Lower Bounds for Scheduling on Identical Parallel Machines with Heads and Tails

Mohamed Haouari and Anis Gharbi

Laboratory of Mathematical Engineering

Ecole Polytechnique de Tunisie, BP 743, 2078, La Marsa. Tunisia.

Abstract

In this paper, we investigate new lower bounds for the $P=r_j;q_j=C_{max}$ scheduling problem. A new bin packing based lower bound, as well as several new lifting procedures are derived for this strongly NP-hard problem. Extensive numerical experiments show that the proposed lower bounds consistently outperform the best existing ones.

Keywords : Scheduling, Identical Parallel Machines, Release Dates, Delivery Times, Makespan, Lower Bound.

1 Introduction

In this paper, we investigate lower bounding strategies for the problem of scheduling identical parallel machines with release dates and delivery times. This problem is formally described as follows: A set J of n jobs has to be scheduled on m identical parallel machines (with $n > m \geq 2$). Each job j has a release date or head r_j; a processing time or body p_j and a delivery time or tail q_j: All data are assumed to be deterministic and integer. Each machine processes at most one job at one time and each job cannot be processed by more than one machine at one time. We assume that preemption is not allowed and that all machines are available from time zero onwards. Let $t_j \geq r_j$ denote the start time of job j, then the completion time of j is defined as $C_j = t_j + p_j + q_j$: The objective is to find a schedule that minimizes the maximum completion time (or makespan) $C_{max} = \max_{j \in J} C_j$. Following Lawler et al. [11], this problem is denoted by $P=r_j;q_j=C_{max}$.

The $P=r_j;q_j=C_{max}$ is NP-hard in the strong sense since it generalizes the well studied $P=\!=C_{max}$ which is strongly NP-hard (Garey and Johnson [7]). However, unlike this latter problem for which there exist very efficient exact algorithms (Dell'Amico and Martello [6]),

computational experiments with optimization algorithms developed by Carlier [3] and Gharbi and Haouari [8] provide evidence that only small or medium-sized instances could be solved within a reasonable computing time. The objective of this paper is to derive several new lower bounds for the $P=r_j;q_j=C_{max}$: These bounds might be useful both to improve the e¤ectiveness of branch and bound algorithms and to benchmark heuristic solutions.

Our motivation for the study of the $P=r_j;q_j=C_{max}$ stems from the fact that it constitutes a fundamental model encompassing several well studied parallel machine scheduling problems, including $P==C_{max}$, $P=r_j=C_{max}$; $P==L_{max}$; and the parallel machine problem with machine availabilities denoted by $P;NC_{inc}==C_{max}$. Hence, many of the results presented in this paper might be applicable for all these problems. Also, the $P=r_j;q_j=C_{max}$ has many interesting theoretical applications. In particular, it arises as a strong relaxation of the Multiprocessor Flow Shop problem (Hoogeveen et al. [9]) and it plays a central role in some exact algorithms for the Resource Constrained Project Scheduling Problem (Carlier and Latapie [4]).

The paper is organized as follows: In Section 2, we survey lower bounds from the literature. In Section 3, a new lower bound based on the bin packing problem is proposed. In Sections 4-6 several strategies aimed at enhancing lower bounds are introduced. These strategies are based on considering subset of jobs, machines, or both. In Section 7, we present the results of an extensive computational analysis of the di¤erent lower bounds. Finally, we conclude by providing a summary of our results and indicating some directions for future research.

For convenience, we assume hereafter that the release dates, processing times, and delivery times are sorted according to the non-decreasing order. The values $r_{[j]}(J)$; $p_{[j]}(J)$ and $q_{[j]}(J)$ denote the j^{th} smallest release date, processing time and delivery time in J; respectively. We denote by $\lceil x \rceil$ the smallest integer that is larger than or equal to x; and by $\lfloor x \rfloor$ the largest integer that is smaller than or equal to x:

2 Lower bounds from the literature

2.1 Simple lower bounds

Obviously, no job j can …nish its processing earlier than $r_j + p_j + q_j$: Thus, a simple lower bound is:

$$LB_0(J) = \max_{j \in J} (r_j + p_j + q_j)$$

LB_0 can be computed in O(n) time. A second simple O(n) lower bound given by Carlier [3] is:

$$LB_1(J) = \min_{j \in J} r_j + \left\lceil \frac{1}{m} \sum_{j \in J} p_j \right\rceil + \min_{j \in J} q_j$$

This bound is based on the fact that all release dates and delivery times are assumed to be equal to $\min_{j \in J} r_j$ and $\min_{j \in J} q_j$; respectively.

2.2 A lower bound based on machine idle times

In any feasible schedule, a machine is necessary idle from time zero to $r_1(J)$; a second one from time zero to $r_2(J)$, ..., and an m^{th} one from time zero to $r_m(J)$. Similarly, a machine is idle from time $C^*_{max}(J) - q_1(J)$ to $C^*_{max}(J)$, ..., and an m^{th} one from time $C^*_{max}(J) - q_m(J)$ to $C^*_{max}(J)$. Moreover, the total amount of activity of the m machines is equal to $\sum_{j \in J} p_j$. By adding the minimal amounts of activity to those of inactivity, we obtain:

$$r_1(J) + r_2(J) + \cdots + r_m(J) + \sum_{j \in J} p_j + q_1(J) + q_2(J) + \cdots + q_m(J) \leq mC^*_{max}(J)$$

This yields the following O(n) lower bound proposed by Carlier [3]:

$$LB_2(J) = \left\lceil \frac{1}{m} \left(r_1(J) + r_2(J) + \cdots + r_m(J) + \sum_{j \in J} p_j + q_1(J) + q_2(J) + \cdots + q_m(J) \right) \right\rceil$$

Clearly, $LB_2(J)$ dominates $LB_1(J)$:

2.3 A lower bound based on machine availability times

In a parallel machine scheduling problem with availability constraints; a machine M_i is available from time $u_i \geq 0$ onwards ($1 \leq i \leq m$). A strong lower bound for the $P; NC_{inc}==C_{max}$; denoted by the General Bound (GB), has been proposed by Webster [13]. Its description is rather long and may affect the clarity of the present paper, so we refer the reader to the original paper. Gharbi and Haouari [8] used GB to construct a new lower bound for the $P=r_j; q_j=C_{max}$, referred to as the Modified General Bound (MGB), that was proven to dominate LB_2: The MGB is computed as follows: First, a subset $J' \subseteq J$ is computed using an O(n log n) preprocessing algorithm. Let $D(J')$ be the set of dummy jobs $j_1; j_2; \ldots; j_{m-1}$ such that:

$$\begin{cases} r_{j_h} = r_m(J') \\ p_{j_h} = q_{h+1}(J') - q_1(J') & \text{for } h = 1; \ldots; m-1 \\ q_{j_h} = q_1(J') \end{cases}$$

One can prove that adding the set $D(J')$ to the jobset J' keeps the value of the optimal makespan unchanged. Hence, the "extended" $P=r_j; q_j=C_{max}$ instance defined on $J' \cup D(J')$ is relaxed as a parallel machine problem with machine availability times by assuming that

all release dates and delivery times are set equal to zero, and that each machine M_i has an availability time $u_i = r_i(J^1 [D(J^1))$ $(i = 1; :::; m)$: The bound MGB is de…ned by:

$$MGB(J) = GB(J^1 [D(J^1)) + \min_{j \in J^1} q_j$$

MGB is obtained in $O(n \log n)$ time. For convenience, in the sequel, we refer to MGB by LB_3:

2.4 The preemptive bound

If we relax the non-preemption constraint, then we obtain a preemptive parallel machine problem denoted by $P=r_j; q_j; pmtn=C_{max}$: Obviously, the makespan of an optimal preemptive schedule constitutes a strong lower bound for $P=r_j; q_j=C_{max}$: Horn [10] proposed a polynomial optimization algorithm for the $P=r_j; q_j; pmtn=C_{max}$. Basically, it consists in checking the existence of a preemptive schedule with makespan equal to a trial value C through a transformation into a maximum ‡ow problem. If LB and UB denote a lower and upper bound on the trial value C; then the optimal preemptive makespan is computed using a bisection search on the interval [LB; UB]: Its computation requires $O(n^3(\log n + \log p_{max}))$ time where p_{max} denotes the largest processing time. In the sequel, we de…ne the preemptive lower bound, denoted by PLB(J); as the optimal preemptive makespan:

2.5 The Jackson's pseudo-preemptive bound

In a pseudo-preemptive schedule, any operation is allowed to be preempted, and each machine can handle several jobs at one time. Moreover, each job can be processed by more than one machine at one time. The Jackson's Pseudo-Preemptive Schedule (JPPS) was introduced by Carlier and Pinson [5]. It was shown that it provides an $O(n \log n + nm \log m)$ lower bound for the $P=r_j; q_j=C_{max}$. The JPPS is a generalization of Jackson's preemptive schedule whose makespan is a tight lower bound for $1=r_j; q_j=C_{max}$: The Jackson's pseudo-preemptive lower bound, denoted hereafter by JPPB; is equal to $dC_{max}(JPPS)e$; where $C_{max}(JPPS)$ denotes the makespan of JPPS:

3 A bin packing based bound

A simple way to relax the $P=r_j; q_j=C_{max}$ is to assume that all the heads and tails are equal to $\min_{j \in J} r_j$ and $\min_{j \in J} q_j$; respectively. Let $C^{\alpha}_{max}(J)$ denote the optimal makespan of the $P=r_j; q_j=C_{max}$ de…ned on J: The following result clearly holds:

$$\min_{j \in J} r_j + LB_{P=r_j; q_j=C_{max}}(J) + \min_{j \in J} q_j \cdot C^{\alpha}_{max}(J)$$

where $LB_{P==C_{max}}(J)$ denotes a lower bound for the $P==C_{max}$ de…ned on J: The $P==C_{max}$ is a fundamental scheduling problem that has been extensively investigated by many researchers, and several lower bounds have been proposed in the literature (Dell'Amico and Martello [6]). These bounds include $\max_{j2J} p_j$; $\left\lceil \frac{1}{m} \sum_{j2J} p_j \right\rceil$, and $\beta_{n-m}(J) + \beta_{n-m+1}(J)$ to quote just a few. Therefore, any of these bounds can be used to derive a lower bound for $P=r_j; q_j=C_{max}$. For instance, taking $LB_{P==C_{max}}(J) = \left\lceil \frac{1}{m} \sum_{j2J} p_j \right\rceil$ yields $LB_1(J)$: However, a much more interesting lower bound can be derived from the optimal solution of a Bin Packing Problem (BPP): The BPP is an NP-hard optimization problem which can be described as follows: Given a set of weighted items and a set of bins, each item has to be assigned to one bin so that the total weight of the items in each bin does not exceed a given capacity C. The objective is to minimize the number of used bins (Martello and Toth [12]). The $P==C_{max}$ is related to the BPP in the following way. Given a trial value C of the optimal makespan, consider the decision problem which consists in checking whether the n jobs can be processed on the m machines such that the maximum completion time does not exceed C: A positive answer to this feasibility problem is provided if the minimal number of bins (machines) does not exceed m in the BPP de…ned for the n items (jobs) with weights p_j and bins' capacity C: Otherwise, we conclude that a valid lower bound for the $P==C_{max}$ is C + 1:

Dell'Amico and Martello [6] proposed a strong BPP based lower bound for the $P==C_{max}$ which can be described as follows: For each integer $p = \beta_1(J); \beta_2(J); ...; \min\left(\beta_{n-m+1}(J); \frac{C}{2}\right)$, the subsets J_1; J_2 and J_3 are de…ned by:

$$J_1 = \{j \in J;\ C - p < p_j\}$$
$$J_2 = \left\{j \in J;\ \frac{C}{2} < p_j \leq C - p\right\}$$
$$J_3 = \left\{j \in J;\ p \leq p_j \leq \frac{C}{2}\right\}$$

The two proposed lower bounds of the minimal number of used bins are:

$$BPP_1(C;p) = |J_1| + |J_2| + \max\left(0; \left\lceil \frac{\sum_{j2J_3} p_j - (C|J_2| - \sum_{j2J_2} p_j)}{C} \right\rceil\right)$$

$$BPP_2(C;p) = |J_1| + |J_2| + \max\left(0; \left\lceil \frac{|J_3| - \left\lfloor \frac{\sum_{j2J_2} (C - p_j)}{p} \right\rfloor}{\frac{C}{p}} \right\rceil\right)$$

Clearly, if $\max_p \{\max(BPP_1(C;p); BPP_2(C;p))\} > m$ then $C + 1$ is a valid lower bound for the $P==C_{max}$: Therefore, an enhanced lower bound for the $P==C_{max}$ can be computed through a bisection search on the interval $[L;U]$; where L and U denote a lower and upper bound on the optimal makespan, respectively. The lower bound is:

$$L = \max\left\{\max_{j\in J} p_j; \left\lceil \frac{1}{m}\sum_{j\in J} p_j \right\rceil; \beta_{n-m}(J) + \beta_{n-m+1}(J)\right\}$$

The upper bound U is the makespan of the schedule constructed according to the Longest Processing Time (LPT) rule. This consists in scheduling the available job with the longest processing time on the first available machine. The computation of L and U requires $O(n)$ and $O(n \log n)$ time, respectively. Let $BPP(J)$ denote the resulting lower bound. Then, $BPP(J)$ can be computed in $O(n^2 \log U)$.

Consequently, a valid lower bound for the $P|r_j;q_j|C_{max}$ is:

$$\min_{j\in J} r_j + BPP(J) + \min_{j\in J} q_j$$

This bound can be further improved in the following way: Consider the sets $D(J)$ and $D'(J)$ of dummy jobs defined by:

$$D(J) = \{j_h (h = 1;\ldots;m-1) | r_{j_h} = \tau_m(J); p_{j_h} = \bar{q}_{h+1}(J) - \bar{q}_1(J) \text{ and } q_{j_h} = \bar{q}_1(J)\}$$
$$D'(J) = \{j_h (h = 1;\ldots;m-1) | r_{j_h} = \tau_1(J); p_{j_h} = \tau_{h+1}(J) - \tau_1(J) \text{ and } q_{j_h} = \bar{q}_m(J)\}$$

Gharbi and Haouari [8] proved that adding the sets $D(J)$ and $D'(J)$ to the jobset J has no effect on the value of the optimal makespan. Let $J_{ext} = J \cup D(J) \cup D'(J)$ denote the extended set of jobs. Clearly, we have:

$$BPP(J) = BPP(J_{ext})$$

Therefore, an improved $O(n^2 \log U)$ lower bound for the $P|r_j;q_j|C_{max}$ is:

$$LB_4(J) = \min_{j\in J_{ext}} r_j + BPP(J_{ext}) + \min_{j\in J_{ext}} q_j$$

Obviously, any progress that would be made in the design of an improved lower bound for the bin packing problem could be advantageously used to improve LB_4:

Proposition 1:

$LB_4(J) \geq LB_2(J)$

Proof. Clearly, $LB_4(J) \geq LB_1(J_{ext})$: Note that

$$LB_1(J_{ext}) = \min_{j \in J_{ext}} r_j + \left\lceil \frac{1}{m} \sum_{j \in J_{ext}} p_j \right\rceil + \min_{j \in J_{ext}} q_j$$

$$= \min_{j \in J} r_j + \left\lceil \frac{1}{m} \left(\sum_{j \in D^0(J)} p_j + \sum_{j \in J} p_j + \sum_{j \in D(J)} p_j \right) \right\rceil + \min_{j \in J} q_j$$

$$= r_1(J) + \left\lceil \frac{1}{m} \left(\sum_{h=2} r_h(J) + \sum_{j \in J} p_j + \sum_{h=2} q_h(J) - (m-1)(r_1(J) + q_1(J)) \right) \right\rceil + q_1(J)$$

$$= \left\lceil \frac{1}{m} \left(\sum_{h=1} r_h(J) + \sum_{j \in J} p_j + \sum_{h=1} q_h(J) \right) \right\rceil$$

$$= LB_2(J) \blacksquare$$

Example 1: Consider the 10 job -2 machine instance defined by Table 1.

j	1	2	3	4	5	6	7	8	9	10
r_j	2	8	3	6	5	3	9	6	3	7
p_j	92	92	97	93	92	4	4	5	3	4
q_j	2	2	10	4	7	7	8	2	10	10

Table 1 - Data of the 10 job - 2 machine instance of example 1

We have $D(J) = \emptyset$ and $D^0(J) = \{j_1 : r_{j_1} = 2; p_{j_1} = 1; q_{j_1} = 2\}$: The lower and upper bounds computed for the corresponding $P||C_{max}$ defined on $J_{ext} = J \cup \{j_1\}$ are $L = \max\left\{97; \left\lceil \frac{487}{2} \right\rceil; 93 + 92\right\} = 244$ and $U = 277$; respectively.

Consider the trial value $C = 275$: For $p_j = 92$; we have $J_1 = J_2 = \emptyset$ and $J_3 = \{1; 2; 3; 4; 5\}$: Thus, $BPP_2(C; p) = \max\left\{0; \left\lfloor \frac{5}{\frac{275}{92}} \right\rfloor\right\} = 3 > m$: Consequently, $C + 1 = 276$ is a lower bound for the considered $P||C_{max}$: Therefore, a valid lower bound for the $P|r_j; q_j|C_{max}$ instance is $LB_4(J) = 2 + 276 + 2 = 280$ (Note that $LB_1(J) = 247$ and $LB_2(J) = 248$):

4 Job subset based lower bounds

In this section as well as in the two following ones, we introduce some lifting procedures that aims at enhancing a given lower bound. Perhaps the simplest example of a lifting procedure is based on the observation that a valid lower bound on the optimal makespan of an instance de…ned on a jobset J is still valid when de…ned on a restricted subset S of jobs. Therefore, we can lift a given lower bound by taking its maximal value over all of the subsets.

For instance, by applying the job based lifting procedure to LB_1; we obtain the following valid bound:

$$JLB_1(J) = \max_{S \mu J} LB_1(S)$$

Carlier [3] proved that JLB_1 can be computed in $O(n \log n)$ time. Indeed, JLB_1 is equal to the optimal makespan of the preemptive one machine problem $1=r_j; q_j; pmtn=C_{max}$, obtained by replacing the m machines by a single one and dividing all the processing times by m. This problem is solved using Jackson's preemptive algorithm (see Carlier [2]).

Also, it is worth noting that applying this lifting procedure to the bound LB_2 would yield, at best, the value of JPPB: Indeed, Carlier and Pinson [5] proved that:

$$JPPB(J) = \max\left\{ LB_0(J); \max_{S \mu J} LB_2(S) \right\}$$

Unfortunately, for several bounds it is not known if their maximal value over all of the subsets can be computed in polynomial time. In these cases, for a given lower bound LB requiring O(g(n)) time, we propose to use a simple $O(n^2 g(n))$ greedy algorithm for …nding a subset $S^¤$ which is an approximate solution of $JLB(J) = \max_{S \mu J} LB(S)$. This algorithm is referred to as the Approximate Subset Procedure (ASP). It consists in initializing $S^¤$ to J and iteratively removing a job j_0 from $S^¤$: The job j_0 that is chosen to be removed is the one that yields the largest improvement to $LB(S^¤)$: Clearly, if $jS^¤j = m + 1$; then removing a job from $S^¤$ yields a bound dominated by $LB_0(J)$: The algorithm stops when no improvement is possible or $jS^¤j = m + 1$. Formally, ASP can be described as follows:

Approximate Subset Procedure (ASP)

Step 0. Set $S^¤ = J$ and $JLB(J) = LB(J)$
Step 1. For all $j \in S^¤$ do
If $LB(S^¤ n fjg) > LB(S^¤)$ set $j_0 = j$ and $LB(S^¤) = LB(S^¤ n fjg)$:
Step 2. If $LB(S^¤) \cdot JLB(J)$ or $jS^¤j = m+1$ then Stop. Otherwise, set $JLB(J) = LB(S^¤)$; $S^¤ = S^¤ n fj_0g$ and go to step 1.

Gharbi and Haouari [8] used ASP to compute an approximate value of $\max_{S \mu J} LB_3(S)$: The resulting value, denoted hereafter by $JLB_3(J)$, requires $O(n^3 \log n)$ time.

Also, an approximation of $\max_{S \mu J} LB_4(S)$ can be obtained using ASP. The resulting $O(n^4 \log U)$ lower bound will be denoted by $JLB_4(J)$:

It is worth noting that, due to its approximate computation, the bound $JLB_3(J)$ does not dominate $\max_{S \mu J} LB_2(S)$: However, if we consider the subset $J^{\alpha} \mu J$ such that $LB_2(J^{\alpha}) = \max_{S \mu J} LB_2(S)$; then the bound $\max_{S \mu J} LB_2(S)$ is dominated by $JLB_3(J^{\alpha})$. The set J^{α} can be computed in $O(n \log n + nm \log m)$ time (Carlier and Pinson [5]). Thus, $JLB_3(J^{\alpha})$ requires $O(n^3 \log n)$ time. With no loss of generality, the lower bound $\max\{JLB_3(J), JLB_3(J^{\alpha})\}$ is denoted by $JLB_3(J)$: Similarly, Proposition 1 proves that $JLB_4(J^{\alpha}) \geq \max_{S \mu J} LB_2(S)$: In the sequel, the bound $JLB_4(J)$ denotes $\max\{JLB_4(J), JLB_4(J^{\alpha})\}$:

5 Machine subset based lower bounds

In this section, we introduce a second basic lifting procedure which consists in considering only a restricted subset of machines. First, we state a lemma provided, with no formal proof, by Blocher and Chand [1] and then we prove it.

Lemma 1: (Blocher and Chand [1])

In any feasible schedule of a parallel machine problem with n jobs and m machines such that $n = \left\lfloor \frac{n}{m} \right\rfloor m + \alpha$; there is at least a set of α machines ($1 \leq \alpha \leq m - 1$) which must process at least $\alpha \left\lfloor \frac{n}{m} \right\rfloor + 1$ jobs.

Proof. The result is shown by induction on α:

i) Firstly, we consider $\alpha = 1$: Assume that $n = \left\lfloor \frac{n}{m} \right\rfloor m + 1$ and that each machine processes at most $\left\lfloor \frac{n}{m} \right\rfloor$ jobs. Thus, the m machines process at most $m \left\lfloor \frac{n}{m} \right\rfloor < n$ jobs which is absurd. Therefore, there is at least one machine processing at least $\left\lfloor \frac{n}{m} \right\rfloor + 1$ jobs. Hence, the result holds for $\alpha = 1$:

ii) Now, assume that the result holds for a given α ($\alpha = 1, \ldots, m - 2$): Then, there is at least a set S of α machines processing $\alpha \left\lfloor \frac{n}{m} \right\rfloor + 1 + \beta$ jobs ($\beta \geq 0$). Assume that each machine

of the $m - \rho$ remaining ones processes at most $\left\lfloor \frac{n}{m} \right\rfloor - \alpha$ jobs. That is, an upper bound on the total number of jobs processed on the m machines is:

$$n' = \rho \left(\left\lfloor \frac{n}{m} \right\rfloor + 1 \right) + \alpha + (m - \rho) \left(\left\lfloor \frac{n}{m} \right\rfloor - \alpha \right) = m \left\lfloor \frac{n}{m} \right\rfloor + \rho + \alpha(1 - m + \rho)$$

which is strictly smaller than n. Therefore, there is at least one machine among the $m - \rho$ ones processing at least $\left\lfloor \frac{n}{m} \right\rfloor - \alpha + 1$ jobs. By adding this machine to the set S; we obtain a set of $\rho + 1$ machines processing at least $\rho \left(\left\lfloor \frac{n}{m} \right\rfloor + 1 \right) + \alpha + \left\lfloor \frac{n}{m} \right\rfloor - \alpha + 1 = (\rho + 1) \left(\left\lfloor \frac{n}{m} \right\rfloor + 1 \right)$ jobs. ■

In fact, we can go a step further and propose the following more general result.

Lemma 2:

In any feasible schedule of a parallel machine problem with n jobs and m machines, there is at least a set of k machines ($1 \leq k \leq m$) which must process at least $\gamma_k = k \left\lfloor \frac{n}{m} \right\rfloor + \min(k, \rho)$ jobs, where $\rho = n - \left\lfloor \frac{n}{m} \right\rfloor m$.

Proof. Two cases have to be considered:

i) If $k \leq \rho$: Using Lemma 1, the result holds if the number of jobs is equal to $\left\lfloor \frac{n}{m} \right\rfloor m + k$. Clearly, the result remains true if the number of jobs is increased to $\left\lfloor \frac{n}{m} \right\rfloor m + \rho$.

ii) If $k > \rho$: First, we check that the result holds for $k = \rho + 1$. Lemma 1 ensures that there is a set S of ρ machines processing $\rho \left(\left\lfloor \frac{n}{m} \right\rfloor + 1 \right) + \alpha$ jobs ($\alpha \geq 0$). Assume that each of the remainder $m - \rho$ machines processes at most $\left\lfloor \frac{n}{m} \right\rfloor - \alpha - 1$ jobs. That is the total number of jobs processed by the m machines is less than or equal to:

$$n' = \rho \left(\left\lfloor \frac{n}{m} \right\rfloor + 1 \right) + \alpha + (m - \rho) \left(\left\lfloor \frac{n}{m} \right\rfloor - \alpha - 1 \right) = m \left\lfloor \frac{n}{m} \right\rfloor + \rho + \alpha - (m - \rho)(\alpha + 1).$$

Clearly, $n' < n$ for $\alpha = 0$: Assume that $n' = n$ for $\alpha > 0$: That is, $\alpha - (m - \rho)(\alpha + 1) = 0$: So, $m = \frac{\alpha}{\alpha + 1} + \rho$ which is impossible since $\frac{\alpha}{\alpha + 1}$ is not integer for $\alpha > 0$: Therefore, $n' < n$ for all $\alpha \geq 0$ which is absurd. Consequently, there is one machine among the $m - \rho$ ones

processing at least $\left\lceil \frac{n}{m} \right\rceil - \alpha$ jobs. By adding this machine to the set S; we obtain a set of $\alpha + 1$ machines processing at least $\alpha \left\lceil \frac{n}{m} \right\rceil + 1 + \cdots + \left\lceil \frac{n}{m} \right\rceil - \alpha = (\alpha + 1) \left\lceil \frac{n}{m} \right\rceil + \alpha$ jobs. Using similar arguments, we can prove that if the result holds for a given k $(\alpha + 1 \leq k \leq m - 1)$; then there is a set of $k + 1$ machines processing at least $(k + 1) \left\lceil \frac{n}{m} \right\rceil + \alpha$ jobs. ■

An immediate consequence of Lemma 2 is that a relaxed instance can be obtained by assuming that there is a subset of k machines $(k = 1, \ldots, m)$ processing β_k jobs having the smallest release dates, processing times and delivery times. The latter $P|r_j; q_j|C_{max}$ instance will be denoted hereafter by P_k: Clearly, a whole family of lifted lower bounds is derived by considering each P_k $(k = 1, \ldots, m)$: Let $MLB_1^k(J)$; $MLB_2^k(J)$; $MLB_3^k(J)$ and $MLB_4^k(J)$ denote the respective values of LB_1; LB_2; LB_3 and LB_4 computed for the instance P_k: This yields the following valid lower bounds:

$$MLB_1(J) = \max_{1 \leq k \leq m} MLB_1^k(J)$$

$$MLB_2(J) = \max_{1 \leq k \leq m} MLB_2^k(J)$$

$$MLB_3(J) = \max_{1 \leq k \leq m} MLB_3^k(J)$$

$$MLB_4(J) = \max_{1 \leq k \leq m} MLB_4^k(J)$$

Clearly, the computation of MLB_1; MLB_2; MLB_3 and MLB_4 requires $O(mn)$; $O(mn)$; $O(mn \log n)$ and $O(mn^2 \log U)$ time, respectively.

The following example illustrates how LB_1 and LB_2 can be lifted using the machine subset lifting procedure.

Example 2: Consider the 10 job - 4 machine instance de…ned by Table 2.

j	1	2	3	4	5	6	7	8	9	10
r_j	9	6	2	9	8	2	2	5	7	8
p_j	95	96	90	99	97	91	97	92	97	93
q_j	9	4	8	8	8	6	7	5	1	2

Table 2 - Data of the 10 job - 4 machine instance of example 2

We have $LB_1(J) = 2 + \left\lceil \frac{947}{4} \right\rceil + 1 = 240$ and $LB_2(J) = \left\lceil \frac{11 + 947 + 12}{4} \right\rceil = 243$:

For $k = 1$; we have $\beta_k = 3$ and

$$MLB_1^k(J) = MLB_2^k(J) = 2 + (90 + 91 + 92) + 1 = 276$$

For k = 2; we have $\alpha_k = 6$ and

$$MLB_1^k(J) = 2 + \left\lceil \frac{90 + 91 + 92 + 93 + 95 + 96}{2} \right\rceil + 1 = 282$$

$$MLB_2^k(J) = \left\lfloor \frac{(2 + 2) + (90 + 91 + 92 + 93 + 95 + 96) + (1 + 2)}{2} \right\rfloor = 282$$

For k = 3; we have $\alpha_k = 8$ and

$$MLB_1^k(J) = 2 + \left\lceil \frac{90 + 91 + 92 + 93 + 95 + 96 + 97 + 97}{3} \right\rceil + 1 = 254$$

$$MLB_2^k(J) = \left\lfloor \frac{(2 + 2 + 2) + (90 + 91 + 92 + 93 + 95 + 96 + 97 + 97) + (1 + 2 + 4)}{3} \right\rfloor = 255$$

For k = 4; we have clearly $MLB_1^k(J) = LB_1(J) = 240$ and $MLB_2^k(J) = LB_2(J) = 243$:

Consequently, the resulting machine subset based bounds are:

$$MLB_1(J) = \max\{276; 282; 254; 240\} = 282$$

$$MLB_2(J) = \max\{276; 282; 255; 243\} = 282$$

6 Job-machine based lower bounds

The lower bounds derived with the machine based lifting procedure can be further improved if they are computed for every subset of J: In this section, we propose for each lower bound computed in the previous section, a speci…c method to compute (exactly or approximately) its maximal value over all the subsets of jobs.

Lemma 2 provides the minimal number of jobs α_k processed by at least one subset of k machines. We recall that $\alpha_k = k \left\lfloor \frac{n}{m} \right\rfloor + \min(k; \rho)$ jobs, where $\rho = n - \left\lfloor \frac{n}{m} \right\rfloor m$: Note that, for …xed k and m; di¤erent values of n may yield the same value of α_k. Given an integer α; let n_α denote the minimal number of jobs yielding $\alpha_k = \alpha$: Consider the subset J_α of the n_α jobs having the largest processing times. We make the following observation:

Observation 1:

1) $MLB_1(J_\alpha) \geq MLB_1(J)$
2) $MLB_2(J_\alpha) \geq MLB_2(J)$

Hence, we can systematically use the above observation to further lift the bounds MLB_1 and MLB_2. The number of jobs n_γ can be computed using the following proposition.

Proposition 2:

Let $\gamma_k(n) = k\left\lfloor \frac{n}{m} \right\rfloor + \min\{k, \rho(n)\}$, where $\rho(n) = n - \left\lfloor \frac{n}{m} \right\rfloor m$. Given the integers m, k and γ, the smallest number of jobs satisfying $\gamma_k(n) = \gamma$ is:

$$n_\gamma = \begin{cases} m\left(\frac{\gamma}{k} - 1\right) + k & \text{if } k \text{ divides } \gamma \\ (m-k)\frac{\gamma}{k} + \gamma & \text{otherwise} \end{cases}$$

Proof. It suffices to prove that $\gamma_k(n_\gamma) = \gamma$, and $\gamma_k(n) < \gamma$ for all $n < n_\gamma$. The result clearly holds if $k = m$. For $k = 1, \ldots, m-1$, two cases have to be considered:

i) If k divides γ: We have $\left\lfloor \frac{n_\gamma}{m} \right\rfloor = \frac{\gamma}{k} - 1 + \frac{k}{m} = \frac{\gamma}{k} - 1$. Thus, $\rho(n_\gamma) = n_\gamma - \left\lfloor \frac{n_\gamma}{m} \right\rfloor m = k$. That is, $\gamma_k(n_\gamma) = k\left(\frac{\gamma}{k} - 1\right) + k = \gamma$. Now, we prove that if $n < n_\gamma$ then $\gamma_k(n) < \gamma$. Clearly, $n < n_\gamma$ yields $\frac{n}{m} < \frac{\gamma}{k} - 1 + \frac{k}{m}$. That is, $\left\lfloor \frac{n}{m} \right\rfloor \cdot \frac{\gamma}{k} - 1$. The result clearly holds if $\rho(n) < k$ since $\gamma_k(n) = k\left\lfloor \frac{n}{m} \right\rfloor + \rho(n) \cdot k\left(\frac{\gamma}{k} - 1\right) + \rho(n) < \gamma$. If $\rho(n) \geq k$ then $\gamma_k(n) = k\left\lfloor \frac{n}{m} \right\rfloor + k \cdot \gamma$. Assume that $\gamma_k(n) = \gamma$. Then, $\left\lfloor \frac{n}{m} \right\rfloor = \frac{\gamma}{k} - 1$. Note that $n < m\left(\frac{\gamma}{k} - 1\right) + k$. That is, $n < m\left\lfloor \frac{n}{m} \right\rfloor + k$. Therefore $\rho(n) < k$ which is absurd. Consequently, $\gamma_k(n) < \gamma$.

ii) If k does not divide γ: We have $n_\gamma = m\left\lfloor \frac{\gamma}{k} \right\rfloor + \gamma - \left\lfloor \frac{\gamma}{k} \right\rfloor k$. Then, $\left\lfloor \frac{n_\gamma}{m} \right\rfloor = \left\lfloor \frac{\gamma}{k} \right\rfloor + \left\lfloor \frac{\gamma - \left\lfloor \frac{\gamma}{k} \right\rfloor k}{m} \right\rfloor$. But, $0 \leq \gamma - \left\lfloor \frac{\gamma}{k} \right\rfloor k < k$. That is, $0 \leq \left\lfloor \frac{\gamma - \left\lfloor \frac{\gamma}{k} \right\rfloor k}{m} \right\rfloor \leq \left\lfloor \frac{k}{m} \right\rfloor = 0$. Thus, $\left\lfloor \frac{n_\gamma}{m} \right\rfloor = \left\lfloor \frac{\gamma}{k} \right\rfloor$. Therefore, $\rho(n_\gamma) = n_\gamma - m\left\lfloor \frac{n_\gamma}{m} \right\rfloor = \gamma - \left\lfloor \frac{\gamma}{k} \right\rfloor k < k$. Consequently, we have $\gamma_k(n_\gamma) = k\left\lfloor \frac{\gamma}{k} \right\rfloor + \gamma - \left\lfloor \frac{\gamma}{k} \right\rfloor k = \gamma$.

Now, we prove that if $n < n_\gamma$ then $\gamma_k(n) < \gamma$: Clearly, $n < n_\gamma$ yields

$$n \cdot m \left\lfloor \frac{\gamma}{k} \right\rfloor + \gamma - 1 - k + 1.$$

That is, $\left\lfloor \frac{n}{m} \right\rfloor \cdot \left\lfloor \frac{\gamma}{k} \right\rfloor + \left\lfloor \frac{\gamma - k\left\lfloor \frac{\gamma}{k} \right\rfloor}{m} \right\rfloor - 1 = \left\lfloor \frac{\gamma}{k} \right\rfloor - 1$: Therefore,

$$\gamma_k(n) = k\left\lfloor \frac{n}{m} \right\rfloor + \min\{k, \rho(n)\} \cdot k\left(\left\lfloor \frac{\gamma}{k} \right\rfloor - 1\right) + k < \gamma. \blacksquare$$

We describe how to compute the lifted lower bound $JMLB_1(J) = \max_{S \subseteq J} MLB_1(S)$. It suffices to compute $JMLB_1^k(J) = \max_{S \subseteq J} MLB_1^k(S)$ for a given k $(1 \cdot k \cdot m)$. Let $f_k(J) = \left\lfloor \frac{1}{k} \sum_{j=1}^{n} p_j(J) \right\rfloor$: Clearly, if $\max_{S \subseteq J} f_k(S)$ can be computed in $O(g(n))$ time, then $JMLB_1^k(J)$ can be computed in $O(n^2 g(n))$ using the following exact algorithm:

Computation of $JMLB_1^k(J)$

Rank r_j and q_j in increasing order and set $JMLB_1^k(J) = 0$:
For all pairs of heads and tails (r, q) do
  Compute the jobset $S_{r,q} = \{j \in J : r_j \geq r \text{ and } q_j \geq q\}$
  Set $JMLB_1^k(J) = \max\left\{JMLB_1^k(J),\ r + \max_{S \subseteq S_{r,q}} f_k(S) + q\right\}$
End (for)

Now; we describe how to compute the subset $S^*_{r,q} \subseteq S_{r,q}$ such that $f_k(S^*_{r,q}) = \max_{S \subseteq S_{r,q}} f_k(S)$. For each integer γ; let $S_\gamma \subseteq S_{r,q}$ denote the job subset which maximizes $\left\lfloor \frac{1}{k} \sum_{j=1}^{n} p_j(S_{r,q}) \right\rfloor$: Clearly, the set $S^*_{r,q}$ is the set which satisfies $f_k(S^*_{r,q}) = \max f_k(S_\gamma)$: Using Observation 1, the set S_γ is the set of n_γ jobs with the largest processing times in $S_{r,q}$:

Obviously, for any subset in $S_{r,q}$; the value of γ cannot exceed $\gamma_{max} = k\left\lfloor \frac{|S_{r,q}|}{m} \right\rfloor + \min(k, \rho)$; where $\rho = |S_{r,q}| - m\left\lfloor \frac{|S_{r,q}|}{m} \right\rfloor$. On the other hand, assume that the set $S^*_{r,q}$ corresponds to a

value of γ which is less than or equal to k: In this case, $MLB_1^k(S'_{r,q})$ is clearly dominated by $LB_0(J)$. Thus, we set w.n.l.g. $JMLB_1^k(J) = \max\{JMLB_1^k(J); LB_0(J)\}$ and restricted our computation to $\gamma \in [\gamma_{min}; \gamma_{max}]$; where $\gamma_{min} = k + 1$:

If the jobs are sorted according to the non-increasing order of their processing times, then the computation of $f_k(S_\gamma)$ requires γ operations: Therefore, computing $f_k(S'_{r,q})$ requires $\left(\sum_{\gamma=\gamma_{min}}^{\gamma_{max}} \gamma\right)$ operations which is bounded by $O(n^2)$. Since $JMLB_1^k(J)$ is the maximal value of $MLB_1^k(S'_{r,q})$ over all pairs (r,q); then $JMLB_1^k(J)$ can be computed in $O(n^4)$ time. Therefore, $JMLB_1(J)$ requires $O(mn^4)$ time.

A second interesting job-machine based lower bound is $JMLB_2(J) = \max_{S \subseteq J} MLB_2(S)$: We propose to compute an approximation of $JMLB_2$: The procedure is similar in spirit to that of $JMLB_1$: That is, given the integer k $(1 \leq k \leq m)$; we compute for each integer $\gamma \in [\gamma_{min}; \gamma_{max}]$ the corresponding subset $S_\gamma \subseteq J$ of n_γ jobs maximizing the value of MLB_2^k; where $\gamma_{min} = k + 1$ and $\gamma_{max} = k\left\lfloor \frac{n}{m} \right\rfloor + \min(k; \lceil \frac{n}{2} \rceil)$.
The set S_γ is obtained using a greedy algorithm starting from the global set J: At each iteration, a particular job is removed such that the value of MLB_2^k is maximized. The algorithm stops when $|S_\gamma| = n_\gamma$: Let S_r; S_p and S_q denote the subsets of jobs with the k smallest release dates, γ smallest processing times, and k smallest delivery times, respectively. Let $\frac{1}{k}\pm_j(S_\gamma)$ denote the increase of MLB_2^k that is achieved when the job j is removed from the current set S_γ: We have:

$$\pm_j(S_\gamma) = \begin{cases} r_{k+1}(S_\gamma) - r_j & \text{for } j \in S_r \setminus (S_p \cup S_q) \\ p_{\gamma+1}(S_\gamma) - p_j & \text{for } j \in S_p \setminus (S_r \cup S_q) \ (p_{n+1} = 0) \\ q_{k+1}(S_\gamma) - q_j & \text{for } j \in S_q \setminus (S_r \cup S_p) \\ r_{k+1}(S_\gamma) - r_j + p_{\gamma+1}(S_\gamma) - p_j & \text{for } j \in (S_r \cap S_p) \setminus S_q \\ p_{\gamma+1}(S_\gamma) - p_j + q_{k+1}(S_\gamma) - q_j & \text{for } j \in (S_p \cap S_q) \setminus S_r \\ r_{k+1}(S_\gamma) - r_j + q_{k+1}(S_\gamma) - q_j & \text{for } j \in (S_r \cap S_q) \setminus S_p \\ r_{k+1}(S_\gamma) - r_j + p_{\gamma+1}(S_\gamma) - p_j + q_{k+1}(S_\gamma) - q_j & \text{for } j \in S_r \cap S_p \cap S_q \\ 0 & \text{otherwise} \end{cases}$$

It is worth noting that $MLB_2^1(J) = MLB_1^1(J)$: Thus, for k = 1; the bound $JMLB_2^k(J)$ can be computed exactly by using the above procedure devoted for $JMLB_1^k(J)$: Formally, the bound $JMLB_2$ can be computed using the following algorithm:

Computation of $JMLB_2(J)$

Set $JMLB_2(J) = JMLB_1^1(J)$:
For all $k = 2,\ldots,m$ do
  Set $\gamma = \gamma_{max}$ and $JMLB_2(J) = \max\{JMLB_2(J); MLB_2^k(J)\}$
  While $\gamma \geq \gamma_{min}$ do
    Compute n_γ and set $S_\gamma = J$
    While $|S_\gamma| > n_\gamma$ do
      Select the job $j_0 \in S_\gamma$ with maximal $\pi_j(S_\gamma)$
      Set $S_\gamma = S_\gamma \setminus \{j_0\}$ and update $\pi_j(S_\gamma)$ for all $j \in S_\gamma$
    End (while)
    Set $JMLB_2(J) = \max\{JMLB_2(J); MLB_2^k(S_\gamma)\}$
    Set $\gamma = \gamma - 1$
  End (while)
End (for)

If the release dates, processing times, and delivery times are sorted in the non-decreasing order, then computing $\pi_j(S_\gamma)$ for all $j \in S_\gamma$ requires $O(n)$ time. Thus, the set S_γ can be computed in $O(n^2)$: Since $JMLB_2(J) = \max_{1 \leq k \leq m} \left\{ \max_{\gamma_{min} \leq \gamma \leq \gamma_{max}} MLB_2^k(S_\gamma) \right\}$; then the computation of $JMLB_2(J)$ requires $O(mn^3)$ time.

The bounds $JMLB_3(J) = \max_{S \subseteq J} MLB_3(S)$ and $JMLB_4(J) = \max_{S \subseteq J} MLB_4(S)$ can be computed approximately using the algorithm ASP. Their computation requires $O(mn^3 \log n)$ and $O(mn^4 \log U)$, respectively.

Let J^* denote the subset of jobs such that $LB_2(J^*) = \max_{S \subseteq J} LB_2(S)$: In the sequel, we set:

$$JMLB_2(J) = \max\{JMLB_2(J); JMLB_2(J^*)\}$$

$$JMLB_3(J) = \max\{JMLB_3(J); JMLB_3(J^*)\}$$

$$JMLB_4(J) = \max\{JMLB_4(J); JMLB_4(J^*)\}$$

7 Computational experiments

In order to assess the quality of the different lower bounds, we have generated 6 different random problem sets. The test problems have been designed with the aim of providing a good picture of the relative performance of the different lower bounds. The heads and tails of all jobs are drawn from the discrete uniform distribution on [1,10]. Each problem set is characterized by a number $\alpha \in \{0; 20; 40; 60; 80; 100\}$ which specifies that the processing

times of ®% of the jobs of each instance of the set are drawn from the discrete uniform distribution on [90,100], and the processing times of the remaining jobs are drawn from the discrete uniform distribution on [1,5]. This type of processing times leads to di¤erent combinations of long/short processing times, and heads and tails. Each problem set contains 250 test problems corresponding to 10 instances for di¤erent combinations of n and m with $n \in \{20; 40; 60; 80; 100\}$ and $m \in \{2; 3; 5; 7; 9\}$: Therefore, 1500 test problems were generated. All the computational experiments were carried out on a PentiumIII 733 MHz Personal Computer with 64 MB RAM.

In Table 3, we report for each basic (i.e. non-lifted) lower bound the percentage of times it yields the maximal value over all of the bounds discussed in this paper (Max), and the CPU time (in sec.) required for processing all of the 1500 instances (Time). We observe that the highest percentages (60:13%) are obtained for PLB and JPPB: Besides, we found that these two bounds are equal in all the generated instances, which is consistent with the preliminary results of Carlier and Pinson [5]. Obviously, the high percentages indicate that these bounds outperform the remaining ones. However, we observe that LB_2; LB_3;and LB_4 perform quite well but require much less CPU time.

	LB_0	LB_1	LB_2	LB_3	PLB	JPPB	LB_4
Max	6.93	45.93	53.00	57.73	60.13	60.13	55.60
Time	0.10	0.10	0.05	1.66	15.80	11.79	0.22

Table 3 - Performance of the basic bounds

In Tables 4-6, we report the performance results of the lifted lower bounds. We see from Table 4 that the job subset lifting procedure improves the quality of all these bounds. The improvement is negligible for LB_1 but more signi…cant for LB_3 and LB_4: Not surprisingly, these improvements caused a high increase of the CPU time. Table 5 shows that the machine subset lifting procedure is more e¤ective, since the quality improvements are comparable but the required CPU time is much less. Interestingly, we observe that this lifting procedure made MLB_2; MLB_3, and MLB_4 to outperform the best existing lower bound (JPPB). Indeed, MLB_4 requires less than one second and yields for 63.53% of the instances the best value, whereas JPPB requires 11.79 seconds and is the best one for 60.13% of the instances. Table 6, provides evidence that combining the two lifting procedures yields the most e¤ective lower bounds. Hence, $JMLB_2$ reached the maximal value over all the bounds for almost all instances (95:66%), and even the lifted version of the (simple) lower bound LB_1 yields the remarkable percentage of 82.86% and requires just 5.15 seconds. We observe that although LB_3 and LB_4 dominate LB_2 and LB_1; the lifted bounds $JMLB_2$ and $JMLB_1$ outperform $JMLB_3$ and $JMLB_4$: The (relatively) disappointing behavior of these two latter bounds

could be explained by the fact that the optimal subsets are computed using a simple greedy algorithm.

	JLB_1	JLB_3	JLB_4
Max	46.46	61.86	63.60
Time	0.33	226.71	40.09

Table 4 - Performance of the job subset based bounds

	MLB_1	MLB_2	MLB_3	MLB_4
Max	52.13	61.00	65.73	63.53
Time	0.24	0.23	3.64	0.99

Table 5 - Performance of the machine subset based bounds

	$JMLB_1$	$JMLB_2$	$JMLB_3$	$JMLB_4$
Max	82.86	95.66	70.33	71.80
Time	5.15	266.68	471.97	141.72

Table 6 - Performance of the job-machine subset based bounds

A more accurate picture of the respective performance of the di¤erent lower bounds is provided in Tables 7-9, where the detailed results are reported with respect to ®; n; and m: We observe that our global analysis is generally consistent with the results displayed in these tables. However, we can emphasize the following points:

- The bound LB_4 is very competitive with the best existing (non-lifted) bounds for large number of jobs ($n \geq 60$), small number of machines ($m = 2; 3$), and in the presence of more jobs with large processing times ($® \geq 40$).
- The e¤ectiveness of the job subset lifting procedure decreases when there are more jobs with large processing times ($® \geq 60$) and when the number of jobs increases ($n \geq 60$). Though, this lifting procedure performs much better on instances with a large number of machines ($m \geq 7$):
- The machine subset lifting procedure is e¤ective when all the jobs have large processing times ($® = 100$). Moreover, we observe that large improvements are achieved for instances with large number of machines.

² The relative e¤ectiveness of $JMLB_2$ seems to be little sensitive to the variation of the parameters ®; n and m: On the contrary, we observe that the bounds $JMLB_3$ and $JMLB_4$ exhibit better performance for instances with homogeneous processing times (® = 0 or 100).

	® = 0	® = 20	® = 40	® = 60	® = 80	® = 100
LB_0	21.60	16.00	4.00	0.00	0.00	0.00
LB_1	61.20	43.60	41.60	45.60	42.00	41.60
LB_2	77.60	46.00	44.40	54.40	43.60	52.00
LB_3	87.60	54.80	48.40	56.40	46.80	52.40
PLB	100.00	62.00	48.40	54.80	43.60	52.00
JPPB	100.00	62.00	48.40	54.80	43.60	52.00
LB_4	77.60	51.60	49.20	56.00	46.80	52.40
JLB_1	63.60	43.60	42.00	46.00	42.00	41.60
JLB_3	100.00	64.80	50.80	56.40	46.80	52.40
JLB_4	100.00	69.60	56.00	56.80	46.80	52.40
MLB_1	61.20	43.60	41.60	45.60	42.00	78.80
MLB_2	77.60	46.00	44.40	54.40	43.60	100.00
MLB_3	87.60	54.80	48.40	56.80	46.80	100.00
MLB_4	77.60	51.60	49.20	56.00	46.80	100.00
$JMLB_1$	84.00	94.80	86.00	70.80	82.80	78.80
$JMLB_2$	100.00	86.00	88.00	100.00	100.00	100.00
$JMLB_3$	100.00	64.80	50.80	57.60	48.80	100.00
$JMLB_4$	100.00	69.60	56.00	57.20	48.00	100.00

Table 7 - Performance of the lower bounds with respect to ®

	n = 20	n = 40	n = 60	n = 80	n = 100
LB_0	23.66	10.00	1.00	0.00	0.00
LB_1	15.66	29.00	50.33	60.66	74.00
LB_2	28.00	36.00	59.66	65.00	76.33
LB_3	46.33	37.33	61.00	67.33	76.66
PLB	53.33	46.00	60.00	65.00	76.33
JPPB	53.33	46.00	60.00	65.00	76.33
LB_4	32.66	38.66	62.66	67.33	76.66
JLB_1	17.33	30.00	50.33	60.66	74.00
JLB_3	57.33	47.00	61.00	67.33	76.66
JLB_4	61.00	49.66	63.33	67.33	76.66
MLB_1	21.00	36.66	56.33	66.00	80.66
MLB_2	38.00	46.00	66.33	71.66	83.00
MLB_3	56.66	47.33	67.66	74.00	83.00
MLB_4	42.66	48.66	69.33	74.00	83.00
$JMLB_1$	68.00	72.66	85.66	91.33	96.66
$JMLB_2$	98.66	95.66	93.00	93.66	97.33
$JMLB_3$	70.00	57.00	67.66	74.00	83.00
$JMLB_4$	72.33	59.66	70.00	74.00	83.00

Table 8 - Performance of the lower bounds with respect to n

	m = 2	m = 3	m = 5	m = 7	m = 9
LB_0	0.00	0.33	6.66	10.00	17.66
LB_1	92.33	67.66	43.66	16.33	9.66
LB_2	99.00	75.00	52.66	20.00	18.33
LB_3	99.66	76.33	54.00	28.66	30.00
PLB	100.00	76.00	59.33	30.00	35.33
JPPB	100.00	76.00	59.33	30.00	35.33
LB_4	99.00	75.33	54.33	25.66	23.66
JLB_1	93.33	68.00	44.33	17.00	9.66
JLB_3	100.00	76.66	60.66	34.00	38.00
JLB_4	100.00	77.00	62.66	36.66	41.66
MLB_1	92.33	73.33	43.66	29.33	22.00
MLB_2	99.00	81.66	52.66	36.66	35.00
MLB_3	99.66	83.00	54.00	45.33	46.66
MLB_4	99.00	82.00	54.33	42.33	40.00
$JMLB_1$	93.33	89.33	81.66	81.33	68.66
$JMLB_2$	100.00	99.00	95.66	91.33	92.33
$JMLB_3$	100.00	83.33	60.66	51.33	56.33
$JMLB_4$	100.00	83.66	62.66	53.66	59.00

Table 9 - Performance of the lower bounds with respect to m

8 Conclusion

In this paper, we have investigated lower bounding strategies for the identical parallel machine scheduling problem with heads and tails. Despite its theoretical importance, this problem has received scant attention in the scheduling literature. We surveyed the existing lower bounds and we introduced a new lower bound based on the bin packing problem. Also, we devised several lifting procedures which are based on considering subset of jobs, machines, or both. These lifting procedures could be systematically used to enhance the quality of lower bounds. The e¤ectiveness of the proposed bounds has been demonstrated by the results of the computational experiments which show that the lifting procedures consistently improve the existing bounds. In particular, new lower bounds based on subset of jobs and machines were proven to outperform the best existing ones (PLB and JPPB).

An interesting issue that is worthy of future research is to embed the best newly derived bounds in a branch and bound algorithm for the exact solution of large $P=r_j;q_j=C_{max}$. Also, a second issue that deserves further investigation, is the generalization of the proposed lifting

procedures to other scheduling problems, such as the uniform parallel machine scheduling problem with heads and tails.

References

[1] J.D. Blocher and S. Chand, Scheduling of parallel processors: A posteriori bound on LPT sequencing and a two-step algorithm, Naval Research Logistics 38 (1991) 273-287.

[2] J. Carlier, The one machine sequencing problem, European Journal of Operational Research 11 (1982) 42-47.

[3] J. Carlier, Scheduling jobs with release dates and tails on identical machines to minimize the makespan, European Journal of Operational Research 29 (1987) 298-306.

[4] J. Carlier and B. Latapie, Une méthode arborescente pour résoudre les problèmes cumulatifs, RAIRO 25 (1991) 311-340.

[5] J. Carlier and E. Pinson, Jackson's pseudo preemptive schedule for the $P_m=r_j;q_j=C_{max}$ scheduling problem, Annals of Operations Research 83 (1998) 41-58.

[6] M. Dell'Amico and S. Martello, Optimal scheduling of tasks on identical parallel processors, ORSA Journal on Computing 7 (1995) 191-200.

[7] M.R. Garey and D.S Johnson, Strong NP-completeness results : Motivation, examples and implications, Journal of the Association of Computer Machinery 25 (1978) 499-508.

[8] A. Gharbi and M. Haouari, Minimizing Makespan on Parallel Machines Subject to Release Dates and Delivery Times, to appear in Journal of Scheduling (2002).

[9] H. Hoogeveen, C. Hurkens, J.K. Lenstra, A. Vandevelde, Lower bounds for the multiprocessor ‡ow shop, in: Second Workshop on Models and Algorithms for Planning and Scheduling, Wernigerode, 1995.

[10] W.A. Horn, Some simple scheduling algorithms, Naval Research Logistics Quarterly 21 (1974) 177-185.

[11] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D. Shmoys, Sequencing and Scheduling : Algorithms and Complexity, Handbooks in Operations Research and Management Science 4 (1993) 445-522.

[12] S. Martello and P. Toth, Knapsack problems: Algorithms and computer implementations, John Wiley & Sons, 1990.

[13] S.T. Webster, A general lower bound for the makespan problem, European Journal of Operational Research 89 (1996) 516-524.