposition of the problem into a fixed number of problems without release dates*).

This bound can be computed in $O(n^2)$ since at most n lower bounds in $O(n)$ are computed, after a single step of sorting jobs in non-decreasing order of processing times.

In Table 2, one can see the different values of $\Gamma(t)$. The maximum value of $\Gamma(t)$ is reached for $t = 4$ and $lb_{no-release-subsets} = 29$. The two sub-schedules are provided on Figure 2.

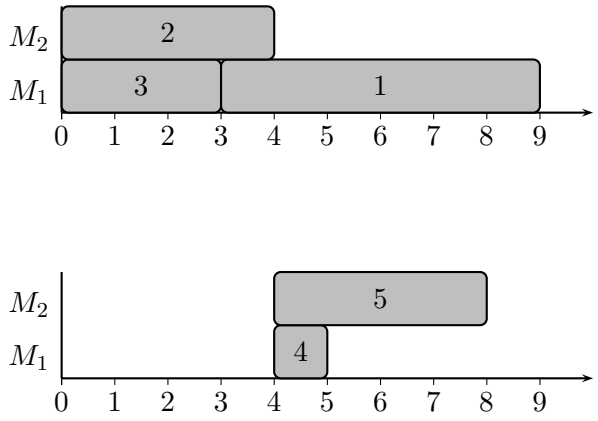

Figure 2: Best Solution for $t = 4$

t	0	1	3	4	5
$\Gamma(t)$	27	26	28	**29**	27

Table 2: Values of Γ

2.3 Extension to Total Tardiness

The lower bounds described above (Sections 2.1 and 2.2) deal only with total completion time. However, a lower bound can be derived for the total tardiness problem, by removing $\sum_{i=1}^{n} d_i$ to the obtained value for the total completion time problem [1]. Indeed, for all schedules (particularly for an optimal one), we have $\sum_{i=1}^{n} \max(0, C_i - d_i) \geq \max(0, \sum_{i=1}^{n} C_i - d_i)$. For example, if a lower bound is equal to V on the instance example, then we know that $V' = \max(0, V - \sum_{i=1}^{n} d_i) = \max(0, V - 39)$ is a lower bound for the total tardiness problem.

3 Flow Based Lower Bounds

These lower bounds deal with the four studied criteria and is based on relaxation of the problem which leads to a flow problem. In Section 3.1, we focus on the total weighted tardiness criterion which is the most general one. In Section 3.1, we show how the computed lower bound can be improved in the case of the total (weighted) completion time case.

3.1 The total (Weighted) Tardiness Case

Associated with our initial problem π, we build a problem π' as follows: each job i of the initial problem π is split (see *e.g.*, [3]) into p_i pieces J_{ik}, $k \in 1, \ldots, p_i$ of processing time $p_{ik} = 1$. With each piece J_{ik}, we associate the weight $\frac{w_i}{p_i}$ and the due date d_i.

Proposition 3.1. *The optimum of problem π' is lower than or equal to the optimum of problem π.*

In [3], the single machine case is presented. The proof is also valid for the parallel machine case, so we do not show the proof here.

A lower bound of π' can be computed considering the following flow based problem. The associated network is a connected and directed graph $G(X, U)$, where X is the set of vertices and U the set of edges. The set X is made of a source S, a sink T, n vertices associated with the n jobs N and u vertices associated with consecutive intervals $I_1, I_2, \ldots, I_u$ that partition $[r_{min}, H)$ where H is the horizon. These intervals are obtained from a set of non-decreasing dates $t_1 = r_{min}, t_2, \ldots, t_{u+1} = H$, by the relation $I_k = [t_k, t_{k+1}[$ (see Figure 3). We denote by l_k the length of interval I_k, *i.e.* $l_k = t_{k+1} - t_k$.

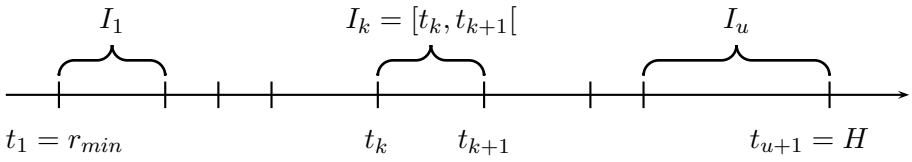

Figure 3: Intervals I_1, I_2, ..., I_u

S is connected to each vertex i by an edge of capacity p_i and of cost 0. Each vertex I_k is connected to the sink T by an edge of capacity ml_k and of cost 0. At last, for each job i and each time interval I_k such that $r_i \geq t_k$, vertex i is connected to vertex I_k by an edge of capacity l_k and of cost c_{ik} (see figure 4). The network is shown in Figure 3. The edges are labeled with pairs (x, y) where x is the edge capacity and y is the cost per unit flow.

It is easy to see that the maximal flow is equal to $\sum p_i$. The following proposition holds:

Proposition 3.2. *If $\forall k$ $l_k = 1$ and each cost $c_{ik} = \frac{w_i}{p_i} \max(0, t_k + 1 - d_i)$, then the problem to find a minimal cost flow which achieves the maximum total flow through the network solves exactly the problem π'. The obtained value is then a lower bound of the total weighted tardiness of the initial problem π.*

Nevertheless note that the number of intervals to consider is $P = \sum p_i$ which can be very large. To obtain a polynomial lower bound we can consider a weaker version in which the set of dates to consider to obtain intervals is $\{r_i\} \cup \{d_i\}$. Values c_{ik} are chosen in such a way that the cost of optimal flow is a lower bound of the cost the optimal schedule for problem π'. We can see that $c_{ik} = \frac{w_i}{p_i} \max(0, t_k + 1 - d_i)$

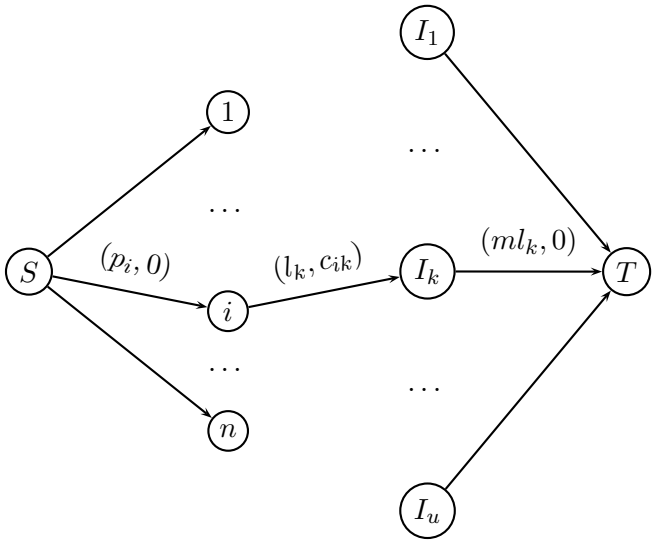

Figure 4: Flow Problem

are such values since each piece of job i allocated to an interval k has a completion time at least equal to $t_k + 1$. Since we have now $O(n)$ vertices and $O(n^2)$ edges, the lower bound can be computed in $O(n^3 log(n)^2)$ [15]. From now on we shall refer to this lower bound as lb_{flow} (*Reduction to a flow Problem*).

The described time splitting may be very poor. It is easy to build an instance with very large intervals. To avoid this drawback, n new breakpoints are added. To insert a breakpoint, all time intervals are checked, and the largest one is split into two equal parts. As the number of intervals is still linear, this lower bound, denoted as $lb_{flow'}$ (*Reduction to a flow problem with a polynomial number of time intervals*), is again $O(n^3 log(n)^2)$.

It can be interesting to bound the length of any time intervals, even if the lower bound becomes pseudo-polynomial. The way to obtain these time intervals is the following one: the initial splitting is made with $\{r_i\} \cup \{d_i\}$. Then, while there is an interval larger than an parameter a, it is split into two parts. This lower bound is denoted $lb_{flow''}(a)$ (*Reduction to a flow problem with a pseudo-polynomial number of time intervals*).

3.2 The total (Weighted) Completion Time Case

Based on ideas from lower bounds of Belouadah, Posner and Potts [3] for the problem $1|r_i|\sum w_i C_i$, we propose to improve the value of the lower bound for the total (weighted) completion time criterion. The idea is to add a quantity which can be seen as the cost of breaking each job i into p_i pieces. For the total (weighted) completion time criterion the following proposition, which is a particular case of splitting of [3], holds:

Proposition 3.3. *The value of an optimal solution of problem π', to which the quantity $\sum_{i=1}^{n} w_i(p_i-1)/2$ is added, is lower than or equal to the value of an optimal solution of the initial problem π.*

Proof. Let S be an optimal schedule of the initial problem π. Build the corresponding solution S' of the problem π' in which each job i are replaced by the corresponding pieces J_{ik}, $k \in 1,\ldots,p_i$ in problem π'. The cost of schedule S is $WC(S) = \sum_{i=1}^{n} w_i C_i(S)$ where $C_i(S)$ is the completion time of job i in schedule S. The cost of schedule S' is then $WC(S') = \sum_{i=1}^{n} w_i/p_i \sum_{k=1}^{p_i} C_i(S) - k + 1 = \sum_{i=1}^{n} w_i C_i(S) + \sum_{i=1}^{n} w_i/p_i \sum_{k=1}^{p_i} 1 - k = WC(S) - \sum_{i=1}^{n} w_i(p_i-1)/2$. We conclude that the value of an optimal solution of π', to which the quantity $\sum_{i=1}^{n} w_i(p_i-1)/2$ is added, is lower than the value of an optimal solution of π. □

On the example, the optimal flow between vertices N and vertices $I_i, i \in \{1,\ldots,u\}$ is presented on Table 3. The cost of the optimal flow is $23,333....$ After adding the value $\sum_{i=1}^{n} w_i(p_i-1)/2$, here equal to 14, the lower bound obtained is equal to 38.

	I_1	I_2	I_3	I_4	I_5
1	1	2	0	0	3
2	0	2	1	0	1
3	0	0	1	1	1
4	0	0	0	1	0
5	0	0	0	0	4

Table 3: Optimal Flow on the Example

4 Minimal Completion Times Based Lower Bounds

Let $[i](S)$ be the job completed in i^{th} position in S and $C_{[i]}(S)$ its completion time. We define $\underline{C}[i]$, $i \in N$ as the minimal completion time of a job completed in i^{th} position in a schedule among all possible schedules. Based on this notion, we derive several lower bounds. In Section 4.1, we show how lower bounds can be computed from $\underline{C}$ values for each criterion. Next, we study simple lower bounds of $\underline{C}$ values in Section 4.2. In Section 4.3, we recall a lower bound based on "Job Splitting". Then, we present some new lower bounds based on Horn Theorem in Section 4.4. Finally, a combination two fast and non dominated lower bounds is presented in Section 4.5.

4.1 Obtaining Lower Bounds from Minimal Completion Times

Let $\lambda[i]$, $i \in N$ denote a value such that $\lambda[i]$ is a lower bound of $\underline{C}[i]$. Let S be an optimal schedule of an instance of the considered problem. Note that by definition we have $\forall i \in N, \lambda[i] \leq \underline{C}[i] \leq C_{[i]}(S)$.

In the following, we explain how lower bounds can be computed from λ values for each criterion.

4.1.1 $Pm|r_i|\sum C_i$

We have $\sum_{i=1}^{n}\lambda[i] \leq \sum_{i=1}^{n}\underline{C}[i] \leq \sum_{i=1}^{n}C_{[i]}(S)$. Therefore, $\sum_{i=1}^{n}\lambda[i]$ is a lower bound of the problem since S is optimal.

4.1.2 $Pm|r_i|\sum w_iC_i$

Let $w'_1 \leq w'_2 \leq \ldots \leq w'_n$ be the list of the weights of jobs which have been resorted in non-increasing order. Note that $w_1C_1 + w_2C_2 \geq w_2C_1 + w_1C_2$ if $w_1 \leq w_2$ and $C_1 \leq C_2$. By interchange argument, it can be proven that $\sum_{i=1}^{n} w'_iC_{[i]}(S) \leq \sum_{i=1}^{n} w_{[i](S)}C_{[i]}(S)$. We have then $\sum_{i=1}^{n} w'_i\lambda[i] \leq \sum_{i=1}^{n} w'_iC_{[i]}(S)$. Therefore, $\sum_{i=1}^{n} w'_i\lambda[i]$ is a lower bound of the problem.

4.1.3 $Pm|r_i|\sum T_i$

Let $d'_1 \leq d'_2 \leq \ldots \leq d'_n$ be the list of the due dates of jobs which have been resorted in non-decreasing order. Note that $\max(0, C_1 - d_1) + \max(0, C_2 - d_2) \geq \max(0, C_1 - d_2) + \max(0, C_2 - d_1)$ if $C_1 \leq C_2$ and $d_1 \geq d_2$. By interchange argument, it can be proved that $\sum_{i=1}^{n}\max(0, C_{[i]}(S) - d'_i) \leq \sum_{i=1}^{n}\max(0, C_{[i]}(S) - d_{[i](S)})$.
It follows that $\sum_{i=1}^{n}\max(0, \lambda[i] - d'_i) \leq \sum_{i=1}^{n}\max(0, C_{[i]}(S) - d_{[i](S)})$. Therefore, $\sum_{i=1}^{n}\max(0, \lambda[i] - d'_i)$ is a lower bound of the problem.

4.1.4 $Pm|r_i|\sum w_iT_i$

We have $\sum_{i=1}^{n} w_i\max(0, C_{[i]}(S) - d_{[i](S)}) = \sum_{i=1}^{n} w_i\max(d_{[i](S)}, C_{[i]}(S)) - \sum_{i=1}^{n} w_id_{[i](S)} \geq \sum_{i=1}^{n} w_iC_{[i]}(S) - \sum_{i=1}^{n} w_id_{[i](S)}$. Therefore $\sum_{i=1}^{n} w'_i\lambda[i] - \sum_{i=1}^{n} w_id_i$ is a lower bound of the problem.

4.2 Simple Lower Bounds

In this section, we describe two simple ways to compute lower bounds λ of the $\underline{C}$ values.

At first, without lost of generality, suppose that jobs are indexed in non-decreasing order of $r_i + p_i$. We can remark that a job completed in i^{th} position cannot be completed before $r_i + p_i$. Thus, we deduce that $r_i + p_i$ is a lower bound of $\underline{C}[i]$. From now on we shall refer to these lower bounds of $\underline{C}[i]$ values as $\lambda^{r+p}[i] = r_i + p_i$, $i \in N$. Moreover, we shall refer to the lower bound associated with the λ^{r+p} values as $lb_{[\ ]r+p}$ (*Minimal completion times based on* $r_i + p_i$).

Suppose now that all release dates are relaxed to the minimal release date and that preemption is allowed. The problem $P|pmtn|C_{max}$ is polynomially solvable in $O(n)$ by McNaughton algorithm [14] with $C^*_{max} = \max(\min_{i\in N} r_i + \lceil \frac{1}{m}\sum_{i=1}^{n} p_i \rceil, \max_{i\in N} p_i)$. Finding the subset of i jobs that leads to the minimal makespan can be done by choosing the jobs with the i smallest processing times. Suppose that jobs are indexed in non-decreasing order of processing times. We have $\min_{k\in N} r_k + \left\lceil \frac{1}{m}\sum_{k=1}^{i} p_k \right\rceil$ is a lower bound of $\underline{C}[i]$. From now on we shall refer to these lower bounds of $\underline{C}$ values as $\lambda^{spt}[i] =$

$\min_{k \in N} r_k + \left\lceil \frac{1}{m} \sum_{k=1}^{i} p_k \right\rceil$, $i \in N$. Moreover, we shall refer to the lower bound associated with the λ^{spt} values as $lb_{[\ ]spt}$ (*Minimal completion times based on McNaughton's rule*).

On the example instance, the λ^{r+p} and λ^{spt} values are provided in Table 4. On this instance, the lower bounds $lb_{[\ ]r+p}$ and $lb_{[\ ]spt}$ computed for the total completion time criterion are respectively equal to 30 and 29.

jobs	1	2	3	4	5
λ^{r+p}	6	5	6	5	9
λ^{spt}	1	3	5	7	11

Table 4: λ^{r+p} and λ^{spt} Values.

4.3 Job Splitting

This lower bound consists in allowing preemption and "Splitting" *i.e.* simultaneous execution of parts of a same job on several machines [16]. The optimal schedule of this relaxed problem is built by sequencing jobs according to the Extended Shortest Remaining Processing Time rule: at time t, the job k with the shortest remaining processing time is chosen. Then, unit parts of job k are scheduled as soon as possible, possibly on different machines during the same time unit, until the job is completed or a new release date has been reached. The i^{th} lowest completion time of a job in this optimal schedule is a lower bound of $\underline{C}[i]$ [16]. From now on we shall refer to these lower bounds of $\underline{C}$ values as $\lambda^{split}[i]$, $i \in N$. Moreover, we shall refer to the lower bound associated with the λ^{split} values as $lb_{[\ ]split}$ (*Minimal completion times based on job splitting*).

On Figure 5, we can see the optimal schedule of the relaxed problem for the example instance and the λ^{split} values. For the total completion time criterion, we have then $lb_{[\ ]split} = 29$.

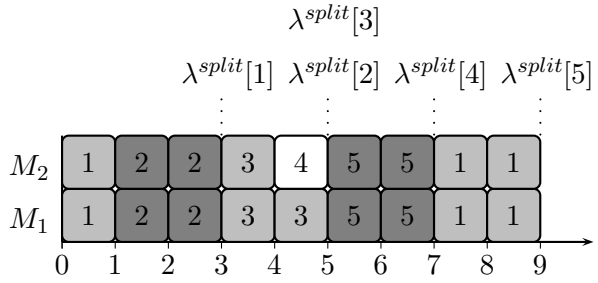

Figure 5: Optimal Solution when Job Splitting is Allowed

4.4 Using Horn Theorem

In this section we describe a set of lower bounds relying on Horn theorem [9]. This theorem allows us to solve polynomially the $Pm|r_i, pmtn|C_{max}$ problem and can be reformulated in this way:

Theorem 4.1 ([9]). *There exists a preemptive schedule in which all jobs are completed before time C if and only if:*

$$\begin{cases} \forall j \in N, \sum_{k=1}^{n} r(k,j) \leq m(C - r_j) \\ \forall j \in N, r_j + p_j \leq C \end{cases}$$

where $r(k,j) = \max(0, \min(p_k, r_k + p_k - r_j))$.

Note that $r(k,j)$ is the minimal amount of time of job k that must be executed after time r_j.

This theorem can be used to compute lower bounds of $\underline{C}$ values relaxing the problem in the preemptive case. For each $i \in N$, the idea is to find the subset of i jobs that leads to the minimal makespan. The obtained makespan is then a lower bound of $\underline{C}[i]$. Indeed, the minimal completion time of the job completed in the i^{th} position in a preemptive schedule is obviously lower than or equal to the minimal completion time of the job completed in the i^{th} position in a non preemptive one. From now on we shall refer to this lower bound of $\underline{C}[i]$ values as $\lambda^{mip}[i]$. Moreover, we shall refer to the lower bound associated with the λ^{mip} values as $lb_{[\ ]mip}$ (*Minimal completion times based on a MIP formulation relying on Horn's Theorem*). Based on Horn's theorem, we propose the following Mixed-Integer Formulation which allows us to find the subset of i jobs that minimizes the makespan $\lambda^{mip}[i]$:

$$\forall j \in N,\ r_j X_j + \frac{1}{m} \sum_{k=1}^{n} r(k,j) X_k \leq \lambda^{mip}[i] \tag{1}$$

$$\forall j \in N,\ (r_j + p_j) X_j \leq \lambda^{mip}[i] \tag{2}$$

$$\sum_{k=1}^{n} X_k = i \tag{3}$$

$$\forall j \in N,\ X_j \in \{0;1\} \tag{4}$$

i	1	2	3	4	5
λ^{mip}	5	5	6	8	10
subsets	$\{4\}$	$\{2,4\}$	$\{1,2,4\}$	$\{1,2,3,4\}$	$\{1,2,3,4,5\}$

Table 5: Subsets of Jobs and λ^{mip} Values

In this particular case, the corresponding subsets are incremental, but this property does not always hold.

4.4.1 Relaxing the Integrity Constraint

To reduce the amount of processing time required to compute the λ^{mip} values, we propose to use a continuous relaxation of the previous one. From now on we shall refer to this lower bound of $\underline{C}[i]$ values as $\lambda^{lp}[i]$. We shall refer to the lower bound associated with the λ^{lp} values as $lb_{[\,]lp}$ (*Continuous relaxation of* $lb_{[\,]mip}$).

4.4.2 Resorting Costs

The previous Mixed Integer Formulation may be very long to solve. To avoid this limitation, we propose here a way to compute a lower bound of λ^{mip} values using Horn theorem. This method is polynomial.

For each job j we define $r'(k,j)$, $k \in N$ obtained by resorting values $r(k,j)$, $k \in N$ in non decreasing order. Obviously, we have $\forall j \in N, \sum_{k=1}^{i} r'(k,j) \leq \sum_{k=1}^{n} r(k,j)X_k$ since, in the best case, the jobs with the i shortest $r(k,j)$ will be set with $X_k = 1$. Therefore, we have $\forall j \in N, r_j X_j + \frac{1}{m}\sum_{k=1}^{i} r'(k,j) \leq r_j X_j + \frac{1}{m}\sum_{k=1}^{n} r(k,j)X_k \leq \lambda^{mip}[i]$.

Suppose that job j is kept in the subset of i jobs and define $\theta_j = r_j + \frac{1}{m}\sum_{k=1}^{i} r'(k,j)$: we have $\theta_j \leq r_j + \frac{1}{m}\sum_{k=1}^{n} r(k,j)X_k \leq \lambda^{mip}[i]$. Let $\theta'_j, j \in N$ be the values obtained by sorting the θ_j values in non decreasing order. Note that $\max_{k\in N} X_k\theta_k = \lambda^{mip}[i]$. It then becomes trivial that $\max_{k\in\{1,\ldots,i\}} \theta'_k \leq \max_{k\in N} X_k\theta_k = \lambda^{mip}[i]$. We conclude that θ'_i is a lower bound of $\lambda^{mip}[i]$.

Moreover, according to the Constraint (2) of the Mixed Integer Formulation, we should have $\forall j \in N, (r_j + p_j)X_j \leq \lambda^{mip}[i]$. In the best case, the jobs j with the shortest quantity $r_j + p_j$ are kept in the optimal subset of i jobs. Relying on the notations of Section 4.2, we conclude that we should also have $\lambda^{r+p}[i] \leq \lambda^{mip}[i]$.

Therefore, the quantity $\max(\theta'_i, \lambda^{r+p}[i])$ is a lower bound of $\lambda^{mip}[i]$ and then a lower bound of $\underline{C}[i]$. From now on we shall refer to these lower bounds of $\underline{C}$ values as $\lambda^{resort}[i] = \max(\theta'_i, \lambda^{r+p}[i])$, $i \in N$. Moreover, we shall refer to the lower bound associated with the λ^{resort} values as $lb_{[\,]resort}$ (*A relaxation of* $lb_{[\,]mip}$ *based on "resorting costs"*).

Quantities $r'(k,j)$ are computed in $O(n^2 \log n))$ since n sorts of n values are performed. Next, values $\lambda^{resort}[i]$ can be incrementally computed: for each i, the n values of θ are computed in $O(n)$ considering the θ values of the previous iteration and are sorted in $O(n^2 \log n)$. Thus, the lower bound $lb_{[\,]resort}$ can be computed in $O(n^2 \log n)$.

In Table 6, we can see the way to compute $\lambda^{resort}[4]$ on the example instance. We know that 4 jobs must be chosen, so the value of θ'_4 is the 4^{th} smallest value among $\{6, 6, 7, 7, 7.5\}$. Thus, $\theta'_4 = 7$. In addition, $\lambda^{r+p}[4] = 6$ and then $\lambda^{resort}[4] = \max(7, 6) = 7$. The total value of lower bound $lb_{[\,]resort}$ on the example equals 34 for the total completion time problem.

$$r(k,j) = \begin{pmatrix} 6 & 5 & 3 & 2 & 1 \\ 4 & 4 & 2 & 1 & 0 \\ 3 & 3 & 3 & 2 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 4 & 4 & 4 & 4 & 4 \end{pmatrix} \quad r'(k,j) = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 3 & 3 & 2 & 1 & 0 \\ 4 & 4 & 3 & 2 & 1 \\ 4 & 4 & 3 & 2 & 1 \\ 6 & 5 & 4 & 4 & 4 \end{pmatrix}$$

j	1	2	3	4	5
θ_j	6	7	7.5	7	6

Table 6: Computing for $\lambda^{resort}[4]$

4.5 Combining Polynomial Lower Bounds

In the previous sections, we have described several ways to compute λ values in polynomial time. In this section, we propose to take advantage of all these lower bounds. Indeed, when values of λ_i are computed, the highest value is not always reached by the same method. The idea is then to take for each i, the maximum λ value we can compute in a reasonable time. We then define for each $i \in N$, the value $\lambda^{combo}[i]$ as:

$$\lambda^{combo}[i] = \max(\lambda^{split}[i], \lambda^{resort}[i]) \tag{5}$$

Moreover, we shall refer to the lower bound associated with the λ^{combo} values as $lb_{[\ ]combo}$ (*Combination of* $lb_{[\ ]resort}$ *and* $lb_{[\ ]split}$). Note that the complexity to compute $lb_{[\ ]combo}$ is then $O(n^2 \log n)$. Table 7 presents values of $lb_{[\ ]combo}$ for the example instance.

k	1	2	3	4	5
$\lambda^{resort}[k]$	5	5	6	7	10
$\lambda^{split}[k]$	3	5	5	7	9
$\lambda^{combo}[k]$	5	5	6	7	10

Table 7: λ^{combo} Values.

5 Time Indexed Formulation Based Lower Bounds

In the following H is an upper bound of the makespan of any active schedules for these criteria. Note that finding the maximal makespan of an active schedule is NP-Hard [2]. The two following lower bounds use a Mixed-Integer Formulation that solve exactly the four problems we study. In this formulation, the variables are:

$$x_{jt} = \begin{cases} 1 \text{ if job } j \text{ begins at time } t \\ 0 \text{ otherwise} \end{cases}$$

Then, we use the following MIP:

$$\min_{j \in N, t \in [0,H[} \sum_{j=1}^{n} \sum_{t=0}^{H} x_{jt} f_j(t + p_j) \tag{6}$$

$$\forall j \in N, \forall t < r_j, x_{jt} = 0 \tag{7}$$

$$\forall t \in [0, H[, \sum_{j=1}^{n} \sum_{t'=t-p_j+1}^{t} x_{jt'} \leq m \tag{8}$$

$$\forall j \in N, \sum_{t=0}^{H} x_{jt} = 1 \tag{9}$$

$$\forall t \in [0, H[, \forall j \in [1, n], x_{jt} \in \{0, 1\} \tag{10}$$

Constraint (7) states that a job can not start before its release date. At most m different jobs can be executed during a unit of time by Constraint (8). Each job has to begin his execution once and only once (Constraint (9)).

5.1 Relaxing Integrity Constraint

To compute a lower bound, we first relax the integrity constraint, *i.e.* Constraint (10) is replaced by:

$$\forall t \in [0, H[, \forall j \in N, x_{jt} \in [0, 1]$$

From now on we shall refer to this lower bound as lb_{t-lp} (*Continuous relaxation of* lb_t). Note that in the particular case where the optimal values of the formulation are integer, the value of lb_{t-lp} is optimal.

5.2 Lagrangian Relaxation on Resource Constraint

We can also use the Lagrangian relaxation on the resource constraint to get a lower bound, from this formulation. Constraint (8) of non temporal overlapping is relaxed to be put in the objective function. The Lagrangian multipliers are $\lambda_t, t \in [1, H]$. The formulation becomes:

$$L(\lambda) = \min_{x_{jt}} \sum_{j=1}^{n} \sum_{t=0}^{H} x_{jt} f_j(t + p_j) + \sum_{t=0}^{H} \lambda_t \left(\sum_{j=1}^{n} \sum_{t'=t-p_j+1}^{t} x_{jt'} - m \right)$$

$$\forall j \in N, \forall t < r_j, x_{jt} = 0$$

$$\forall j \in N, \sum_{t=0}^{H} x_{jt} = 1$$

$$\forall t \in [0, H[, \forall j \in N, x_{jt} \in \{0, 1\}$$

We now introduce the function δ and variables α_{jt}:

$$\begin{aligned}\delta(t_1, t_2, j) &= \begin{cases} \lambda_{t_2} & \text{if } t_1 \in [t_2 - p_j + 1, t_2[\\ 0 & \text{otherwise} \end{cases} \\ \alpha_{jt} &= w_j * (f_j(t + p_j) + \sum_{t'=0}^{H} \delta(t, t', j)\end{aligned}$$

Then, the cost of a solution becomes:

$$L(x, \lambda) = \sum_{t=0}^{H} \alpha_{jt} x_{jt} - m \sum_{t=0}^{H} \lambda_t$$

We can remark that $m \sum_{t=0}^{H} \lambda_t$ is not function of x_{jt}, then it is a constant term in the objective function. So removing $m \sum_{t=0}^{H} \lambda_t$ from the objective function does not change the optimal solution. In addition, the variables x_{jt} are only mutually independent according to the parameter j. Then, finding the optimal solution is equivalent to minimizing separately $\sum_{t=0}^{H} \alpha_{jt} x_{jt}$ for $j \in N$. Furthermore, only one x_{jt} for a fixed value of j can be non null because of the constraint $\forall j \in N, \sum_{t=0}^{H} x_{jt} = 1$. Last, minimizing $\sum_{t=0}^{H} \alpha_{jt} x_{jt}$ is made by choosing $t = t^*$ such that $\alpha_{jt^*} = \min_{t' \in [r_j, H]} \alpha_{jt'}$.

Then, we use a standard sub-gradient (see [6]) to get iterative values of this lower bound. From now on we shall refer to this lower bound as $lb_{t-lag-mach}$ (*Lagrangian resource relaxation of* lb_t).

Proposition 5.1. *Lower bounds* lb_{t-lp} *and* $lb_{t-lag-mach}$ *are equal.*

Proof. We have remarked that the optimal value of the Lagrangian relaxation is obtained by choosing for each value of j the value t^* of t such that $\alpha_{jt^*} = \min_{t' \in [r_j, H]} \alpha_{jt'}$. Hence the integrity constraint in the Lagrangian relaxation is useless. Moreover, the Lagrangian relaxation of a constraint of a linear program does not change the value of the optimum. Hence the two lower bounds are equal. □

5.3 Lagrangian Relaxation on the Number of Occurrences Constraint

In this section, we follow the idea of Rivreau and we relax the constraint (9) stating that a job has to be executed exactly once (see [5]). The relaxation is dualized to obtain another Lagrangian relaxation. The formulation becomes:

$$\min_{j \in N, t \in [0, H[} \sum_{j=1}^{n} \sum_{t=0}^{H} x_{jt} f_j(t + p_j) + \sum_{j=1}^{n} \lambda_j (1 - \sum_{t=0}^{H} x_{jt}) \tag{11}$$

$$\forall j \in N, \forall t < r_j, x_{jt} = 0 \tag{12}$$

$$\forall t \in [0, H[, \sum_{j=1}^{n} \sum_{t'=t-p_j+1}^{t} x_{jt'} \leq m \tag{13}$$

$$\forall t \in [0, H[, \forall j \in N, x_{jt} \in \{0, 1\} \tag{14}$$

In this problem, we have to find the optimal schedule on m parallel machines, where jobs are allowed to be "unprocessed" or to be processed several times. The cost of any feasible solution is composed by two parts: the sum of the costs of all jobs $\sum_{j=1}^{n}\sum_{t=0}^{H} x_{jt}(f_j(t+p_j)-\lambda_j)$, and a constant value $\sum_{j=1}^{n}\lambda_j$. As scheduling no jobs is a feasible solution, we can say that the minimal sum of all costs is negative. In addition, we can remark that all the machines must be equivalent according to their costs in the optimal solution. Otherwise, it would be possible to build a better solution by duplicating m times the better machine in this schedule. It means that we only have to find an optimal schedule on a single machine. We denote by $\Lambda(t)$ the minimal cost between time t and H on a single machine. We have $\Lambda(H)=0$. Then, we use the following dynamic relation:

$$\Lambda(t) = \min(\Lambda(t+1), \min_{j\in[1,n] \text{ with } r_j \geq t}(f_j(t+p_j)-\lambda_j+\Lambda(t+p_j)))$$

The optimal cost is $\lambda^* = m\Lambda(0)+\sum_{j=1}^{n}\lambda_j$. From now, on we shall refer to this lower bound as $lb_{t-lag-occ}$ (*Lagrangian relaxation of the number of occurrences of* lb_t).

5.3.1 Baning Local Repetitions

To improve this lower bound, we can add a constraint stating that a job can not be processed twice consecutively. This notion is a particular case of the $P(\lambda)-path$ defined by Rivreau [5]. To find the optimal solution without local repetition, we have to use the following definitions:

- $\Lambda_1(t)$: The minimal cost of a schedule between time t and time H without local repetition.
- $J(t)$: The job which is executed in first position in the schedule that realizes the value of $\Lambda_1(t)$.
- $\Lambda_2(t)$: The minimal cost of a schedule between time t and time H without repetition, and not beginning with job $J(t)$.

Then, we can compute the optimal schedule without repetition according to the following scheme [5]:

First, we need to introduce the following notations:

$$\Lambda_b(j,t) = \begin{cases} \Lambda_1(t) & \text{if } j \neq J(t) \\ \Lambda_2(t) & \text{otherwise} \end{cases}$$

$$J_b(j,t) = \underset{\substack{j\in[1,n]\\ r_j\geq t}}{\arg\min}\,(f_j(t+p_j)-\lambda_j+\Lambda_b(j,t+p_j))$$

Hence, it follows:

$$\Lambda_1(t) = \min \begin{cases} \Lambda_1(t+1) \\ \min\limits_{\substack{j \in [1,n] \\ r_j \geq t}} (f_j(t+p_j) - \lambda_j + \Lambda_b(j, t+p_j)) \end{cases}$$

Now, we have:

$$J(t) = \begin{cases} J(t+1) & \text{if } \Lambda_1(t) = \Lambda_1(t+1) \\ J_b(t) & \text{otherwise} \end{cases}$$

Finally, we compute $\Lambda_2(t)$ as follows:

$$\Lambda_2(t) = \min \begin{cases} \Lambda_b(J(t), t+1) \\ \min\limits_{\substack{j \in [1,n] \\ r_j \geq t \\ j \neq J(t)}} f_j(t+p_j) - \lambda_j + \Lambda_b(j, t+p_j) \end{cases}$$

Using this recursive scheme, value $\Lambda_1(0)$, which is a lower bound, is computed. From now on we shall refer to this lower bound as $lb_{t-lag-ltd-occ}$ (*Lagrangian relaxation of the number of occurrences of* lb_t *without local repetitions*).

6 Experimental Results

In this section, we provide experimental results to compare efficiency of all lower bounds. There is a part for each criterion. All the computations have been done on a Pentium-M 1,6GHz running MS-Windows XP. We have used instances generated from schemes of the literature.

For the total completion time and the total weighted completion time, schemes of Hariri and Potts [7] and Belouadah, Posner and Potts [3] have been adapted. This scheme depends on one parameter R. When n, m and R are fixed, five instances are generated: values p_i, r_i and w_i follow uniform distributions. Values p_i are distributed in $[1, 100]$, w_i in $[1, 10]$ and r_i in $[0, 50.5nR/m]$. Parameter R takes ten values $\{0.2, 0.4, 0.6, 1.0, 1.25, 1.5, 1.75, 2.0, 3.0\}$. Finally, there are 50 instance for each couple (n, m).

For the total tardiness and the total weighted tardiness, we have adapted the schemes from Chu [4] and Akturk and Ozdemir [1] for the single machine problem. For each n, m, we generate m sets of $\lfloor \frac{n}{m} \rfloor$ jobs and one set of $n - m * \lfloor \frac{n}{m} \rfloor$ jobs are done to build an instance. This method is controlled by two parameters α and β. Values r_i, p_i, d_i and w_i (for the weighted case) are generated from uniform distributions: p_i are uniformly distributed in $[1, 100]$, w_i in $[1, 10]$. Then, r_i are distributed in $[0, \alpha \sum p_i]$ and $d_i - r_i + p_i$ in $[0, \beta \sum p_i]$. Parameter α takes values $\{0, 0.5, 1, 1.5\}$ and β takes values in $\{0.05, 0.25, 0.5\}$. When n, m, α and β are fixed, 10 instances are created. Finally, there are 120 instances for each couple (n, m).

Tests have been made on instances of 20, 50 and 100 jobs. For each size, there are five sets of instances depending on the number of machines: 2,3,4,5 or 10. For each criterion and for each instance, we have computed all lower bounds. Then, the distance between each lower bound and the best one is computed. In each table, we report the average of these distances "dist" (in percents) grouped by number of jobs. Average computational times "cpu" (in seconds) are also provided.

Some preliminary tests have been made to guess "good" values of the parameters for the flow based lower bounds. When the number of intervals is allowed to be pseudo-polynomial, our tests lead us to choose $a = 15$ (see Section 3.1).

6.1 $Pm|r_i|\sum C_i$

All results for the total completion time are presented in Table 8. One can see that lower bound $lb_{t-lag-mach}$ provides the best results. However, the cpu time needed is quite large. Improved flow based lower bounds are also efficient and faster than $lb_{t-lag-mach}$. Among minimal completion times based lower bounds, we can observe the dominance (theoretically proved) of lower bound $lb_{[\ ]mip}$. However, lower bound $lb_{[\ ]combo}$ seems to be, among all lower bounds, the best trade-off between quality and speed.

6.2 $Pm|r_i|\sum w_iC_i$

Results for total weighted completion time are shown on Table 9. Lower bounds lb_{t-lp} and $lb_{t-lag-mach}$ provide the best results. However, the cpu time needed is very important. Minimal completion times based lower bounds are far from the best lower bound. The same consideration can be done about lower bounds based on an occurrence constraint relaxation of the time indexed formulation ($lb_{t-lag-occ}$ and $lb_{t-lag-ltd-occ}$). Last, we can say that all flow based lower bounds provide very good results during a quite reasonable amount of time.

6.3 $Pm|r_i|\sum T_i$

Results for the total tardiness problem are provided on Table 10. One can see that the continuous relaxation of the time indexed formulation (lb_{t-lp}) is really better than all others lower bounds. Naturally, the cpu time involved is really important. On instances of 100 jobs, as lower bound lb_{t-lp} has not been ran, we can compare others lower bounds. It seems that the resource relaxation lower bound $lb_{t-lag-mach}$ obtains the best results. But flow based lower bounds may be a good trade-off between quality and speed.

	n=20		n=50		n=100	
lower bound	**dist**	**cpu**	**dist**	**cpu**	**dist**	**cpu**
$lb_{no-release}$	37.4	0.00	38.7	0.00	39.5	0.00
$lb_{no-release-subsets}$	15.0	0.00	18.4	0.00	19.6	0.00
lb_{flow}	14.2	0.00	11.7	0.07	10.3	0.81
$lb_{flow'}$	3.6	0.01	1.7	0.42	0.5	5.14
$lb_{flow''}(15)$	3.7	0.02	1.9	0.46	0.5	5.41
$lb_{[\ ]r+p}$	10.3	0.00	13.4	0.00	14.2	0.00
$lb_{[\ ]spt}$	46.1	0.00	44.2	0.00	42.6	0.00
$lb_{[\ ]split}$	18.0	0.00	9.0	0.00	4.5	0.00
$lb_{[\ ]mip}$	4.2	0.06	2.8	0.32	1.3	1.81
$lb_{[\ ]lp}$	5.0	0.07	4.2	0.32	3.2	1.78
$lb_{[\ ]resort}$	5.3	0.00	5.3	0.00	5.2	0.00
$lb_{[\ ]combo}$	4.8	0.00	3.6	0.00	2.0	0.00
lb_{t-lp}	0.0	2.79	6.0	73.71	-	-
$lb_{t-lag-mach}$	0.8	1.17	0.9	11.55	0.5	54.49
$lb_{t-lag-occ}$	18.9	0.05	18.0	0.31	17.0	1.71
$lb_{t-lag-ltd-occ}$	22.1	0.05	20.5	0.44	18.7	2.35

Table 8: Results for Total Completion Time

	n=20		n=50		n=100	
lower bound	**dist**	**cpu**	**dist**	**cpu**	**dist**	**cpu**
lb_{flow}	12.5	0.00	9.7	0.08	8.2	0.95
$lb_{flow'}$	3.4	0.02	1.8	0.50	0.5	6.61
$lb_{flow''}(15)$	3.5	0.02	1.9	0.51	0.6	6.98
$lb_{[\ ]r+p}$	28.4	0.00	32.9	0.00	35.2	0.00
$lb_{[\ ]spt}$	62.0	0.00	64.7	0.00	64.2	0.00
$lb_{[\ ]split}$	38.6	0.00	33.3	0.00	30.5	0.00
$lb_{[\ ]mip}$	25.0	0.05	26.8	0.31	27.1	1.98
$lb_{[\ ]lp}$	25.7	0.06	28.4	0.33	29.3	1.80
$lb_{[\ ]resort}$	26.0	0.00	29.4	0.00	31.1	0.00
$lb_{[\ ]combo}$	25.6	0.00	27.6	0.00	27.8	0.00
lb_{t-lp}	0.0	2.40	0.0	49.63	-	-
$lb_{t-lag-mach}$	0.8	1.53	0.9	13.01	0.3	58.48
$lb_{t-lag-occ}$	30.3	0.05	30.8	0.43	30.8	2.12
$lb_{t-lag-ltd-occ}$	32.7	0.06	32.4	0.53	32.1	2.88

Table 9: Results for Total Weighted Completion Time

	n=20		n=50		n=100	
lower bound	**dist**	**cpu**	**dist**	**cpu**	**dist**	**cpu**
$lb_{no-release}$	51.29	0.00	48.07	0.00	41.09	0.00
$lb_{no-release-subsets}$	50.56	0.00	47.34	0.00	40.46	0.00
lb_{flow}	58.74	0.01	50.29	0.17	34.20	1.57
$lb_{flow'}$	48.04	0.02	32.53	0.53	12.09	6.10
$lb_{flow''}(15)$	48.50	0.02	33.10	0.49	12.73	5.79
$lb_{[\ ]r+p}$	68.50	0.00	69.17	0.00	64.67	0.00
$lb_{[\ ]spt}$	51.47	0.00	42.21	0.00	31.70	0.00
$lb_{[\ ]split}$	49.63	0.00	39.14	0.00	28.33	0.00
$lb_{[\ ]mip}$	47.13	0.07	36.66	0.51	25.58	3.66
$lb_{[\ ]lp}$	47.75	0.07	38.99	0.35	29.22	2.00
$lb_{[\ ]resort}$	49.53	0.00	41.00	0.00	32.36	0.00
$lb_{[\ ]combo}$	48.62	0.00	38.47	0.00	27.98	0.00
lb_{t-lp}	1.18	0.00	0.40	0.09	-	-
$lb_{t-lag-mach}$	17.92	0.73	14.71	9.72	6.37	43.07
$lb_{t-lag-occ}$	42.48	0.03	46.02	0.35	46.78	1.69
$lb_{t-lag-ltd-occ}$	41.01	0.06	41.04	0.63	38.58	3.76

Table 10: Results for Total Tardiness

6.4 $Pm|r_i|\sum w_iT_i$

Results for the total weighted tardiness are presented on Table 11. Behaviours of lower bounds are closed to those obtained for the total tardiness problem. In particular, lower bound lb_{t-lp} provides the best results, with a larger gap than others lower bounds. The cpu time is very important. The resource relaxation lower bound is not very bad and its cpu time is reasonable. Minimal completion times based lower bounds do not provide interesting results. Finally, flow based lower bounds seem to be interesting when lb_{t-lp} can not be used.

	n=20		**n=50**		**n=100**	
lower bound	**dist**	**cpu**	**dist**	**cpu**	**dist**	**cpu**
lb_{flow}	56.04	0.01	48.13	0.16	31.38	1.42
$lb_{flow'}$	45.87	0.02	32.39	0.55	11.45	6.90
$lb_{flow''}(15)$	46.42	0.02	33.11	0.51	12.37	6.88
$lb_{[\ ]r+p}$	66.17	0.00	69.00	0.00	65.33	0.00
$lb_{[\ ]spt}$	64.54	0.00	65.17	0.00	59.69	0.00
$lb_{[\ ]split}$	64.69	0.00	65.20	0.00	59.66	0.00
$lb_{[\ ]mip}$	64.49	0.08	65.10	0.51	59.64	3.64
$lb_{[\ ]lp}$	64.50	0.08	65.12	0.33	59.64	1.78
$lb_{[\ ]resort}$	64.50	0.00	65.12	0.00	59.64	0.00
$lb_{[\ ]combo}$	64.50	0.00	65.12	0.00	59.64	0.00
lb_{t-lp}	0.03	2.51	0.00	50.89	-	-
$lb_{t-lag-mach}$	16.78	0.71	16.39	4.81	6.89	19.30
$lb_{t-lag-occ}$	43.19	0.03	50.44	0.28	52.35	1.34
$lb_{t-lag-ltd-occ}$	42.27	0.05	46.79	0.47	45.24	2.71

Table 11: Results for Total Weighted Tardiness

7 Conclusion

In this paper, we have presented a large overview of lower bounds for the $Pm|r_i|\sum C_i$, $Pm|r_i|\sum w_iC_i$, $Pm|r_i|\sum T_i$ and $Pm|r_i|\sum w_iT_i$ scheduling problems. In particular, we have proposed several lower bounds based on different principles. Some of these lower bounds give very good results for the four studied problems. Experimental results are given to compare all lower bounds. Finally, we have tried to analyse the behaviour of the lower bounds depending on the criterion involved.

The lower bound lb_{t-lp} is very often the best one. We can also remark that minimal completion times based lower bounds are efficient for the total completion time. For other criteria, it seems to be

a bit less efficient. For instance in the weighted case, parameters of the problem (typically the weights) are reassigned in such a way that the obtained value is minimal. This comes from the fact that we want to ensure that we obtain a lower bound.

Flow based lower bounds provide better results for total (weighted) completion time than for total (weighted) tardiness. This comes from the fact that for total weighted completion time, the loss associated to the relaxation is partially balanced by some constant value (see Proposition 3.3).

Finally, note that release dates relaxation and occurrence constraint relaxation based lower bounds seem to be often dominated.

A Notations

- $lb_{no-release}$ as *Release dates are relaxed to the smallest one.*
- $lb_{no-release-subsets}$ as *Decomposition of the problem into a fixed number of problems without release dates.*
- lb_{flow} as *Reduction to a flow Problem.*
- $lb_{[\ ]}$ as *Minimal completion times based lower bounds.*
  - $lb_{[\ ]r+p}$ as *Minimal completion times based on* $r_i + p_i$.
  - $lb_{[\ ]spt}$ as *Minimal completion times based on McNaughton's rule.*
  - $lb_{[\ ]split}$ as *Minimal completion times based on job splitting.*
  - $lb_{[\ ]mip}$ as *Minimal completion times based on a MIP formulation relying on Horn's Theorem* .
  - $lb_{[\ ]lp}$ as *Continuous relaxation of* $lb_{[\ ]mip}$.
  - $lb_{[\ ]resort}$ as *A relaxation of* $lb_{[\ ]mip}$ *based on "resorting costs".*
  - $lb_{[\ ]combo}$ as *Combination of* $lb_{[\ ]resort}$ *and* $lb_{[\ ]split}$.
- lb_t as *Time indexed formulation.*
  - lb_{t-lp} as *Continuous relaxation of* lb_t.
  - $lb_{t-lag-mach}$ as *Lagrangian resource relaxation of* lb_t.
  - $lb_{t-lag-occ}$ as *Lagrangian relaxation of the number of occurrences of* lb_t.
  - $lb_{t-lag-ltd-occ}$ as *Lagrangian relaxation of the number of occurrences of* lb_t *without local repetitions.*

References

[1] M.S. Akturk and D. Ozdemir. An exact approach to minimizing total weighted tardiness with release dates. *IIE Transactions*, 32:1091–1101, 2000.

[2] M.A. Aloulou, M.Y. Kovalyov, and M.-C. Portmann. Evaluating flexible solutions in single machine scheduling via objective function maximization: the study of computational complexity. *RAIRO - Operations Research*, to appear.

[3] H. Belouadah, M.E. Posner, and Potts C.N. Scheduling with release dates on a single machine to minimize total weighted completion time. *Discrete Applied Mathematics*, 36:213–231, 1992.

[4] C. Chu. A branch and bound algorithm to minimize total flow time with unequal release dates. *Naval Research Logistics*, 39:859–875, 1992.

[5] Rivreau D. *Problèmes d'Ordonnancement Disjonctifs : Règles d'Elimination et Bornes Inférieures.* PhD thesis, Université de Technologie de Compiègne, France, 1999.

[6] M. Gondran and M. Minoux. *Graphes et Algorithmes.* Eyrolles, third edition, 1995.

[7] A. Hariri and C. Potts. An algorithm for single machine sequencing with release dates to minimize total weighted completion time. *Discrete Applied Mathematics*, 5:99–109, 1983.

[8] W.A. Horn. Minimizing average flow time with parallel machines. *Operations Research*, 21:846–847, 1973.

[9] W.A. Horn. Some simple scheduling algorithms. *Naval Research Logistics Quaterly*, 21:177–185, 1974.

[10] A. Jouglet. *Ordonnancer une machine pour minimiser la somme des coûts.* PhD thesis, Université de Technologie de Compiègne, France, 2002.

[11] A. Jouglet, Ph. Baptiste, and J. Carlier. *Handbook of scheduling : algorithms, models, and performance analysis.*, chapter Branch-and-Bound Algorithms for Total Weighted Tardiness. Leung, J. Y-T, Chapman & Hall / CRC edition, 2004.

[12] S. Kedad-Sidhoum, Y. Rios Solis, and F. Sourd. Lower bounds for the earliness-tardiness scheduling problem on single and parallel machines. *European Journal of Operational Research*, to appear.

[13] J. Lenstra, A R. Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.

[14] R. McNaughton. Scheduling with deadlines and loss functions. *Management Sciences*, 6:1–12, 1959.

[15] J. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41:338–350, 1993.

[16] F. Yalaoui and C. Chu. A new exact method to solve the $Pm|r_i|\sum C_i$ problem. *International Journal of Production Economics*, In press, 2005.