WORKING PAPERS DA FEP

Investigação - Trabalhos em curso - nº 125, Abril de 2003

# IMPROVED LOWER BOUNDS FOR THE EARLY/TARDY SCHEDULING PROBLEM WITH NO IDLE TIME

Jorge M. S. Valente

Rui A. F. S. Alves

FACULDADE DE ECONOMIA

UNIVERSIDADE DO PORTO

**www.fep.up.pt**

# Improved Lower Bounds for the Early/Tardy Scheduling Problem with No Idle Time

Jorge M. S. Valente and Rui A. F. S. Alves
Faculdade de Economia do Porto
Rua Dr. Roberto Frias, 4200-464 Porto, Portugal
e-mails: jvalente@fep.up.pt; ralves@fep.up.pt

April 28, 2003

**Abstract**

In this paper we consider the single machine earliness/tardiness scheduling problem with no idle time. Two of the lower bounds previously developed for this problem are based on lagrangean relaxation and the multiplier adjustment method, and require an initial sequence. We investigate the sensitivity of the lower bounds to the initial sequence, and experiment with different dispatch rules and some dominance conditions. The computational results show that it is possible to obtain improved lower bounds by using a better initial sequence. The lower bounds are also incorporated in a branch-and-bound algorithm, and the computational tests show that one of the new lower bounds has the best performance for larger instances.

**Keywords:** scheduling, early/tardy, lower bound

**Resumo**

Neste artigo é considerado um problema de sequenciamento com uma única máquina e custos de posse e de atraso no qual não é permitida a existência de tempo morto. Dois dos *lower bounds* anteriormente apresentados para este problema são baseados na relaxação lagrangeana e no método de ajustamento dos multiplicadores, e requerem uma sequência inicial. A sensibilidade destes *lower bounds* à sequência inicial é analisada, sendo testadas diversas heurísticas e algumas regras de dominância. Os resultados computacionais

mostram que a utilização de melhores sequências iniciais permite melhorar os *lower bounds*. Os *lower bounds* são também incorporados num algoritmo do tipo *branch-and-bound* e os resultados computacionais mostram que um dos novos métodos permite a obtenção de melhores desempenhos para as instâncias de maior dimensão.

**Palavras-chave:** sequenciamento, custos de posse e atraso, *lower bound*

# 1 Introduction

In this paper we consider a single machine scheduling problem with earliness and tardiness costs that can be stated as follows. A set of $n$ independent jobs $\{J_1, J_2, \cdots, J_n\}$ has to be scheduled without preemptions on a single machine that can handle at most one job at a time. The machine and the jobs are assumed to be continuously available from time zero onwards and machine idle time is not allowed. Job $J_j, j = 1, 2, \cdots, n$, requires a processing time $p_j$ and should ideally be completed on its due date $d_j$. For any given schedule, the earliness and tardiness of $J_j$ can be respectively defined as $E_j = \max\{0, d_j - C_j\}$ and $T_j = \max\{0, C_j - d_j\}$, where $C_j$ is the completion time of $J_j$. The objective is then to find the schedule that minimizes the sum of the earliness and tardiness costs of all jobs $\sum_{j=1}^{n}(h_j E_j + w_j T_j)$, where $h_j$ and $w_j$ are the earliness and tardiness penalties of job $J_j$.

The inclusion of both earliness and tardiness costs in the objective function is compatible with the philosophy of just-in-time production, which emphasizes producing goods only when they are needed. The early cost may represent the cost of completing a project early in PERT-CPM analyses, deterioration in the production of perishable goods or a holding cost for finished goods. The tardy cost can represent rush shipping costs, lost sales and loss of goodwill. The assumption that no machine idle time is allowed reflects a production setting where the cost of machine idleness is higher than the early cost incurred by completing any job before its due date, or the capacity of the machine is limited when compared with its demand, so that the machine must indeed be kept running. Korman [4] and Landis [5] provide some specific examples.

As a generalization of weighted tardiness scheduling ([6]), the problem is strongly NP-hard. A large number of papers consider scheduling problems with both earliness and tardiness costs. We will only review those papers that examine a problem that is exactly the same as ours. For more information on earliness and tardiness scheduling, interested readers are referred to Baker and Scudder [2], who provide an excellent

review.

Abdul-Razaq and Potts [1] presented a branch-and-bound algorithm. Their lower bound procedure is based on the subgradient optimization approach and the dynamic programming state-space relaxation technique. The computational results indicate that the lower bound procedure is tight, but time consuming, and therefore problems with more than 25 jobs may require excessive solution times. Ow and Morton [9] develop several early/tardy dispatch rules and a filtered beam search procedure. Their computational studies show that the early/tardy dispatch rules, although clearly outperforming known heuristics that ignored the earliness costs, are still far from optimal. The filtered beam search procedure consistently provides very good solutions for small or medium size problems, but requires excessive computation times for larger problems (more than 100 jobs). Li [7] presented a branch-and-bound algorithm as well as a neighbourhood search heuristic procedure. The branch-and-bound algorithm is based on a decomposition of the problem into two subproblems and two efficient multiplier adjustment procedures for solving two Lagrangean dual subproblems. Their computational results show that the heuristic procedure is superior to Ow and Morton's filtered beam search approach in terms of efficiency and solution quality, and the branch-and-bound algorithm can obtain optimal solutions for problems with up to 50 jobs. Liaw [8] also proposed a branch-and-bound algorithm. The lower bounding procedure is based on a Lagrangean relaxation that decomposes the problem into two subproblems: a total weighted completion time subproblem, solved by a multiplier adjustment method, and a slack variable subproblem. Valente and Alves [14] propose two new heuristics, a dispatch rule and a greedy procedure, and also consider the best of the existing dispatch rules. They present functions that map some instance statistics into appropriate values for a lookahead parameter used by both dispatch rules and consider the use of dominance rules to improve the solutions obtained by the heuristics. The computational results show that the function-based versions of the heuristics outperform their fixed value counterparts and that the use of the dominance rules can indeed improve solution quality with little additional computational effort.

The multiplier adjustment procedures used in the lower bounds proposed by Li and Liaw require an initial sequence. In this paper we experiment with different initial sequences and analyse their effect on both the accuracy and the effectiveness of the lower bounds. Li and Liaw used Smith's [13] WSPT and WLPT rules to generate the initial sequences. We consider these two rules, as well as Jackson's [3]

EDD rule, the ATC heuristic for the weighted tardiness problem presented in [12], and an adaptation of the ATC heuristic to the weighted earliness problem, which we will denote as AEC. We also consider using dominance rules to improve the sequence generated by these heuristics. Rachamadugu's [11] rule for the weighted tardiness problem and a similar rule that is presented for the weighted earliness problem are used for this purpose. The multiplier adjustment procedures developed by Li assume that the initial sequences are produced by the WSPT and WLPT heuristics. Therefore, we had to make some slight changes to these procedures so that an arbitrary sequence could be used.

This paper is organized as follows. Section 2 describes the changes that had to be made to Li's multiplier adjustment procedures. The heuristics that were used to generate the initial sequence are presented in section 3. Section 4 describes the dominance rules that were used to improve the sequence generated by the heuristics. Section 5 describes the lower bounds that were considered, as well as the details of a branch-and-bound algorithm that was used to determine if the improvement provided by the most promising lower bounds is worthwhile in the context of an exact algorithm. The computational results are presented in section 6. Finally, conclusions are provided in section 7.

# 2 Modification of Li's lower bound procedure

In this section we describe how to modify Li's multiplier adjustment procedures so that any initial sequence can be used. Li decomposes the early/tardy problem into a weighted earliness subproblem and a weighted tardiness subproblem. The multiplier adjustment procedure presented by Potts and van Wassenhove [10] for the weighted tardiness problem can replace the procedure used by Li for the tardiness subproblem. In fact, Li's procedure is a simplified version of Potts and van Wassenhove's method, in that it assumes that the initial sequence is generated by the WSPT heuristic. Therefore, we will only focus on the changes required by the multiplier adjustment procedure for the earliness subproblem, and the reader is referred to Potts and van Wassenhove's paper for details concerning the weighted tardiness procedure.

Throughout this section, assume the jobs have been renumbered so that the initial sequence generated for the weighted earliness subproblem is $(J_1, J_2, \cdots, J_n)$. Li shows that a lower bound for the weighted earliness subproblem can be obtained by solving the following Lagrangean dual subproblem

$$\max \sum_{j=1}^{n} \lambda_j \left(d_j - C_j^*\right) \qquad (D_1)$$

subject to

$$\frac{\lambda_j}{p_j} \leq \frac{\lambda_{j+1}}{p_{j+1}}, j = 1, \ldots, n, \qquad (1)$$

$$0 \leq \lambda_j \leq h_j, j = 1, \ldots, n. \qquad (2)$$

where the $\lambda_j$'s and $C_j^*$'s are, respectively, the Lagrange multipliers and the jobs' completion times in the initial sequence (see Li's paper for details on the derivation of this dual subproblem). We define adjusted earliness penalties $\overline{h}_j$ as

$$\overline{h}_j = p_j * \min\left\{\frac{h_i}{p_i} : i = j, j+1, \ldots, n\right\}.$$

When the jobs are ordered according to the WLPT rule, as is the case in the procedure proposed by Li, we have $\overline{h}_j = h_j$ for all $j$. We now show that constraints 2 can be replaced by

$$0 \leq \lambda_j \leq \overline{h}_j, j = 1, \ldots, n, \qquad (3)$$

without altering the solution of problem $(D_1)$.

**Lemma 1** *Constraints (3) may replace constraints (2) without altering the solution of problem* $(D_1)$.

**Proof.** Suppose that for any job $J_j$ we have $\overline{h}_j = p_j * h_i/p_i$ for some $i, j \leq i \leq n$. The definition of the adjusted earliness penalties then implies that $\overline{h}_i = h_i$. From (1) and (2) we then have $\lambda_j/p_j \leq \lambda_i/p_i \leq h_i/p_i$, which implies that $\lambda_j \leq p_j * h_i/p_i = \overline{h}_j$. Therefore, constraints (3) are implicit in (1) and (2). From the definition of the adjusted earliness penalties we have $\overline{h}_j \leq h_j$, so when constraints (3) are imposed, constraints (2) are redundant and can therefore be dropped. ■

We now present a multiplier adjustment procedure that can be used to solve problem $(D_1)$ after an arbitrary initial sequence is provided. This procedure replaces $h_j$ with the adjusted penalties $\overline{h}_j$, but is otherwise identical to the method proposed by Li.

**Procedure Dual1**. Multiplier adjustment procedure to solve $(D_1)$

**Step 1**: Set $e_j = d_j - C_j^*$, for $j = 1, \ldots, n$, and compute $V_j = \sum_{i=j}^{n} p_i e_i$, for $j = 1, \ldots, n$.

**Step 2**: Set $V_{n+1} = 0$, $S_1 = \{n+1\}$ and $k = n$.

While $k \geq 1$ do

Let $m$ be the smallest integer in $S_1$.

If $V_m < V_k$, set $S_1 = S_1 \cup \{k\}$.

Set $k = k - 1$.

**Step 3**: Set $k = 1$ and $S_1 = S_1 - \{n+1\}$.

**Step 4**: While $k \leq n$ do

If $k \in S_1$, set $\lambda_k = \overline{h}_k$.

Else, if $k = 1$ set $\lambda_k = 0$ and if $k \neq 1$ set $\lambda_k = \lambda_{k-1}\,(p_k/p_{k-1})$.

set $k = k + 1$.

In the above procedure $S_1$ is the set of jobs that have a positive contribution to problem $(D_1)$. Therefore, the larger $\lambda_j$ is for $j \in S_1$, the larger is the solution to problem $(D_1)$ and the lower bound. In $(D_1)$, the largest feasible value for $\lambda_j$ is $\overline{h}_j$. Each job $J_j$ with $j \notin S_1$ has a negative contribution to problem $(D_1)$, so the smaller $\lambda_j$ is for $j \notin S_1$, the larger is the solution to problem $(D_1)$ and the lower bound. In $(D_1)$, the smallest feasible value for $\lambda_j$ is $\lambda_{j-1}\,(p_j/p_{j-1})$ for $j \notin S_1$ and $j \neq 1$, or 0 for $j \notin S_1$ and $j = 1$.

**Theorem 2** *Procedure Dual1 optimally solves $(D_1)$, i.e., the $\lambda_j^*$, for $j = 1, \ldots, n$ obtained from Dual1 are the optimal solution to $(D_1)$, where*

$$\lambda_1^* = 0, \text{ if } 1 \notin S_1 \tag{4}$$

$$\lambda_j^* = \overline{h}_j, \text{ if } j \in S_1 \tag{5}$$

$$\lambda_j^* = \lambda_{j-1}^*\,(p_j/p_{j-1})\,, \text{ if } j \notin S_1 \text{ and } j > 1. \tag{6}$$

**Proof.** The $\lambda_j^*$ are clearly feasible, so we need to prove their optimality. In procedure Dual1, $S_1$ can be regarded as an ordered integer set $\{s_k, \ldots, s_1\}$ with its elements in decreasing order of their values, where $k$ is the number of jobs in $S_1$. Equations

(4) and (6) were already present in the original procedure, and the proof of their optimality is identical to the one presented by Li. To establish equation (5) we first show, by contradiction, that $\lambda^*_{s_k} = \overline{h}_{s_k}$. Suppose $\lambda^*_{s_k} < \overline{h}_{s_k}$. Then

$$\begin{aligned}
\sum_{j=1}^{n} \lambda^*_j e_j &= \sum_{j=1}^{s_k-1} \lambda^*_j e_j + \sum_{j=s_k}^{n} \lambda^*_j e_j \\
&= \sum_{j=1}^{s_k-1} \lambda^*_j e_j + \sum_{j=s_k}^{n} \left(\lambda^*_{s_k}/p_{s_k}\right) p_j e_j \\
&< \sum_{j=1}^{s_k-1} \lambda^*_j e_j + \sum_{j=s_k}^{n} \left(\overline{h}^*_{s_k}/p_{s_k}\right) p_j e_j.
\end{aligned}$$

Setting

$$\lambda'_j = \begin{cases} \left(\overline{h}^*_{s_k}/p_{s_k}\right) p_j,\ j \in \{s_k, \ldots, n\}, \\ \lambda^*_j,\ j \in \{1, \ldots, s_k - 1\}. \end{cases}$$

we can obtain a solution $\lambda'$ that is also feasible since $\lambda^*_{s_k-1}/p_{s_k-1} \leq \overline{h}_{s_k}/p_{s_k}$ (because, by the definition of the adjusted earliness penalties, $\overline{h}^*_{s_k-1}/p_{s_k-1} \leq \overline{h}_{s_k}/p_{s_k}$ and, from constraint (3), $\lambda^*_{s_k-1} \leq \overline{h}_{s_k-1}$). Furthermore, this new solution has a larger objective function value, contradicting the assumption that the original solution was optimal. Therefore we must have $\lambda^*_{s_k} = \overline{h}_{s_k}$. The above argument can be repeated for $j = 1, \ldots, s_k - 1$, thus establishing (5). ■

# 3 Heuristic procedures

In this section we describe the several dispatch heuristics that were used to generate initial sequences for the lower bounding procedures. These heuristics and their main characteristics are summarized in Table 1.

| Rule | Rank and priority index | Time complexity |
|---|---|---|
| WSPT | $\max\left(\frac{w_j}{p_j}\right)$ | $O(n \log n)$ |
| WLPT | $\max\left(\frac{p_j}{h_j}\right)$ | $O(n \log n)$ |
| EDD | $\min(d_j)$ | $O(n \log n)$ |
| ATC | $\max\left[\frac{w_j}{p_j} \exp\left(-\frac{(d_j - t - p_j)^+}{k\overline{p}}\right)\right]$ | $O(n^2)$ |
| AEC | $\max\left[\frac{h_j}{p_j} \exp\left(-\frac{(t - d_j)^+}{k\overline{p}}\right)\right]$ | $O(n^2)$ |

Table 1: Dispatch rules used in lower bounding procedures

The weighted shortest processing time (WSPT) rule was introduced by Smith [13] and sorts the jobs in non increasing order of the ratio $\frac{w_j}{p_j}$. This rule is optimal for the weighted tardiness problem if it results in a schedule that does not have any early jobs. The weighted longest processing time (WLPT) rule was also introduced by Smith and sorts the jobs in non increasing order of the ratio $\frac{p_j}{h_j}$. If this rule results in a schedule that does not have any tardy jobs, then it is optimal for the weighted earliness problem with no idle time allowed (if idle time is allowed, we can simply delay the jobs so that no job is completed before its due date). The earliest due date (EDD) rule, presented by Jackson [3], simply sorts the jobs in non decreasing order of their due dates. Since these three dispatch rules only involve simple sorting procedures, their time complexity is $O(n \log n)$.

The Apparent Tardiness Cost (ATC) heuristic, presented in [12], selects, whenever the machine becomes available, the unscheduled job with the highest priority index $\frac{w_j}{p_j} \exp\left(-\frac{(d_j - t - p_j)^+}{k\overline{p}}\right)$, where $\overline{p}$ is the average processing time, $t$ is the current time and $k$ is a lookahead empirical parameter. The priority of a job is low when that job is still quite early, and gradually increases until it achieves its maximum value of $\frac{w_j}{p_j}$ when the job is late (or on time). Several computational studies have consistently shown that the ATC is one of the best dispatch heuristics available for the weighted tardiness problem. If the WSPT sequence results in a schedule that does not have any early jobs, and is therefore optimal for the weighted tardiness problem, the ATC rule will always generate that optimal WSPT sequence. The Apparent Earliness Cost is an adaptation of the ATC rule to the weighted earliness problem with no idle time allowed. It differs from the ATC rule in that the schedule is built backwards, i.e., at each iteration we select a job that will be scheduled just before the current partial sequence. At each iteration we select the unscheduled job with the highest priority index $\frac{h_j}{p_j} \exp\left(-\frac{(t - d_j)^+}{k\overline{p}}\right)$, where $\overline{p}$ is the average processing time, $k$ is an empirical parameter and $t$ is the time at which the next selected job will be completed. The priority of a job is low when that job is still quite tardy, and gradually increases until it achieves its maximum value of $\frac{h_j}{p_j}$ when the job is early (or on time). The time complexity of both the ATC and AEC heuristics is $O(n^2)$. If the WLPT sequence results in a schedule that does not have any tardy jobs, and is therefore optimal for the weighted earliness problem, the AEC heuristic will always produce this optimal WLPT sequence. In the first iteration, there exists at least one job that is early or on time: the job scheduled last in the WLPT sequence. This job will have the highest $h_j/p_j$ of all jobs (since it was selected last by the WLPT rule),

which is also the highest priority any unscheduled job can attain. Therefore, this will indeed be the job selected by the AEC heuristic. This reasoning can be repeated for the remaining iterations, thus proving that the AEC heuristic will generate the WLPT sequence.

# 4 Dominance rules

In this section we present two dominance rules that were used to improve the sequences generated by the heuristics described in the previous section. These dominance rules identify a condition that holds for adjacent jobs in an optimal sequence. The following rule has been developed by Rachamadugu [11] for the weighted tardiness problem.

**Theorem 3** *Consider any two adjacent jobs in an optimal sequence for the weighted tardiness problem. Either the following condition holds or an alternative optimal sequence can be constructed by interchanging the adjacent jobs in the optimal sequence:*

$$\frac{w_i}{p_i}\left(1-\frac{(d_i-t-p_i)^+}{p_j}\right) \geq \frac{w_j}{p_j}\left(1-\frac{(d_j-t-p_j)^+}{p_i}\right).$$

*In this expression $i$ denotes the index of the job in the $i$th position, $j$ is the index of the job in the $(i+1)$st and $t$ is the start time of $J_i$.*

**Proof.** See the proof of Proposition 1 in [11]. ■

If this condition does not hold for two adjacent jobs, interchanging them will either lower the schedule cost, or leave it unchanged when both jobs are early in either position. We now present an adaptation of this rule to the weighted earliness problem.

**Theorem 4** *Consider any two adjacent jobs in an optimal sequence for the weighted earliness problem. Either the following condition holds or an alternative optimal sequence can be constructed by interchanging the adjacent jobs in the optimal sequence:*

$$\frac{h_i}{p_i}\left(1-\frac{(t+p_i+p_j-d_i)^+}{p_j}\right) \leq \frac{h_j}{p_j}\left(1-\frac{(t+p_i+p_j-d_j)^+}{p_i}\right).$$

*In this expression $i$ denotes the index of the job in the $i$th position, $j$ is the index of the job in the $(i+1)$st and $t$ is the start time of $J_i$.*

**Proof.** We must show that when the condition does not hold for two adjacent jobs, interchanging those jobs either lowers the schedule cost or leaves it unchanged. This can be done using simple pairwise interchange arguments. When the interchange of two adjacent jobs is considered, there are 9 possible cases, as shown in Table 2 (E is for early and T is tardy). Jobs that are on time are considered tardy, since both have no cost. Let $C_{ij}$ be the cost of the subsequence $(J_i, J_j)$ and let $C_{ji}$ be the cost of the reversed subsequence $(J_j, J_i)$. In case 1 both jobs are tardy in either position, so $C_{ij} = C_{ji} = 0$. Therefore, if the rule is violated, the jobs can be interchanged without changing the schedule cost. In all other cases, the condition is necessary, i.e., if the rule is violated, interchanging jobs will lower the schedule cost. In case 2, both jobs are early even when scheduled on the second position, so we have $t+p_i+p_j < d_i$ and $t + p_i + p_j < d_j$. Therefore, the rule reduces to $h_i/p_i \leq h_j/p_j$ or $h_j/p_i \geq h_i/p_j$. We also have

$$\begin{aligned} C_{ij} &= h_i\left(d_i - t - p_i\right) + h_j\left(d_j - t - p_i - p_j\right) \\ C_{ji} &= h_j\left(d_j - t - p_j\right) + h_i\left(d_i - t - p_i - p_j\right) \\ &= C_{ij} + h_j p_i - h_i p_j. \end{aligned}$$

Therefore, when the rule does not hold we have $C_{ij} > C_{ji}$, and an interchange will decrease the schedule cost. The same procedure can be repeated for the remaining cases to complete the proof. For the sake of brevity, we omit the details. ■

| Case | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| before interchange | | | | | | | | | |
| Job $i$ | T | E | E | T | E | E | E | E | T |
| Job $j$ | T | E | E | T | T | T | T | T | E |
| after interchange | | | | | | | | | |
| Job $i$ | T | E | T | T | E | T | E | T | T |
| Job $j$ | T | E | E | E | T | E | E | T | E |

Table 2: Possible cases for job interchanges

# 5 Lower bounds and implementation of the branch-and-bound algorithm

In this section we describe the lower bounds that were considered, as well as the details of a branch-and-bound algorithm that was used to compare the best of the

existing bounds with the most promising of the new methods. We considered 6 lower bounds based on the procedure presented by Li, with the adaptations described in this paper. The first of these lower bounds, denoted by Li, is simply the original procedure that uses the WLPT and WSPT rules to generate the initial sequences for the weighted earliness and weighted tardiness subproblems, respectively. Lower bound Li EDD uses the EDD rule to generate the initial sequence for both subproblems, while Li AC denotes the procedure that uses the AEC (ATC) heuristic for the earliness (tardiness) subproblem. The remaining 3 lower bounds based on Li's procedure use these same heuristics and then apply the dominance rules presented in the previous section to improve the sequences generated by the heuristics. These lower bounds will be denoted by appending DR to the identifier of the corresponding lower bound where no dominance rules are applied. Rachamadugu's rule is used for the weighted tardiness subproblem and the rule we have developed for the earliness criterion is used for the weighted earliness subproblem. When a pair of adjacent jobs in a sequence violates a rule, those jobs are swapped if that change reduces the objective function value. The rules are applied repeatedly until no improvement is found in a complete iteration. The complexity of the dominance rules is $O(n)$ per iteration, and the total complexity depends on the number of times the rule produces an improvement.

Similarly, we considered 6 lower bounds based on the procedure presented by Liaw. The first of these, denoted by Lw, is once again the procedure originally proposed by Liaw. In this procedure, the initial sequence is generated by the WLPT rule when the lateness factor of a problem is low ($\leq 0.5$), and by the WSPT rule when the lateness factor is high ($\geq 0.5$). The lower bound denoted by Lw EDD uses the EDD rule, while Lw AC uses the AEC (ATC) rule when the lateness factor is low (high). The remaining 3 lower bounds, that will once more be denoted by appending DR to the identifiers of the simpler procedures, use the same heuristics and then apply the dominance rules. Rachamadugu's rule is used when the tardiness factor is high and the weighted earliness rule is used when the tardiness factor is low.

We now consider the implementation details of the branch-and-bound algorithm. We first present two dominance rules for the early/tardy scheduling problem that were used to reduce the number of nodes in the search tree. In the following, Theorem 5 is a result presented in [9] and Theorem 6 is developed in [8].

**Theorem 5** *All adjacent pairs of jobs in an optimal schedule must satisfy the fol-*

*lowing condition:*

$$w_i p_j - \Omega_{ij}\,(w_i + h_i) \geq w_j p_i - \Omega_{ji}\,(w_j + h_j)$$

*where job $J_i$ immediately precedes $J_j$, and $\Omega_{ij}$ and $\Omega_{ji}$ are defined as*

$$\Omega_{xy} = \begin{cases} 0 & \textit{if} \quad s_x \leq 0 \\ s_x & \textit{if} \quad 0 < s_x < p_y \\ p_y & \textit{otherwise,} \end{cases}$$

*where $s_x = d_x - t - p_x$ is the slack of job $J_x$ and $t$ is the sum of the processing times of all jobs preceding $J_i$.*

**Proof.** See the proof of Theorem 1 in [9]. ■

**Theorem 6** *All non-adjacent pairs of jobs $J_i$ and $J_j$ with $p_i = p_j$ and $J_i$ preceding $J_j$ must satisfy the following condition in an optimal schedule:*

$$w_i\,(p_j + \Delta) - \Lambda_{ij}\,(w_i + h_i) \geq w_j\,(p_i + \Delta) - \Lambda_{ji}\,(w_j + h_j)$$

*with $\Lambda_{ij}$ and $\Lambda_{ji}$ defined as*

$$\Lambda_{xy} = \begin{cases} 0 & \textit{if} \quad s_x \leq 0 \\ s_x & \textit{if} \quad 0 < s_x < p_y + \Delta \\ p_y + \Delta & \textit{otherwise,} \end{cases}$$

*where $s_x = d_x - t - p_x$ is the slack of job $J_x$, $\Delta$ is the sum of the processing times of all jobs between $J_i$ and $J_j$ and $t$ is the sum of the processing times of all jobs preceding $J_i$.*

**Proof.** See Theorem 5 in [8]. ■

We can now describe the implementation of our branch-and-bound algorithm. First, we use one of the heuristic procedures presented in [14] to calculate an upper bound on the optimal schedule cost. This procedure is a modified version of the EXP-ET heuristic developed in [9], since the value of a lookahead parameter is determined by a function, and dominance rules are used to further improve the sequence generated by this heuristic. The upper bound is updated whenever a feasible schedule with a lower cost is found during the branching process. Motivated

by results presented in [7] and [8], when the lateness factor of a problem is high, we adopt a forward-sequencing branching rule where a node at level $l$ of the search tree corresponds to a sequence with $l$ jobs fixed in the first $l$ positions. When the lateness factor is low, we adopt a backward-sequencing branching rule where a node at level $l$ of the search tree corresponds to a sequence with $l$ jobs fixed in the last $l$ positions.

The depth-first strategy is used to search the tree, and ties are broken by selecting the node with the smallest value of the associated partial schedule cost plus the associated lower bound for the unscheduled jobs. We apply the following three tests to decide whether a node should be discarded or not. In the first test, the adjacent dominance rule of Theorem 5 is applied to the two jobs most recently added to the node's partial schedule. In the second test, the non adjacent rule of Theorem 6 is applied. Finally, if the node is not eliminated by the two previous tests, a lower bound is calculated for that node. If the lower bound plus the cost of the associated partial schedule is larger than or equal to the current upper bound, the node is discarded.

# 6 Computational Results

The lower bounds were tested on a set of randomly generated problems with 15, 20, 25, 30, 40, 50, 100, 500 and 1000 jobs. These problems were generated as follows. For each job $J_j$ an integer processing time $p_j$, an integer earliness penalty $h_j$ and an integer tardiness penalty $w_j$ were generated from one of the two uniform distributions $[1, 10]$ and $[1, 100]$, to create low and high variability, respectively. For each job $J_j$, an integer due date $d_j$ is generated from the uniform distribution $[P(1 - LF - RDD/2), P(1 - LF + RDD/2)]$, where $P$ is the sum of the processing times of all jobs, $LF$ is the lateness factor, set at 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0, and $RDD$ is the range of due dates, set at 0.2, 0.4, 0.6 and 0.8. The values considered for each of the factors involved in the instance generation process are summarized in Table 3. For each combination of problem size, processing time and penalty variability, $LF$ and $RDD$, 20 instances were generated. All the algorithms were coded in Visual C++ 6.0 and executed on a Pentium IV-1500 personal computer. Throughout this section, and in order to avoid excessively large tables, we will sometimes present results only for some representative cases.

In Table 4 we present the average value of the lower bounds before and after

| Factors | Settings |
|---|---|
| Number of jobs | 15, 20, 25, 30, 40, 50, 100, 500, 1000 |
| Processing time and penalties variability | $[1, 10]$, $[1, 100]$ |
| Lateness factor | 0.0, 0.2, 0.4, 0.6, 0.8, 1.0 |
| Range of due dates | 0.2, 0.4, 0.6, 0.8 |

Table 3: Experimental design

applying the dominance rules, and the average of the relative improvements (% imp.), calculated as $\frac{LBDR-LB}{LB} * 100$, where $LB$ and $LBDR$ represent the values of the lower bound before and after the use of the dominance rules, respectively. In Table 5 we give the number of instances for which the lower bounds that use the dominance rules perform better (>), equal (=) or worse (<) than the corresponding lower bounds that do not use those rules. We also performed a test to determine if the differences between these lower bounds are statistically significant. Given that the lower bounds were used on exactly the same instances, a paired-samples test is appropriate. Since some of the hypothesis of the paired-samples t-test were not met, the non-parametric Wilcoxon test was selected. In Table 5 we also present the significance (sig.) values of this test, i.e., the confidence level values above which the equal distribution hypothesis is to be rejected.

| | | n = 25 | | | n = 50 | | | n = 100 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| var. | LB | before | after | % imp. | before | after | % imp. | before | after | % imp. |
| low | Li | 2848 | 2884 | 1,27 | 11177 | 11336 | 1,12 | 43700 | 44455 | 1,32 |
| | Li EDD | 1253 | 2523 | 84,86 | 3810 | 8906 | 104,85 | 12027 | 30347 | 108,71 |
| | Li AC | 2887 | 2888 | 0,06 | 11348 | 11352 | 0,05 | 44182 | 44190 | 0,03 |
| | Lw | 2877 | 2913 | 1,27 | 11239 | 11398 | 1,11 | 43823 | 44579 | 1,31 |
| | Lw EDD | 1341 | 2579 | 75,42 | 4006 | 9031 | 97,76 | 12446 | 30666 | 104,97 |
| | Lw AC | 2915 | 2917 | 0,32 | 11414 | 11417 | 0,04 | 44335 | 44343 | 0,02 |
| high | Li | 219836 | 223302 | 1,68 | 858537 | 875108 | 1,67 | 3379035 | 3451238 | 1,60 |
| | Li EDD | 76489 | 192715 | 196,97 | 187091 | 648861 | 250,49 | 460196 | 2212643 | 346,00 |
| | Li AC | 223520 | 223716 | 0,14 | 876627 | 877132 | 0,08 | 3437143 | 3438207 | 0,04 |
| | Lw | 222136 | 225596 | 1,58 | 863673 | 880230 | 1,63 | 3389548 | 3461669 | 1,58 |
| | Lw EDD | 83319 | 196716 | 151,23 | 202679 | 659598 | 208,15 | 493318 | 2234936 | 303,52 |
| | Lw AC | 225847 | 226042 | 0,12 | 881968 | 882462 | 0,05 | 3449396 | 3450485 | 0,04 |

Table 4: Lower bound average values and relative improvement

From these results, we can see that both Li and Liaw's lower bounds are sensitive to the choice of the initial sequence. The EDD rule, particularly when not followed by the dominance rules, is clearly inferior to the other heuristics. For instances

| | | n = 25 | | | | n = 50 | | | | n = 100 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| var. | LB | > | = | < | sig. | > | = | < | sig. | > | = | < | sig. |
| low | Li DR | 248 | 232 | 0 | 0,000 | 254 | 226 | 0 | 0,000 | 262 | 218 | 0 | 0,000 |
| | Li EDD DR | 414 | 66 | 0 | 0,000 | 403 | 77 | 0 | 0,000 | 395 | 85 | 0 | 0,000 |
| | Li AC DR | 183 | 297 | 0 | 0,000 | 249 | 231 | 0 | 0,000 | 299 | 181 | 0 | 0,000 |
| | Lw DR | 271 | 170 | 39 | 0,000 | 323 | 81 | 76 | 0,000 | 364 | 18 | 98 | 0,000 |
| | Lw EDD DR | 434 | 23 | 23 | 0,000 | 431 | 15 | 34 | 0,000 | 430 | 5 | 45 | 0,000 |
| | Lw AC DR | 185 | 277 | 18 | 0,000 | 269 | 148 | 63 | 0,000 | 355 | 22 | 103 | 0,000 |
| high | Li DR | 252 | 228 | 0 | 0,000 | 266 | 214 | 0 | 0,000 | 259 | 221 | 0 | 0,000 |
| | Li EDD DR | 410 | 70 | 0 | 0,000 | 427 | 53 | 0 | 0,000 | 421 | 59 | 0 | 0,000 |
| | Li AC DR | 235 | 245 | 0 | 0,000 | 276 | 204 | 0 | 0,000 | 295 | 184 | 1 | 0,000 |
| | Lw DR | 251 | 220 | 9 | 0,000 | 291 | 167 | 22 | 0,000 | 275 | 156 | 49 | 0,000 |
| | Lw EDD DR | 413 | 62 | 5 | 0,000 | 434 | 40 | 6 | 0,000 | 432 | 37 | 11 | 0,000 |
| | Lw AC DR | 235 | 241 | 4 | 0,000 | 279 | 189 | 12 | 0,000 | 316 | 133 | 31 | 0,000 |

Table 5: Comparison of lower bound values and statistical test

with up to 50 jobs the AEC/ATC heuristics provided the best results, while the WSPT/WLPT rules produced the highest lower bounds for the larger (500 and 1000 jobs) instances. The use of the dominance rules leads to improved lower bound performance, since the average lower bound value is larger (particularly for the EDD rule), and the results are better or equal for most, or even all, of the test instances. The Wilcoxon test values also indicate that the difference is statistically significant. The effect of the dominance rules also seems to be higher for instances with a larger processing time and penalty variability.

In Table 6 we present the number of times each version of Liaw's lower bound performs better (>), equal (=) or worse (<) than the corresponding Li lower bound, and the average of the relative improvements (% imp.), calculated as $\frac{Lw-Li}{Li} * 100$, where $Lw$ and $Li$ represent Liaw and Li's lower bound values, respectively. A test was also performed to determine if the differences are statistically significant. The Wilcoxon test was once again chosen and its significance values (sig.) are given in Table 6. In Table 7 we present the $LF$ and $RDD$ effect on the average relative improvement for the original versions on instances with 50 jobs and low variability.

From Tables 4, 6 and 7 we can conclude that Liaw's lower bound is superior: the average lower bound value is higher, and for most or even all of the test instances it is better than or equal to Li's lower bound. The Wilcoxon test significance values also indicate that the differences between these lower bounds are statistically significant. The relative improvement decreases as the instance size increases, and is higher for the instances with a larger processing time and penalty variability. The results in

| | | n = 25 | | | | | n = 50 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| var. | LB | > | = | < | % imp. | sig. | > | = | < | % imp. | sig. |
| low | Lw | 348 | 124 | 8 | 12,31 | 0,000 | 377 | 82 | 21 | 4,31 | 0,000 |
| | Lw DR | 355 | 123 | 2 | 12,30 | 0,000 | 391 | 72 | 17 | 4,30 | 0,000 |
| | Lw EDD | 480 | 0 | 0 | 21,09 | 0,000 | 480 | 0 | 0 | 11,09 | 0,000 |
| | Lw EDD DR | 395 | 85 | 0 | 20,56 | 0,000 | 437 | 43 | 0 | 8,08 | 0,000 |
| | Lw AC | 352 | 128 | 0 | 11,65 | 0,000 | 394 | 84 | 2 | 4,66 | 0,000 |
| | Lw AC DR | 355 | 124 | 1 | 16,52 | 0,000 | 392 | 87 | 1 | 4,64 | 0,000 |
| high | Lw | 372 | 108 | 0 | 22,18 | 0,000 | 409 | 69 | 2 | 7,02 | 0,000 |
| | Lw DR | 377 | 103 | 0 | 22,02 | 0,000 | 415 | 65 | 0 | 6,95 | 0,000 |
| | Lw EDD | 480 | 0 | 0 | 56,06 | 0,000 | 480 | 0 | 0 | 32,88 | 0,000 |
| | Lw EDD DR | 411 | 69 | 0 | 41,63 | 0,000 | 445 | 34 | 1 | 23,02 | 0,000 |
| | Lw AC | 380 | 100 | 0 | 21,52 | 0,000 | 416 | 64 | 0 | 7,08 | 0,000 |
| | Lw AC DR | 379 | 101 | 0 | 21,49 | 0,000 | 416 | 64 | 0 | 7,02 | 0,000 |

Table 6: Comparison of Li and Liaw's lower bounds and statistical test

| | RDD | | | |
|---|---|---|---|---|
| LF | 0.2 | 0.4 | 0.6 | 0.8 |
| 0.0 | 0,00 | 0,01 | 0,01 | 0,01 |
| 0.2 | 0,42 | 0,48 | 0,62 | 0,42 |
| 0.4 | 13,73 | 16,78 | 18,14 | 16,61 |
| 0.6 | 8,70 | 7,21 | 7,52 | 11,26 |
| 0.8 | 0,37 | 0,30 | 0,38 | 0,46 |
| 1.0 | 0,00 | 0,01 | -0,02 | 0,08 |

Table 7: Relative improvement of Liaw's lower bound for instances with 50 jobs and low variability

Table 7 indicate that the $LF$ parameter has an important effect. Liaw's lower bound values are much higher than Li's for lateness factors of 0.6 and (particularly) 0.4. For the remaining $LF$ values, however, the two lower bound procedures are quite close.

In Table 8 we present the number of times each lower bound achieves the best result and the average of the relative deviations from the optimum (% dev. opt.), calculated as $\frac{O-LB}{O} * 100$, where $O$ and $LB$ represent the optimum objective function value and the lower bound value, respectively. In table 9 we present the $LF$ and $RDD$ effect on the relative deviation from the optimum for Liaw's original lower bound on instances with 30 jobs and low variability. The results on Table 8 once again show that the use of the dominance rules leads to improved lower bounds, and that Liaw's lower bounds provide better results than Li's counterparts. We can also see that the lower bounds are not very tight on average, as even the best are still more than 20% below the optimum. We can see from Table 9 that the $LF$ parameter once again has an important effect. When the lateness factor is either 0.0 or 1.0, the lower bound value is quite close to the optimum. As the $LF$ moves towards more intermediate values, the lower bound performance degrades considerably, and it's quite poor for $LF$ values of 0.6 and (especially) 0.4. This result is to be expected, since the early/tardy problem is more difficult for intermediate $LF$ values. If all the jobs were early (tardy), an optimum schedule could be easily determined, and for a $LF$ of 0.0 (1.0) most jobs will indeed be early (tardy). For the intermediate $LF$ values, however, there is a greater balance between the number of early and tardy jobs, and the problem becomes harder.

In Table 10 we present the lower bounds average runtimes, in seconds, for instances with 500 and 1000 jobs. These results show that Liaw's lower bounds not only provide better results, but also require less computation time than Li's lower bounds. We have already seen that the results of Liaw's original lower bound can be improved by the use of the dominance rules or the AEC/ATC heuristics. However, these new lower bounds, particularly the ones that use the AEC/ATC heuristics, require a noticeably higher computation time. Therefore, it cannot be guaranteed that their use will reduce the computation time of a branch-and-bound algorithm. In order to determine if the improvement due to these lower bounds is indeed worthwhile in the context of an exact algorithm, the instances with up to 30 jobs were solved to optimality with a branch-and-bound algorithm using Liaw's original lower bound, as well as the Lw DR, Lw AC and Lw AC DR lower bounds.

| | var. low | | | | var. high | | | |
|---|---|---|---|---|---|---|---|---|
| | n = 20 | | n = 30 | | n = 20 | | n = 30 | |
| LB | % dev. opt. | best | % dev. opt. | best | % dev. opt. | best | % dev. opt. | best |
| Li | 25,55 | 38 | 23,49 | 13 | 26,08 | 19 | 26,20 | 10 |
| Li DR | 24,59 | 114 | 22,25 | 68 | 24,70 | 94 | 24,82 | 72 |
| Li EDD | 60,85 | 0 | 65,43 | 0 | 69,04 | 0 | 76,39 | 0 |
| Li EDD DR | 33,26 | 91 | 35,21 | 60 | 35,85 | 79 | 42,07 | 57 |
| Li AC | 24,44 | 81 | 22,05 | 44 | 24,52 | 40 | 24,60 | 24 |
| Li AC DR | 24,38 | 129 | 22,00 | 96 | 24,40 | 112 | 24,51 | 87 |
| Lw | 22,87 | 178 | 21,46 | 118 | 23,52 | 191 | 23,81 | 173 |
| Lw DR | 21,95 | 297 | 20,24 | 216 | 22,17 | 338 | 22,44 | 288 |
| Lw EDD | 56,00 | 26 | 61,58 | 2 | 64,03 | 20 | 72,18 | 7 |
| Lw EDD DR | 29,52 | 199 | 32,22 | 145 | 32,25 | 214 | 38,72 | 167 |
| Lw AC | 21,75 | 245 | 20,00 | 189 | 21,95 | 240 | 22,21 | 201 |
| Lw AC DR | 21,69 | 366 | 19,95 | 353 | 21,83 | 432 | 22,13 | 401 |

Table 8: Relative deviation from the optimum and number of times lower bound is the best

| | RDD | | | |
|---|---|---|---|---|
| LF | 0.2 | 0.4 | 0.6 | 0.8 |
| 0.0 | 0,23 | 0,91 | 2,53 | 3,37 |
| 0.2 | 16,99 | 5,54 | 7,82 | 10,55 |
| 0.4 | 75,50 | 69,29 | 57,44 | 44,94 |
| 0.6 | 59,57 | 56,09 | 41,57 | 33,42 |
| 0.8 | 7,42 | 3,11 | 4,84 | 8,85 |
| 1.0 | 0,15 | 0,63 | 1,60 | 2,63 |

Table 9: Relative deviation from the optimum for lower bound Lw on instances with 30 jobs and low variability

| | var. low | | var. high | |
|---|---|---|---|---|
| LB | n = 500 | n = 1000 | n = 500 | n = 1000 |
| Li | 0,001 | 0,003 | 0,002 | 0,003 |
| Li DR | 0,005 | 0,016 | 0,006 | 0,019 |
| Li EDD | 0,000 | 0,001 | 0,000 | 0,001 |
| Li EDD DR | 0,014 | 0,057 | 0,016 | 0,055 |
| Li AC | 0,037 | 0,152 | 0,039 | 0,150 |
| Li AC DR | 0,038 | 0,153 | 0,041 | 0,153 |
| Lw | 0,001 | 0,002 | 0,001 | 0,002 |
| Lw DR | 0,003 | 0,009 | 0,003 | 0,010 |
| Lw EDD | 0,000 | 0,001 | 0,000 | 0,001 |
| Lw EDD DR | 0,012 | 0,051 | 0,014 | 0,049 |
| Lw AC | 0,015 | 0,060 | 0,016 | 0,060 |
| Lw AC DR | 0,015 | 0,061 | 0,017 | 0,062 |

Table 10: Lower bound runtimes (in seconds)

In Table 11 we give the branch-and-bound average computation times (in seconds), for each lower bound. In Table 12 we present the effect of the $LF$ and $RDD$ parameters on the branch-and-bound runtimes for the 30-job instances and both Lw and Lw DR lower bounds. We can see from Table 11 that the lower bounds that use the AEC/ATC heuristics lead to higher computation times than the original procedure. The Lw DR lower bound, however, leads in some cases to lower computation times, particularly for the larger instances and when the processing time and penalty variability is high. From Table 12 it is clear that the lateness factor has an important effect on the computation times. The branch-and-bound algorithm is much slower for the intermediate $LF$ values, as a consequence of the much lower accuracy of the lower bound. The new versions of Liaw's lower bound provide better results than the original method when the $RDD$ value is high.

In Table 13 we present the average number of nodes generated by the branch-and-bound algorithm (nodes), as well as some data on the relative importance of the three fathoming tests, namely the average percentage of nodes eliminated by the adjacent rule (% AR.), the non-adjacent rule (% NAR) and the lower bound (% LB). In Table 14 we present the $LF$ and $RDD$ effect on the average number of nodes generated and the average percentage of nodes eliminated by the lower bound test for the 30-job instances when lower bound Lw DR is used.

Only a very small percentage of nodes is eliminated by the non-adjacent dominance rule. The proportion of nodes fathomed by this rule increases with the

| | | Lower bound | | | |
|---|---|---|---|---|---|
| var. | n | Lw | Lw DR | Lw AC | Lw AC DR |
| low | 15 | 0,011 | 0,012 | 0,012 | 0,014 |
| | 20 | 0,102 | 0,107 | 0,118 | 0,132 |
| | 25 | 1,015 | 1,019 | 1,181 | 1,328 |
| | 30 | 15,590 | 12,537 | 15,858 | 17,814 |
| high | 15 | 0,010 | 0,011 | 0,011 | 0,013 |
| | 20 | 0,088 | 0,084 | 0,094 | 0,104 |
| | 25 | 1,079 | 0,983 | 1,146 | 1,283 |
| | 30 | 9,816 | 9,357 | 11,016 | 12,392 |

Table 11: Branch-and-Bound runtimes (in seconds)

| | | Lw | | | | Lw DR | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | RDD | | | | | | | |
| var. | LF | 0.2 | 0.4 | 0.6 | 0.8 | 0.2 | 0.4 | 0.6 | 0.8 |
| low | 0.0 | 0,019 | 0,069 | 0,591 | 2,272 | 0,016 | 0,052 | 0,221 | 0,209 |
| | 0.2 | 0,980 | 0,348 | 3,592 | 10,845 | 1,166 | 0,280 | 1,467 | 1,285 |
| | 0.4 | 104,359 | 33,695 | 51,360 | 66,972 | 131,862 | 39,801 | 36,762 | 12,782 |
| | 0.6 | 21,472 | 12,811 | 26,050 | 31,559 | 27,466 | 15,096 | 12,135 | 18,320 |
| | 0.8 | 0,358 | 0,147 | 1,696 | 2,977 | 0,383 | 0,128 | 0,473 | 0,537 |
| | 1.0 | 0,009 | 0,056 | 0,300 | 1,634 | 0,008 | 0,029 | 0,121 | 0,294 |
| high | 0.0 | 0,029 | 0,080 | 0,187 | 1,091 | 0,023 | 0,047 | 0,071 | 0,166 |
| | 0.2 | 0,736 | 0,645 | 1,002 | 9,004 | 0,859 | 0,554 | 0,403 | 2,414 |
| | 0.4 | 24,956 | 20,977 | 30,029 | 85,734 | 31,834 | 24,202 | 18,973 | 69,966 |
| | 0.6 | 15,797 | 8,036 | 12,678 | 8,445 | 50,730 | 7,725 | 10,024 | 3,660 |
| | 0.8 | 0,365 | 0,597 | 3,327 | 10,575 | 0,410 | 0,436 | 0,609 | 1,211 |
| | 1.0 | 0,013 | 0,057 | 0,218 | 1,001 | 0,013 | 0,027 | 0,066 | 0,151 |

Table 12: Branch-and-Bound runtimes (in seconds) for instances with 30 jobs

| | | n = 20 | | | | n = 30 | | | |
|---|---|---|---|---|---|---|---|---|---|
| var. | LB | nodes | % AR | % NAR | % LB | nodes | % AR | % NAR | % LB |
| low | Lw | 9952 | 27,62 | 0,45 | 71,93 | 1454847 | 33,41 | 0,59 | 66,00 |
| | Lw DR | 8640 | 25,05 | 0,40 | 74,55 | 1040926 | 28,26 | 0,48 | 71,26 |
| | Lw AC | 8574 | 24,89 | 0,41 | 74,70 | 1109366 | 27,75 | 0,48 | 71,77 |
| | Lw AC DR | 8513 | 24,68 | 0,40 | 74,92 | 1109173 | 27,50 | 0,47 | 72,03 |
| high | Lw | 8706 | 30,77 | 0,05 | 69,18 | 785837 | 35,50 | 0,08 | 64,42 |
| | Lw DR | 6987 | 27,39 | 0,05 | 72,56 | 659586 | 30,03 | 0,07 | 69,90 |
| | Lw AC | 7061 | 27,36 | 0,05 | 72,59 | 654292 | 29,75 | 0,07 | 70,18 |
| | Lw AC DR | 6939 | 27,14 | 0,05 | 72,81 | 648324 | 29,48 | 0,06 | 70,45 |

Table 13: Average number of nodes and relative importance of the fathoming tests

| | | RDD | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.2 | | 0.4 | | 0.6 | | 0.8 | |
| var. | LF | nodes | % LB | nodes | % LB | nodes | % LB | nodes | % LB |
| low | 0.0 | 443 | 98,29 | 1375 | 96,79 | 6461 | 92,11 | 5697 | 94,53 |
| | 0.2 | 83853 | 34,33 | 9704 | 77,51 | 48404 | 81,73 | 43280 | 84,92 |
| | 0.4 | 14736255 | 16,10 | 2548859 | 31,28 | 1834030 | 55,53 | 538800 | 68,40 |
| | 0.6 | 2631359 | 22,57 | 1000330 | 35,63 | 638521 | 60,01 | 776971 | 69,98 |
| | 0.8 | 27185 | 49,53 | 4422 | 85,13 | 15337 | 82,88 | 16710 | 86,41 |
| | 1.0 | 261 | 99,28 | 927 | 97,97 | 3899 | 94,82 | 9139 | 94,44 |
| high | 0.0 | 616 | 97,86 | 1391 | 95,71 | 1836 | 93,63 | 5044 | 90,95 |
| | 0.2 | 60742 | 38,24 | 19385 | 75,54 | 13107 | 79,72 | 91145 | 80,52 |
| | 0.4 | 3735300 | 15,07 | 1669014 | 33,28 | 786980 | 56,36 | 3254559 | 63,59 |
| | 0.6 | 5021649 | 17,37 | 419549 | 40,97 | 495766 | 59,83 | 138991 | 70,79 |
| | 0.8 | 24856 | 49,88 | 17046 | 75,36 | 22784 | 79,93 | 42382 | 83,84 |
| | 1.0 | 430 | 97,92 | 911 | 95,45 | 1977 | 93,34 | 4594 | 92,36 |

Table 14: Nodes generated and percentage of nodes eliminated by lower bound test for Lw DR and instances with 30 jobs

instance size and decreases with the variability of the processing times. This result is to be expected, since it's more likely to find two jobs with the same processing time when the number of jobs is high and the variability of the processing times is low. As the instance size and the processing time and penalty variability increase, the percentage of nodes fathomed by the adjacent rule tends to increase, and the effectiveness of the lower bound test correspondingly decreases. We can also see that the percentage of nodes eliminated by the lower bound test is higher for the tighter lower bound versions. The $LF$ parameter has a significant effect on the number of nodes generated and the relative effectiveness of the node-fathoming tests. As the $LF$ value becomes closer to the middle of its range, the number of nodes generated increases, and the percentage of nodes eliminated by the lower bound test decreases. The importance of the adjacent rule becomes correspondingly higher, since the non-adjacent rule has only a marginal effect. These results can once more be explained by the much lower accuracy of the lower bounds for the intermediate $LF$ values.

# 7 Conclusion

In this paper we considered the lower bound procedures developed by Li and Liaw for the early/tardy problem with no idle time. These procedures use the multiplier adjustment method and require an initial sequence. We investigated the sensitiv-

ity of these lower bounds to the initial sequence, and experimented with different scheduling rules and dominance conditions. The computational results show that tighter lower bounds can be obtained through the use of better heuristics and dominance conditions. Liaw's lower bounds also outperform Li's, particularly when the lateness factor is 0.4 or 0.6. The most promising of the new lower bounds were incorporated in a branch-and-bound algorithm and compared with the best of the existing methods. The new lower bound that simply incorporates the dominance conditions allows for a reduction in computation time, particularly for the larger instances and when the processing time and penalty variability is high. The new versions also perform better than the existing one when the range of due dates is high.

# References


[1] Abdul-Razaq, T., and Potts, C. N. Dynamic programming state-space relaxation for single machine scheduling. *Journal of the Operational Research Society 39* (1988), 141–152.

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

[3] Jackson, J. R. Scheduling a production line to minimize maximum tardiness. Management Science Research Project, Research Report 43, University of California, Los Angeles, 1955.

[4] Korman, K. A pressing matter. *Video* (February 1994), 46–50.

[5] Landis, K. Group technology and cellular manufacturing in the westvaco los angeles vh department. Project report in iom 581, School of Business, University of Southern California, 1993.

[6] Lenstra, J. K., Rinnooy Kan, A. H. G., and Brucker, P. Complexity of machine scheduling problems. *Annals of Discrete Mathematics 1* (1977), 343–362.

[7] Li, G. Single machine earliness and tardiness scheduling. *European Journal of Operational Research 96* (1997), 546–558.

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

[9] Ow, P. S., and Morton, E. T. The single machine early/tardy problem. *Management Science 35* (1989), 177–191.

[10] Potts, C. N., and van Wassenhove, L. N. A branch-and-bound algorithm for the total weighted tardiness problem. *Operations Research 33* (1985), 363–377.

[11] Rachamadugu, R. M. V. A note on the weighted tardiness problem. *Operations Research 35* (1987), 450–452.

[12] Rachamadugu, R. V., and Morton, T. E. Myopic heuristics for the single machine weighted tardiness problem. Working Paper 28-81-82, Graduate School of Industrial Administration, Carnegie-Mellon University, 1981.

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

[14] Valente, J. M. S., and Alves, R. A. F. S. Improved heuristics for the early/tardy scheduling problem with no idle time. Working paper 126, Faculdade de Economia do Porto, Portugal, 2003.