WORKING PAPERS DA FEP

Investigação - Trabalhos em curso - nº 129, Maio de 2003

AN EXACT APPROACH TO EARLY/TARDY SCHEDULING WITH RELEASE DATES

Jorge M. S. Valente

Rui A. F. S. Alves

FACULDADE DE ECONOMIA

UNIVERSIDADE DO PORTO

www.fep.up.pt

An exact approach to early/tardy scheduling with release dates

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

May 29, 2003

Abstract

In this paper we consider the single machine earliness/tardiness scheduling problem with different release dates and no unforced idle time. The problem is decomposed into a weighted earliness subproblem and a weighted tardiness subproblem. Lower bounding procedures are proposed for each of these subproblems, and the lower bound for the original problem is then simply the sum of the lower bounds for the two subproblems. The lower bounds and several versions of a branch-and-bound algorithm are then tested on a set of randomly generated problems, and instances with up to 30 jobs are solved to optimality. To the best of our knowledge, this is the first exact approach for the early/tardy scheduling problem with release dates and no unforced idle time.

Keywords: scheduling, early/tardy, release dates, lower bounds, branch-and-bound

Resumo

Neste artigo é considerado um problema de sequenciamento com uma única máquina, custos de posse e de atraso e datas de disponibilidade distintas no qual não é permitida a existência de tempo morto não forçado. Este problema é decomposto num subproblema *weighted earliness* e num subproblema *weighted tardiness*. Procedimentos de *lower bound* são propostos para cada um destes subproblemas, e um *lower bound* para o problema original pode ser obtido somando os *lower bounds* dos dois subproblemas. Os *lower*

bounds e várias versões de um algoritmo *branch-and-bound* são testados num conjunto de problemas gerado aleatoriamente, tendo instâncias com até 30 trabalhos sido resolvidas de forma óptima.

Palavras-chave: sequenciamento, custos de posse e atraso, datas de disponibilidade, *lower bounds*, *branch-and-bound*

1 Introduction

In this paper we consider a single machine scheduling problem with release dates and 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 is assumed to be continuously available from time zero onwards and unforced machine idle time is not allowed. Job $J_j, j = 1, 2, \cdots, n$, becomes available for processing at its release date r_j, 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 minimises 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. It is assumed that no unforced machine idle time is allowed, so the machine is only idle if no job is currently available for processing. This assumption 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. Some specific examples of production settings with these characteristics are provided by Korman [6] and Landis [7]. The existence of different release dates is compatible with the assumption of no unforced idle time, as long as the forced idle time caused by the presence of distinct release dates is small or inexistent. If that is not the case, that

assumption becomes unrealistic, since it is then highly unlikely that either the machine idleness cost is higher than the early cost or the machine capacity is limited when compared with the demand.

As a generalization of weighted tardiness scheduling [8], the problem is strongly NP-hard. To the best of our knowledge, we know of no published work on this problem. The early/tardy problem with equal release dates and no idle time, however, has been considered by several authors, and both exact and heuristic approaches have been proposed. Among the exact approaches, branch-and-bound algorithms were presented by Abdul-Razaq and Potts [1], Li [9] and Liaw [10]. The lower bounding procedure of Abdul-Razaq and Potts was based on the subgradient optimization approach and the dynamic programming state-space relaxation technique, while Li and Liaw used Lagrangean relaxation and the multiplier adjustment method. Among the heuristics, Ow and Morton [11] developed several dispatch rules and a filtered beam search procedure. Valente and Alves [12] presented an additional dispatch rule and a greedy procedure, and also considered the use of dominance rules to further improve the schedule obtained by the heuristics. A neighbourhood search algorithm was also presented by Li [9]. The weighted tardiness problem with release dates has also been considered by Akturk and Ozdemir ([3], [2]). In [3] they present a dominance rule that is used as an improvement step after a dispatch heuristic has generated an initial schedule, and is also implemented in two local search heuristics, in order to guide them to the areas that will most likely contain the good solutions. In [2], Akturk and Ozdemir present some new dominance rules and two lower bounding procedures that are incorporated in a branch-and-bound algorithm.

In this paper we present a branch-and-bound algorithm based on a decomposition of the problem into a weighted earliness subproblem and a weighted tardiness subproblem. We propose lower bound procedures for each of these subproblems, and the lower bound for the original problem is then simply the sum of the lower bounds for the two subproblems. We also propose using two dominance rules originally derived for the problem with equal release dates in order to eliminate dominated nodes from the search tree. These rules can still be used in the presence of release dates provided a slight adjustment is made. Several versions of the branch-and-bound algorithm are then tested on a set of randomly generated problems with up to 30 jobs.

This paper is organized as follows. In section 2 we describe the decomposition of the problem and the derivation of the lower bound procedures. The dominance

rules are presented in section 3. Section 4 describes the implementation details of the branch-and-bound algorithm. The computational results are presented in section 5. Finally, conclusions are provided in section 6.

2 Decomposition of the problem and derivation of the lower bounds

In this section we first formulate the problem and decompose it into two subproblems with a simpler structure. This decomposition is similar to the one presented by Li [9] for the early/tardy problem with equal release dates. We then present two general lower bound procedures for each of the subproblems. Finally, we describe the specific procedures used to obtain the lower bound.

2.1 Decomposition of the problem

The early/tardy scheduling problem we consider can be formulated as problem (P):

$$V = \min \sum_{j=1}^{n} (h_j E_j + w_j T_j) \qquad (P)$$

s. t.

$$E_j \geq 0, j = 1, \ldots, n, \qquad (1)$$

$$E_j \geq d_j - C_j, j = 1, \ldots, n, \qquad (2)$$

$$T_j \geq 0, j = 1, \ldots, n, \qquad (3)$$

$$T_j \geq C_j - d_j, j = 1, \ldots, n, \qquad (4)$$

$$r_j \leq C_j - p_j, \qquad (5)$$

$$\text{machine capacity constraints}, \qquad (6)$$

where constraints (1)-(4) reflect the definitions of job earliness and tardiness, and constraint (5) specifies that no job can start before its release date. If we consider only the earliness costs or the tardiness costs in the objective function, it is possible to decompose problem (P) into two subproblems (P_1) and (P_2). Constraints (1) and (2) are only relevant to the subproblem with earliness costs, while constraints (3) and (4) are needed only when the tardiness costs are considered. The subproblems

(P_1) and (P_2) can be formulated as follows.

$$V_1 = \min \textstyle\sum_{j=1}^{n} h_j E_j \qquad (P_1)$$

s. t.

$$E_j \geq 0, j = 1, \ldots, n,$$

$$E_j \geq d_j - C_j, j = 1, \ldots, n,$$

$$r_j \leq C_j - p_j,$$

machine capacity constraints.

$$V_2 = \min \textstyle\sum_{j=1}^{n} w_j T_j \qquad (P_2)$$

s. t.

$$T_j \geq 0, j = 1, \ldots, n,$$

$$T_j \geq C_j - d_j, j = 1, \ldots, n,$$

$$r_j \leq C_j - p_j,$$

machine capacity constraints.

The motivation for this decomposition is twofold. First, subproblems (P_1) and (P_2) have a simpler structure than the original problem (P). Second, subproblem (P_2) is the weighted tardiness problem with release dates, for which a lower bounding procedure already exists. Given that unforced idle time is not allowed, subproblem (P_1) is symmetrical in structure to subproblem (P_2), so lower bounding procedures similar to those for (P_1) may be used. Nevertheless, (P_2) is NP-hard, since it is a generalization of weighted tardiness scheduling with equal release dates. Therefore, subproblem (P_1) can also be considered as NP-hard, given its symmetry in structure to (P_2). Even if this were not the case, solving $(P1)$ and (P_2) would not yield a direct solution to (P). So instead of directly solving the two subproblems, we will develop efficient lower bounding procedures for (P_1) and (P_2) in order to obtain a lower bound for (P).

Theorem 1 *$V_1^* + V_2^* \leq V^*$, where V_1^*, V_2^* and V^* are the minimum objective function values of (P_1), (P_2) and (P), respectively.*

Proof. Similar to the proof of Theorem 3.1 in [9]. ■

Theorem 2 *If L_1 and L_2 are lower bounds for problems (P_1) and (P_2), respectively, the $L_1 + L_2$ is a lower bound for problem (P).*

Proof. Similar to the proof of Lemma 3.1 in [9]. ■

2.2 Lower bound procedures for subproblem (P_1)

We will now present two lower bounding procedures for subproblem (P_1). The first procedure relaxes the assumption that a job cannot be scheduled before its release date and calculates a lower bound for a problem with equal release dates. The second procedure uses a lower bound for the weighted completion time problem with release dates.

Let S be a partial schedule (possibly empty) for problem (P_1) and U be the set of yet unscheduled jobs. Our objective is to obtain, for problem (P_1), a lower bound on the minimum cost of scheduling the jobs in U after the partial schedule S. Let $C^U_{\max}$ be the time at which the last job in U to be scheduled will be completed (since no unforced idle time is allowed, this time is sequence-independent) and $(V_1)^*$ denote the optimal objective function value of problem (P_1) on set U. Finally, let $s^1_U = C^U_{\max} - \sum_{j \in U} p_j$ and let $s^2_U = \max\left(C^S_{\max}, r^U_{\min}\right)$ denote the time at which the next job to be scheduled will start, where $C^S_{\max}$ is the completion time of the last job in S (0 if $S = \emptyset$) and $r^U_{\min} = \min\{r_j : J_j \in U\}$. The following propositions provide two lower bounds for subproblem (P_1).

Proposition 3 *Given a problem (P_1) on the set of unscheduled jobs U, let $\left(P_1'\right)$ be a new problem in which the release dates of all jobs in U are set equal to s^1_U. The following relation holds: $lb^e_1 \leq \left(V_1'\right)^* \leq (V_1)^*$, where $\left(V_1'\right)^*$ is the optimal objective function value of problem $\left(P_1'\right)$ and lb^e_1 is any lower bound for that problem.*

Proof. Any permutation of the jobs in U that is feasible for (P_1) is also feasible for $\left(P_1'\right)$. Also, the completion time of any job in such a permutation cannot be lower in $\left(P_1'\right)$ than it is in (P_1). Therefore, the weighted earliness of each job cannot then be higher in $\left(P_1'\right)$ than it is in (P_1), and we have $\left(V_1'\right)^* \leq (V_1)^*$. ■

Proposition 4 *Given a problem (P_1) on the set of unscheduled jobs U, let $lb\left(\sum\left[(-h_j)\,C_j\right]\right)$ be a lower bound for the weighted completion time problem with release dates $1|r_j|\sum\left[(-h_j)\,C_j\right]$ on set U and starting at time s^2_U. The following relation holds: $lb^e_2 \leq (V_1)^*$, where $lb^e_2 = \max\left(\sum h_j d_j - \left(-lb\left(\sum\left[(-h_j)\,C_j\right]\right)\right), 0\right)$.*

Proof. Clearly, $(V_1)^* \geq 0$ and $h_j d_j - h_j C_j \leq h_j E_j$. Let $(\sum h_j C_j)^*$ and $(\sum (-h_j) C_j)^*$ denote the optimum objective function values of the problems $1\,|r_j|\max \sum h_j C_j$ and $1|r_j|\sum (-h_j) C_j$, respectively. We then have $(V_1)^* \geq \sum h_j d_j - (\sum h_j C_j)^*$ and

$$\begin{aligned}\sum h_j d_j - \left(\sum h_j C_j\right)^* &= \sum h_j d_j - \left[-\left(\sum (-h_j) C_j\right)^*\right] \\ &\geq \sum h_j d_j - \left(-lb\left(\sum (-h_j) C_j\right)\right),\end{aligned}$$

which concludes the proof. ■

2.3 Lower bound procedures for subproblem (P_2)

For subproblem (P_2) we use two lower bounding procedures that were proposed in [2]. The first procedure relaxes the assumption that a job cannot be scheduled before its release date and calculates a lower bound for the problem with equal release dates, while the second uses a lower bound for the weighted completion time problem with release dates.

Let S be a partial schedule (possibly empty) for problem (P_2) and U be the set of yet unscheduled jobs. Our objective is to obtain, for problem (P_2), a lower bound on the minimum cost of scheduling the jobs in U after the partial schedule S. Let $s_U = \max\left(C^S_{\max}, r^U_{\min}\right)$ denote the time at which the next job to be scheduled will start, where $C^S_{\max}$ is the completion time of the last job in S (0 if $S = \emptyset$) and $r^U_{\min} = \min\{r_j : J_j \in U\}$. Also let $(V_2)^*$ denote the optimal objective function value of problem (P_2) on set U. The following propositions provide two lower bounds for subproblem (P_2).

Proposition 5 (Akturk and Ozdemir) *Given a problem (P_2) on the set of unscheduled jobs U, let $\left(P_2'\right)$ be a new problem in which the release dates of all jobs in U are set equal to s_U. The following relation holds: $lb_1^t \leq \left(V_2'\right)^* \leq (V_2)^*$, where $\left(V_2'\right)^*$ is the optimal objective function value of problem $\left(P_2'\right)$ and lb_1^t is any lower bound for that problem.*

Proposition 6 (Akturk and Ozdemir) *Given a problem (P_2) on the set of unscheduled jobs U, let $lb\left(\sum w_j C_j\right)$ be a lower bound for the weighted completion time problem with release dates $1|r_j|\sum w_j C_j$ on set U and starting at time s_U. The following relation holds: $lb_2^t \leq (V_2)^*$, where $lb_2^t = \max\left(lb\left(\sum w_j C_j\right) - \sum w_j d_j, 0\right)$.*

2.4 Lower bound procedures for problem (P)

The lower bound methods presented in the previous two subsections are general procedures. Lower bounds lb_1^e and lb_1^t can use any lower bound for the weighted earliness and weighted tardiness problems, respectively, while lower bounds lb_2^e and lb_2^t can use any lower bound for the weighted completion time problem with release dates. The lower bounds presented by Li [9] were used to calculate lb_1^e and lb_1^t. Hariri and Potts [5] and Belouadah, Posner and Potts [4] presented lower bounding procedures for the weighted completion time problem with release dates. We chose the latter lower bound, since preliminary tests indicated its computation time was lower and it provided better or equal results for nearly all of our test instances. The preliminary tests also indicated that the lower bounds for the problem with identical release dates usually provided better results than their weighted completion time problem counterparts. Based on these results, we decided to test four lower bounding procedures, denoted as $E2T2$, $E1T1$, $E2T1$ and $E1T2$. These lower bounds are calculated as follows: $E2T2 = \max(lb_1^e, lb_2^e) + \max(lb_1^t, lb_2^t)$; $E1T1 = lb_1^e + lb_1^t$; $E2T1 = \max(lb_1^e, lb_2^e) + lb_1^t$ and $E1T2 = lb_1^e + \max(lb_1^t, lb_2^t)$.

Theorem 7 *Let U be the set of yet unscheduled jobs and t be the current time. If $t \geq r_{\max}$, where $r_{\max}$ denotes the largest release date, we have $lb_1^e \geq lb_2^e$ and $lb_1^t \geq lb_2^t$.*

Proof. If $t \geq r_{\max}$, all unscheduled jobs are already available, and lb_2^t is then equal to $\sum w_j C_j^{WSPT} - \sum w_j d_j$, where C_j^{WLPT} is the completion time of job j when the jobs are scheduled in weighted shortest processing time order (see [4] for details concerning lb_2^t). Lower bound lb_1^t (see [9] for details) is obtained by solving the following problem

$$\begin{aligned} & \max \textstyle\sum_{j=1}^{n} \lambda_j \left(C_j^{WSPT} - d_j\right) \\ \text{s. t.} \\ & \tfrac{\lambda_j}{p_j} \geq \tfrac{\lambda_{j+1}}{p_{j+1}}, j = 1, \ldots, n, \\ & 0 \leq \lambda_j \leq w_j, j = 1, \ldots, n. \end{aligned}$$

Since w_j is a feasible value for λ_j, lower bound lb_1^t dominates lb_2^t. A similar reasoning can be used to show that $lb_1^e \geq lb_2^e$. ■

Based on this result, when the current time is greater than or equal to the largest release date only the lb_1^e and lb_1^t lower bounds are calculated, and therefore all the procedures become identical to the $E1T1$ lower bound.

3 Dominance rules

In this section we present the dominance rules that were used to reduce the number of nodes in the search tree. These rules were developed for the problem with identical release dates, but can still be used when the release dates are allowed to be different, provided care is taken to avoid making unfeasible job swaps. Ow and Morton [11] proved that in an optimal schedule all adjacent pairs of jobs J_i and J_j, with J_i preceding J_j, must satisfy the following condition:

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

with Ω_{xy} defined as

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

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.

Liaw [10] demonstrated that 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:

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

where Δ is the sum of the processing times of all jobs between J_i and J_j and Λ_{xy} is defined as

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

where s_x and t are defined as before.

When different release dates are allowed, the previous conditions must still be satisfied whenever $r_j \leq t$. When this is the case, J_i and J_j can be feasibly swapped, and the above rules must still apply.

4 Implementation of the branch-and-bound algorithm

In this section we briefly discuss the implementation of the branch-and-bound algorithm. We first calculate an upper bound on the optimum schedule cost. The EXP-ET dispatch rule originally developed in [11] for the problem with identical release dates is used to generate an initial sequence. The dominance rules described in the previous section are then applied to improve this sequence. First, the adjacent dominance rule of Ow and Morton is used. When a pair of adjacent jobs violates that rule, those jobs are swapped. This procedure is repeated until no improvement is found by the adjacent rule in a complete iteration. Then Liaw's non-adjacent rule is applied. Once again, if a pair of jobs violates the rule those jobs are swapped, and the procedure is repeated until no improvement is made in a complete iteration. The above two steps are repeated while the number of iterations performed by the non-adjacent rule is greater than one (i.e., while that rule detects an improvement). The upper bound value is updated whenever a feasible schedule with a lower cost is found during the branching process.

We use 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. 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 also use some tests to decide whether a node should be discarded or not. In one version of the branch-and-bound algorithm, three tests are used. In the first test, the adjacent dominance rule of Ow and Morton is applied to the two jobs most recently added to the node's partial schedule. In the second test, Liaw's non-adjacent rule 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. The non-adjacent rule is of more limited applicability, since it applies only to jobs with identical processing times, and the existence of different release dates can further limit its use. Therefore, we decided to test another version of the branch-and-bound algorithm that does not use the non adjacent rule, and only applies the other two tests. The branch-and-bound algorithms will be identified by the lower bound used, followed by "+N" when the non-adjacent rule dominance test is applied.

5 Computational results

In this section we present the results from the computational tests. A set of problems with 15, 20, 25, 30, 40, 50, 75, 100, 250, 500 and 1000 jobs was randomly 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 release date r_j was generated from the uniform distribution $\left[0, \alpha \sum_{j=1}^{n} p_j\right]$, where α was set at 0.25, 0.50 and 0.75. The maximum value of the range of release dates α was chosen so that the forced idle time would be small or inexistent. Preliminary tests showed that a higher value of 1.00 would lead to excessive amounts of forced idle time, which would be incompatible with the assumption that no unforced idle time may be inserted in a schedule. Instead of determining due dates directly, we generated slack times between a job's due date and its earliest possible completion time. For each job J_j, an integer due date slack s_j^d was generated from the uniform distribution $\left[0, \beta \sum_{j=1}^{n} p_j\right]$, where the due date slack range β was set at 0.10, 0.25 and 0.50. The due date d_j of J_j was then set equal to $d_j = (r_j + p_j) + s_j^d$. The values considered for each of the factors involved in the instance generation process are summarized in table 1. For each combination of instance size, processing time and penalty variability, α and β, 20 instances were randomly generated. All the algorithms were coded in Visual C++ 6.0 and executed on a Pentium IV-1500 personal computer. The lower bounds were calculated for all test instances, while the branch-and-bound algorithm was used to solve to optimality the instances with up to 30 jobs. Throughout this section, and in order to avoid excessively large tables, we will sometimes present results only for some representative cases.

Factors	Settings
Number of jobs	15, 20, 25, 30, 40, 50, 75, 100, 250, 500, 1000
Processing time and penalties variability	$[1, 10]$, $[1, 100]$
Range of release dates	0.25, 0.50, 0.75
Due date slack range	0.10, 0.25, 0.50

Table 1: Experimental design

In table 2 we present the average value of the lower bounds (avg) and the relative improvement (% imp) over the $E1T1$ lower bound, calculated as $\frac{LB - E1T1}{E1T1} * 100$, where LB and $E1T1$ represent the average value of the appropriate lower bound

and the $E1T1$ lower bound, respectively. Procedures $E2T2$ and $E1T2$ provide a noticeable increase in the lower bound value over $E1T1$ for the smaller instances, but that increase is negligible for larger problems. The improvement provided by the weighted completion time lower bound is usually higher for the tardiness subproblem, as can be seen from the results for lower bounds $E1T2$ and $E2T1$. The relative improvement over $E1T1$ decreases with the instance size and with the processing time and penalty variability.

		lower bound						
		E2T2		E1T2		E2T1		E1T1
var	n	avg	% imp	avg	% imp	avg	% imp	avg
low	15	351	8,82	348	7,69	326	1,12	323
	25	854	2,78	854	2,78	831	0,00	831
	50	3556	1,03	3544	0,68	3532	0,36	3520
	100	13438	0,47	13389	0,10	13424	0,36	13375
high	15	22818	3,00	22669	2,32	22304	0,67	22154
	25	65793	2,58	65753	2,51	64181	0,06	64141
	50	247567	0,79	247104	0,60	246094	0,19	245631
	100	973536	0,05	973536	0,05	973057	0,00	973057

Table 2: Lower bound values

In table 3 we present the average of the relative deviations from the optimum, calculated as $\frac{O-LB}{O} * 100$, where O and LB represent the optimum objective function value and the lower bound value, respectively. The α and β effect on the relative deviation from the optimum for the $E2T2$ lower bound is given in table 4. The lower bounds performance is poor, since on average they are 50% to 60% below the optimum. The performance is better when the processing time and penalty variability is low, and it improves as the instance size increases. We can see from table 4 that the lower bounds performance is adequate when α and β are both at their lowest value. The performance degrades as α and β increase, the only exception being the $(\alpha = 0.75, \beta = 0.50)$ parameter combination.

In table 5 we give the branch-and-bound average computation times (in seconds). In Table 6 we present the effect of α and β on the branch-and-bound runtimes for the 30 job instances and the $E1T1$+N and $E1T1$ branch-and-bound versions. The $E1T1$+N algorithm provides the best results. It can be seen that the increased accuracy of the lower bounding procedures that use the weighted completion time problem lower bounds is more than offset by their higher computational requirements. The non-adjacent rule should be used, as it usually leads to lower computation times,

		lower bound			
var	n	E2T2	E1T2	E2T1	E1T1
low	15	55,13	55,60	58,29	58,75
	20	54,37	54,37	56,73	56,73
	25	55,67	55,67	56,97	56,97
	30	52,63	52,82	53,79	53,98
high	15	63,23	63,43	63,99	64,19
	20	61,24	61,81	62,45	63,02
	25	60,62	60,64	61,62	61,64
	30	58,25	58,25	58,92	58,92

Table 3: Relative deviation from the optimum

		n = 20			n = 30		
var	Alfa	β=0.10	β=0.25	β=0.50	β=0.10	β=0.25	β=0.50
low	0.25	11,12	22,69	70,43	10,11	21,20	60,59
	0.50	27,41	53,96	86,38	22,22	55,97	88,96
	0.75	56,36	88,84	72,15	56,32	89,91	68,40
high	0.25	17,16	39,07	72,50	12,55	29,66	62,37
	0.50	30,39	69,93	92,56	33,11	64,13	89,16
	0.75	66,55	87,22	75,78	63,21	91,44	78,65

Table 4: Relative deviation from the optimum for lower bound E2T2

even for instances with high processing time and penalty variability (though the reduction in computation time is much lower in this case). For the larger instances, the computation time is higher when the processing time and penalty variability is high. The branch-and-bound computation times also increase significantly with α and β.

In Table 7 we present the average number of nodes generated by the branch-and-bound algorithm (nodes), as well as the average percentage of these nodes that were eliminated by the three fathoming tests (%elim). We also give some data on the relative importance of these tests, namely the average percentage of nodes eliminated by the lower bound (%lb), the adjacent rule (%adj) and, when appropriate, the non-adjacent rule (%non). In Table 8 we present the α and β 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 the $E1T1$ algorithm is used.

Only a very small percentage of nodes is eliminated by the non-adjacent rule. The proportion of nodes fathomed by this rule increases with the instance size and decreases with the variability of the processing times. This result is to be expected,

	var							
	low				high			
algorithm	n=15	n=20	n=25	n=30	n=15	n=20	n=25	n=30
E2T2+N	0,005	0,041	0,280	8,499	0,005	0,039	0,545	16,644
E2T2	0,006	0,043	0,326	13,000	0,006	0,039	0,549	16,775
E1T2+N	0,005	0,033	0,227	6,666	0,006	0,032	0,416	12,956
E1T2	0,005	0,037	0,263	10,190	0,004	0,032	0,419	13,133
E2T1+N	0,005	0,033	0,229	6,474	0,005	0,032	0,410	12,552
E2T1	0,005	0,036	0,261	9,818	0,005	0,031	0,411	12,624
E1T1+N	0,004	0,024	0,172	4,667	0,004	0,024	0,282	8,961
E1T1	0,004	0,026	0,201	6,885	0,004	0,026	0,283	8,960

Table 5: Branch-and-Bound runtimes (in seconds)

		algorithm					
		E1T1+N			E1T1		
var	Alfa	β=0.10	β=0.25	β=0.50	β=0.10	β=0.25	β=0.50
low	0.25	0,071	0,406	3,726	0,070	0,263	4,744
	0.50	0,231	0,385	7,270	0,260	0,412	8,844
	0.75	0,735	0,967	28,213	0,976	1,277	45,116
high	0.25	0,047	0,398	3,511	0,046	0,402	3,449
	0.50	0,124	0,312	28,688	0,121	0,305	28,256
	0.75	0,288	4,085	43,196	0,285	4,090	43,686

Table 6: Branch-and-Bound runtimes for 30 job instances

		n = 20					n = 30				
var	algorithm	nodes	%elim	%lb	%adj	%non	nodes	%elim	%lb	%adj	%non
low	E2T2+N	1655	83,2	74,3	24,9	0,7	318665	89,0	68,5	30,3	1,1
	E2T2	1760	83,2	75,5	24,5	–	479973	89,1	70,8	29,2	–
	E1T2+N	1655	83,2	74,3	24,9	0,7	318665	89,0	68,5	30,3	1,1
	E1T2	1760	83,2	75,5	24,5	–	479973	89,1	70,8	29,2	–
	E2T1+N	1663	83,2	74,1	25,1	0,7	319062	89,0	68,4	30,5	1,1
	E2T1	1770	83,2	75,3	24,7	–	480452	89,1	70,6	29,4	–
	E1T1+N	1663	83,2	74,1	25,1	0,7	319063	89,0	68,4	30,5	1,1
	E1T1	1770	83,2	75,3	24,7	–	480452	89,1	70,6	29,4	–
high	E2T2+N	1634	83,4	73,6	26,3	0,1	642652	89,0	67,4	32,5	0,1
	E2T2	1638	83,4	73,7	26,3	–	650950	89,0	67,6	32,4	–
	E1T2+N	1634	83,4	73,6	26,3	0,1	642652	89,0	67,4	32,5	0,1
	E1T2	1638	83,4	73,7	26,3	–	650950	89,0	67,6	32,4	–
	E2T1+N	1639	83,4	73,6	26,4	0,1	643440	89,0	67,2	32,7	0,1
	E2T1	1643	83,4	73,7	26,3	–	651788	89,0	67,5	32,5	–
	E1T1+N	1639	83,4	73,6	26,4	0,1	643440	89,0	67,2	32,7	0,1
	E1T1	1643	83,4	73,7	26,3	–	651788	89,0	67,5	32,5	–

Table 7: Average number of nodes and relative importance of the fathoming tests

		Beta					
		0.10		0.25		0.50	
var	Alfa	nodes	%lb	nodes	%lb	nodes	%lb
low	0.25	2320	87,06	9890	79,93	267901	64,87
	0.50	10734	77,00	18576	73,88	528312	69,32
	0.75	47210	64,30	71293	62,39	3367833	56,61
high	0.25	1653	82,24	15811	74,21	194890	61,26
	0.50	5207	74,65	14515	71,29	1762889	62,43
	0.75	16289	60,65	221520	63,43	3633314	57,01

Table 8: Nodes generated and importance of lower bound test for E1T1 and 30 job instances

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. For the larger instances, the number of nodes generated is much higher when the processing time and penalty variability is high. The number of nodes generated also increases with α and β. The proportion of nodes eliminated by the lower bound test usually decreases as α and β increase, and the importance of the adjacent rule becomes correspondingly higher, since even when the non-adjacent rule is used, it only has a marginal effect.

6 Conclusion

In this paper we considered the single machine earliness/tardiness scheduling problem with different release dates and no unforced idle time. This problem was decomposed into weighted earliness and weighted tardiness subproblems, and lower bounding procedures were presented for each of these subproblems. A lower bound for the original problem is then simply the sum of the lower bounds for the two subproblems. We also proposed using two dominance rules originally derived for the problem with equal release dates in order to eliminate dominated nodes from the search tree. These rules can still be used in the presence of release dates provided a slight adjustment is made. The lower bounds and several versions of a branch-and-bound algorithm were tested on a set of randomly generated problems, and instances with up to 30 jobs were solved to optimality.

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

[3] Akturk, M. S., and Ozdemir, D. A new dominance rule to minimize total weighted tardiness with unequal release dates. *European Journal of Operational Research 135* (2001), 394–412.

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

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

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

[7] 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.

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

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

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

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

[12] 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.