International Journal of Operations Research Vol. 2, No. 2, 9−16 (2005)

Improved Lower Bounds for the Single Machine Earliness/Tardiness Scheduling Problem with Release Dates

Jorge M. S. Valente[*]

Faculdade de Economia, Universidade do Porto, Rua Dr. Roberto Frias, 4200-464 Porto, Portugal

Received February 2005; *Revised June 2005*; *Accepted June 2005*

Abstract—In this paper, we consider the single machine earliness/tardiness scheduling problem with different release dates and no unforced idle time. The most effective lower bound uses multiplier adjustment procedures that require an initial sequence. We investigate the sensitivity of the lower bounding procedure to the initial sequences, and experiment with different scheduling rules and dominance conditions. The computational results show that it is possible to obtain improved lower bounds by using better initial sequences. The lower bounds are also incorporated in a branch-and-bound algorithm, and the new lower bounds were clearly superior for the larger instances. The new procedures were also much more consistent than the existing method, and the improvement they provided became larger as the instance difficulty increased.

Keywords—Scheduling earliness/tardiness, Release dates, Lower bounds, Branch-and-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 $\{J_1, J_2, \ldots, J_n\}$ has to be scheduled without preemption on a single machine that can handle only one job at a time. The machine is assumed to be continuously available from time zero onwards. Job J_j, $j = 1, \ldots, n$, 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. Given a schedule, the earliness of J_j is defined as $E_j = \max\{0, d_j - C_j\}$, while the tardiness of J_j can be defined as $T_j = \max\{0, C_j - d_j\}$, where C_j is the completion time of J_j. The objective is then to find a schedule that minimises the sum of weighted earliness and weighted tardiness $\sum_{j=1}^{n}(h_j E_j + w_j T_j)$ subject to the constraint that no unforced machine idle time is allowed, where h_j and w_j are the earliness and tardiness penalties of job J_j.

Scheduling models with both earliness and tardiness costs are compatible with the philosophy of just-in-time production, which emphasizes producing goods only when they are needed, since jobs are scheduled to complete as close as possible to their due dates. It is assumed that no unforced machine idle time is allowed, and therefore the machine is only idle when no jobs are available for processing. This assumption represents a type of production environment where the machine idleness cost is higher than the cost incurred by completing a job early, or the machine is heavily loaded, so it must be kept running in order to satisfy the demand. Korman (1994) and Landis (1993) give some specific examples of production settings with these characteristics. 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 distinct release dates is inexistent or quite small. If that is not the case, the assumption becomes unrealistic, since the machine capacity is then clearly not limited when compared with the demand and it is unlikely that the cost of the machine being kept idle is higher than the early cost.

As a generalization of weighted tardiness scheduling (Lenstra et al., 1977), the problem is strongly NP-hard. Several lower bounding procedures and a branch-and-bound algorithm were presented by Valente and Alves (2005a). The performance of various heuristics, including dispatching rules, a greedy procedure and a decision theory algorithm, was analysed in Valente and Alves (2003).

The earliness/tardiness problem with equal release dates and no idle time has also 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 (1988), Li (1997) and Liaw (1999). Valente and Alves (2005c) showed that using better initial sequences can improve the lower bounding procedures developed by Li and Liaw. Among the heuristics, Ow and Morton (1989) developed several earliness/tardiness dispatching rules and a filtered beam search procedure. Valente and Alves (2005b) presented an additional dispatching rule and a greedy procedure. A neighbourhood search algorithm was also presented by Li (1997). Recently, Hassin and Shani (2005) have generalized the earliness/tardiness problem by allowing the non-execution of tasks and introducing non-execution penalties. The weighted tardiness problem with release dates was also previously considered. A dominance rule and several heuristics were presented by Akturk and Ozdemir (2001), while Akturk and Ozdemir

[*] Corresponding author's email: jvalente@fep.up.pt

(2000) developed lower bounding procedures and a branch-and-bound algorithm.

The most effective lower bounding procedure presented in Valente and Alves (2005a) uses multiplier adjustment procedures that require an initial sequence. In this paper, we investigate the sensitivity of the lower bounding procedure to the initial sequence, in order to determine if it is possible to improve the lower bound through the use of better schedules. We experiment with different initial sequences, and analyse their effect on both the accuracy of the lower bound and their effectiveness in a branch-and-bound algorithm. Several heuristics are used to generate the initial sequences, and dominance rules are also applied to improve the schedules created by the heuristics.

The remainder of the paper is organized as follows. In section 2, we mention some previous results that provided the motivation for our current research. The heuristics and the dominance rules used to generate the initial sequences are presented in section 3. In section 4, we describe the specific lower bounding procedures. The implementation details of the branch-and-bound algorithm are discussed in section 5. In section 6, we present the computational results. Finally, some concluding remarks are given in section 7.

2. PREVIOUS RESULTS AND RESEARCH MOTIVATION

In this section, we succinctly review some previous results regarding lower bounds for the earliness/tardiness problem, particularly those that have motivated our present research. Valente and Alves (2005a) decomposed the earliness/tardiness problem with release dates into weighted earliness and weighted tardiness subproblems. This decomposition was motivated by the fact that a sum of lower bounds for the two subproblems is a lower bound for the original problem. Therefore, the lower bounds developed for the single machine weighted tardiness problem with release dates could directly be used for the tardiness subproblem. Also, these procedures could be modified and adapted in order to provide lower bounds for the weighted earliness subproblem. The weighted tardiness problem with release dates had been previously considered by Akturk and Ozdemir (2000), and two general lower bounding procedures were developed. The first of these procedures relaxed the assumption that a job could not be scheduled before its release date, and then calculated a lower bound for a problem with a modified common release date. The second procedure instead used a lower bound for the single machine weighted completion time problem with release dates. Valente and Alves (2005a) modified and adapted these lower bounds for the weighted earliness subproblem.

The two lower bounding procedures presented for each subproblem are general methods: the first procedure can use any lower bound for the weighted earliness and weighted tardiness problems with identical release dates, whereas the second can use any lower bound for the weighted completion time problem. Valente and Alves (2005a) used the weighted earliness and weighted tardiness lower bounds presented by Li (1997), and the weighted completion time lower bound proposed by Belouadah et al. (1992). Their computational tests showed that the earliness/tardiness lower bounds were more efficient and superior to their weighted completion time problem counterparts.

The multiplier adjustment procedures used in the earliness and tardiness lower bounds proposed by Li require an initial sequence. In Valente and Alves (2005a), these initial sequences were generated using the weighted shortest processing time (WSPT) and the weighted longest processing time (WLPT) rules, just as originally proposed by Li. However, Valente and Alves (2005c) analysed the earliness/tardiness problem with identical release dates, and showed that in this case the lower bounds proposed by Li are sensitive to the choice of initial schedule, and tighter lower bounds could then be obtained through the use of a better initial sequence. Therefore, in this paper we investigate the sensitivity of the earliness/tardiness lower bounds to the initial sequence in the context of the problem with different release dates, in order to determine if it is also possible to improve these lower bounds through the use of better schedules for problems where the release dates are allowed to differ. We experiment with different initial sequences, and analyse their effect on both the accuracy of the lower bounds and their effectiveness in a branch-and-bound algorithm. Several heuristics are used to generate the initial sequences, and dominance rules are also applied to improve the schedules created by the heuristics.

3. HEURISTICS AND DOMINANCE RULES

In this section, we describe the several dispatching heuristics and dominance rules that were used to generate initial sequences for the lower bounding procedures. The WSPT rule was presented by Smith (1956) and sorts the jobs in non-increasing order of w_j/p_j. The WLPT rule, also proposed by Smith, sorts the jobs in non-increasing order of p_j/h_j. Since these two dispatching rules only require sorting, their time complexity is $O(n\log n)$.

The Apparent Tardiness Cost (ATC) dispatching procedure was developed by Rachamadugu and Morton (1981) for the single machine weighted tardiness problem. This heuristic selects, whenever the machine becomes available, the unscheduled job with the highest priority index $(w_j/p_j)\exp(-(d_j-t-p_j)^+/k\overline{p})$, where $\overline{p}$ is the average processing time, t is the current time and k is a lookahead empirical parameter.

The Apparent Earliness Cost (AEC) heuristic was presented by Valente and Alves (2005c) and 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

$(h_j / p_j)\exp(-(t-d_j)^+ / k\overline{p})$, 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 time complexity of both the ATC and AEC heuristics is $O(n^2)$.

Two dominance rules were also used to improve the sequences generated by these heuristics. These rules identify a condition that holds for adjacent jobs in an optimal sequence. Rachamadugu (1987) proved that for 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 those adjacent jobs:

$$(w_i / p_i)(1-(d_i - t - p_i)^+ / p_j) \geq (w_j / p_j)(1-(d_j - t - p_j)^+ / p_i)$$

In this expression, i denotes the index of the job in the ith position, j is the index of the job in the $(i+1)$st and t is the current available time. 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.

Valente and Alves (2005c) presented a dominance rule for the weighted earliness problem with no idle time allowed. This rule is an adaptation of the weighted tardiness dominance condition and is symmetric to the rule developed by Rachamadugu. For 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 those adjacent jobs:

$$(h_i / p_i)(1-(t + p_i + p_j - d_i)^+ / p_j) \leq (h_j / p_j)(1-(t + p_i + p_j - d_j)^+ / p_i)$$

In this expression, i, j and t are as previously defined. 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 tardy in either position.

4. THE LOWER BOUNDS

In this section, we describe the four lower bounds that were considered. These lower bounds use only the procedures that relax the assumption that jobs cannot be scheduled before their release dates and calculate a lower bound for a problem with a modified identical release date. The lower bound denoted by WPT uses the WLPT and WSPT rules to generate the initial sequences for the weighted earliness and weighted tardiness subproblems, respectively. This procedure was previously considered in Valente and Alves (2005a). The lower bound values obtained with this method were most often quite similar to those given by procedures that additionally used the weighted completion time problem bounds. Also, this procedure was the most effective of those analysed by Valente and Alves when incorporated in a branch-and-bound algorithm.

Lower bound WPTDR uses these same heuristics and then applies the dominance rules presented in the previous section to improve the sequence generated by the heuristics. Rachamadugu's rule is used for the weighted tardiness subproblem and the rule presented by Valente and Alves (2005c) is used for the weighted earliness subproblem. Each rule iteration considers in succession all adjacent job positions and when a pair of adjacent jobs in a sequence violates a rule, those jobs are swapped if that change reduces the objective function value. This procedure is applied repeatedly until it converges and no improvement is found in a complete iteration (i.e., until the sequence is locally optimal and it cannot be further improved through adjacent swaps). The complexity of the dominance rules is $O(n)$ per iteration, and the total complexity depends on the number of times a rule iteration produces an improvement. The lower bound AC uses the AEC (ATC) heuristic for the earliness (tardiness) subproblem. Lower bound ACDR uses these same heuristics, but also applies the dominance rules, just as previously described for the WPTDR lower bound.

5. IMPLEMENTATION OF THE BRANCH-AND-BOUND ALGORITHM

In this section, we discuss the implementation details of the branch-and-bound algorithm. 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. Two dominance rules are 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 (1989) presented a condition that must be satisfied for all adjacent pairs of jobs in an optimal schedule. The dominance rule developed by Liaw (1999), on the other hand, applies to non–adjacent pairs of jobs with identical processing times.

In the first fathoming test, the rule presented by 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. During the initialization, the algorithm checks if at least two jobs have identical processing times and this second test is skipped when all P_js are different. 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 initial upper bound on the optimum schedule cost is calculated using the best of the procedures presented in Valente and Alves (2003). The decision theory local search heuristic is first used to generate an initial sequence, and the dominance rules of Ow and Morton and Liaw are then applied to improve this sequence (see Valente and Alves (2003) for details). The upper bound value is updated whenever a feasible schedule with a lower cost is found during the branching process.

The configuration just described is used consistently in all branch-and-bound procedures. Therefore, the only difference between the branch-and-bound algorithms is in the choice of the lower bounding procedure.

6. 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, 200, 250, 300, 400, 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 $\left[0, R\sum_{j=1}^{n} P_j\right]$, where R was set at 0.25, 0.50 and 0.75. The maximum value of the range of release dates R was chosen so that the forced idle time would be small or inexistent. Preliminary tests showed that $R = 1.00$ 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 $\left[0, D\sum_{j=1}^{n} P_j\right]$, where the due date slack range D was set at 0.10, 0.25 and 0.50. The due date d_j of J_j was then set equal to $d_j = r_j + p_j + s_j^d$. For each combination of instance size n, processing time and penalty variability (var), R and D, 50 instances were randomly generated. Therefore, 450 instances were generated for each (var, n) combination. All the algorithms were coded in Visual C++ 6.0 and executed on a Pentium IV – 1700 Mhz 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.

In table 1, we present the average value of the lower bounds (mean) and the relative improvement (%imp) over the WPT lower bound, calculated as (LB - WPT) / WPT × 100, where LB and WPT represent the average value of the appropriate lower bound (i.e., AC, ACDR or WPTDR) and the WPT lower bound, respectively. A test was also performed 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 the hypotheses of the paired-samples t-test were not all met, the non-parametric Wilcoxon test was selected. The significance values of this test, i.e., the level of significance values above which the equal distribution hypothesis is to be rejected, were always equal to 0.000

Table 1. Lower bound values

		lower bound						
		AC		ACDR		WPTDR		WPT
var	*n*	mean	%imp	mean	%imp	mean	%imp	mean
low	15	335	1.46	336	1.67	334	1.09	330
	25	867	1.21	869	1.45	865	0.98	856
	50	3456	0.98	3465	1.26	3442	0.59	3422
	100	13657	0.96	13711	1.36	13601	0.55	13526
	250	84394	0.96	84743	1.37	83902	0.37	83594
	500	341768	0.97	343000	1.34	339736	0.37	338475
	1000	1351564	0.96	1355813	1.28	1342228	0.26	1338704
high	15	23147	0.99	23192	1.19	23132	0.93	22920
	25	62510	1.32	62743	1.69	62398	1.14	61698
	50	242669	1.32	243820	1.80	242166	1.11	239510
	100	976446	1.18	981502	1.70	972720	0.79	965090
	250	5994784	1.14	6031518	1.76	5964738	0.63	5927256
	500	24017992	1.07	24149732	1.63	23884579	0.51	23763427
	1000	96056715	1.15	96548228	1.67	95427634	0.49	94963472

Table 2. Relative deviation from the optimum

		lower bound			
var	*n*	AC	ACDR	WPT	WPTDR
low	15	57.59	57.45	58.47	57.87
	20	55.37	55.22	55.94	55.60
	25	55.78	55.62	56.47	55.92
	30	53.68	53.50	54.54	53.91
high	15	64.28	64.14	64.83	64.35
	20	61.76	61.56	62.57	61.90
	25	60.41	60.19	61.05	60.48
	30	58.56	58.30	59.57	58.85

Table 3. Relative deviation from the optimum for lower bound WPTDR

		$n = 20$			$n = 30$		
var	*R*	$D = 0.10$	$D = 0.25$	$D = 0.50$	$D = 0.10$	$D = 0.25$	$D = 0.50$
low	0.25	13.55	26.31	65.94	10.79	22.65	58.51
	0.50	30.60	56.42	87.79	29.64	52.97	86.90
	0.75	62.86	90.08	66.89	67.42	88.61	67.65
high	0.25	21.56	34.74	70.29	15.01	29.05	64.42
	0.50	35.38	65.42	91.42	31.43	58.86	93.96
	0.75	76.17	88.31	73.84	71.68	92.92	72.30

From these results, we can conclude that the lower bounding procedure is sensitive to the choice of initial sequence. Tighter lower bound values can be obtained through the use of better initial sequences, obtained with more sophisticated heuristics and/or dominance rules. In fact, the lower bounds that use the AEC/ATC dispatching heuristics outperform those that use the simple WLPT/WSPT rules. The use of the dominance rules also improves the lower bound value, since the ACDR and WPTDR bounds provide better results than the AC and WPT procedures, respectively. The improvement of lower bound WPTDR over WPT is larger than that of ACDR over AC. Therefore, the improvement provided by the dominance rules is higher when the less sophisticated WLPT/WSPT heuristics are used. The Wilcoxon test values also indicate that the differences in distribution between the lower bounding procedures are statistically significant. The WPTDR lower bound is up to 1% above the WPT procedure, while the AC and ACDR lower bounds provide a 1-2% improvement. The relative improvement is usually higher for instances with a high processing time and penalty variability, and tends to decrease with the instance size for the WPTDR lower bound.

In table 2, we present the average of the relative deviations from the optimum, calculated as $(O - LB) / O \times 100$, where O and LB represent the optimum objective function value and the lower bound value (i.e., AC, ACDR, WPT or WPTDR), respectively. The *R* and *D* effect on the relative deviation from the optimum for the WPTDR lower bound is given in table 3. 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. The lower bounds performance is adequate when *R* and *D* are both at their lowest value, and it deteriorates considerably as *R* and *D* increase (the only exception being the ($R = 0.75$, $D = 0.50$) parameter combination). This result is to be expected, since the earliness/tardiness problem lower bounds used in these procedures should be more accurate for problems with small release date and due date ranges. The earliness/tardiness lower bounds were developed for the problem with identical release dates, so their performance is better when the release dates are only slightly scattered. Also, most jobs will likely be tardy when *D* is low, and as *D* increases there will be a greater balance between the number of early and tardy jobs. Previous research on the problem with identical release dates has shown that the earliness/tardiness lower bounds perform better when most jobs are indeed tardy (or early). When the number of tardy and early jobs is similar, the problem is much harder, and the lower bounds become more inaccurate.

In Table 4, we present the lower bounds average runtimes, in seconds, for instances with 500 and 1000 jobs. The procedures that use more sophisticated heuristics, and/or dominance rules, require higher computation times. Therefore, it cannot be guaranteed that they 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, instances with up to 30 jobs were solved to optimality with a branch-and-bound algorithm.

Table 4. Lower bound runtimes (in seconds)

	low var		high var	
LB	n = 500	n = 1000	n = 500	n = 1000
AC	0.039	0.153	0.039	0.153
ACDR	0.043	0.172	0.044	0.175
WPT	0.001	0.002	0.002	0.003
WPTDR	0.008	0.024	0.009	0.028

Table 5. Branch-and-bound runtimes (in seconds)

		lower bound			
var	n	AC	ACDR	WPT	WPTDR
low	15	0.004	0.005	0.003	0.004
	20	0.028	0.034	0.025	0.023
	25	0.153	0.177	0.227	0.122
	30	1.348	1.564	10.089	1.421
high	15	0.004	0.004	0.003	0.004
	20	0.022	0.026	0.021	0.018
	25	0.207	0.241	0.622	0.165
	30	1.151	1.316	23.829	1.045

Table 6. Branch-and-bound runtimes (in seconds) for 30 job instances

var	LB	min	p50	p75	p95	max	cov
low	AC	0.000	0.188	0.656	3.782	84.718	447.6
	ACDR	0.000	0.219	0.750	4.281	100.406	451.8
	WPT	0.000	0.188	0.890	11.297	1002.220	835.6
	WPTDR	0.000	0.141	0.563	3.297	146.437	576.7
high	AC	0.000	0.219	0.797	4.093	53.734	362.8
	ACDR	0.000	0.235	0.843	4.843	61.703	369.5
	WPT	0.000	0.219	1.062	11.985	7081.840	1492.7
	WPTDR	0.000	0.156	0.610	3.578	41.219	378.3

Table 7. Effect of R and D parameters on the branch-and-bound runtimes for 30 job instances

		lower bound					
		WPT			WPTDR		
var	R	D = 0.10	D = 0.25	D = 0.50	D = 0.10	D = 0.25	D = 0.50
low	0.25	0.035	0.168	2.761	0.037	0.165	2.989
	0.50	0.334	0.261	23.748	0.178	0.225	6.118
	0.75	0.346	2.078	61.073	0.214	1.125	1.737
high	0.25	0.060	0.217	2.477	0.053	0.199	2.542
	0.50	0.166	0.415	4.461	0.115	0.324	2.907
	0.75	1.031	0.881	204.754	0.296	0.429	2.540

In table 5, we give the branch-and-bound average computation times, in seconds. In table 6, we present several additional statistics for the computation times on instances with 30 jobs, namely the minimum (min) and maximum (max) values, the coefficient of variation (cov) and the percentiles 50, 75 and 95 (p50, p75 and p95, respectively) of the distribution of the runtimes. The effect of the R and D parameters on the branch-and-bound runtimes for instances with 30 jobs is given in table 7. The computation times are quite similar for all the branch-and-bound algorithms for the smaller problems with 15 or 20 jobs. The difference in the runtimes becomes much clearer for the larger instances (n = 25, 30). For these instances, all the three new lower bounds provide much better results and are clearly superior to the existing procedure. Therefore, the higher computational requirements of the new procedures are more than offset by their increased accuracy. The branch-and-bound runtimes for the three new procedures are quite close, although the best results are usually given by the WPTDR lower bound.

The results in table 6 once again show that the new procedures outperform the existing lower bound. Furthermore, these results show that the new procedures are clearly more consistent, and the improvement they provide becomes larger as the instance difficulty increases. For the easier instances, which require low computation times, the runtimes are quite close for all procedures, as can be seen by the minimum and percentile 50 values. As the instance difficulty, and correspondingly the computation time, increase, the new procedures become increasingly more efficient than the existing lower bound, since the increase in runtime is much slower for the new procedures. For the most difficult instances, as can be seen by the percentile 95 and maximum runtime values, the computation times are substantially lower for the new lower bounds. The coefficient of variation values also indicate that the new procedures are considerably more consistent, since the variability in their runtimes is significantly lower. The branch-and-bound runtimes also usually increase with both *R* and *D*.

Table 8. Average number of nodes and relative importance of the fathoming tests

		n = 20					n = 30				
var	LB	NG	%EL	%LB	%A	%NA	NG	%EL	%LB	%A	%NA
low	AC	997	82.65	76.22	23.08	0.70	34546	88.04	72.16	27.04	0.81
	ACDR	971	82.60	76.41	22.90	0.69	32345	87.96	72.45	26.75	0.80
	WPT	1637	83.37	72.83	26.41	0.76	777700	88.63	68.52	30.52	0.97
	WPTDR	1029	82.71	75.90	23.42	0.69	48849	88.10	71.66	27.52	0.82
high	AC	802	82.19	74.92	25.00	0.08	28067	88.01	70.87	29.04	0.09
	ACDR	777	82.13	75.15	24.77	0.08	25620	87.91	71.32	28.59	0.09
	WPT	1370	82.78	72.16	27.75	0.09	2112480	88.60	67.30	32.58	0.11
	WPTDR	825	82.20	74.75	25.17	0.08	35286	88.03	70.57	29.33	0.10

Table 9. Nodes generated and importance of lower bound test for the WPTDR lower bound and 30 job instances

		D = 0.10		D = 0.25		D = 0.50	
var	R	NG	%LB	NG	%LB	NG	%LB
low	0.25	895	86.57	4639	75.98	95328	67.78
	0.50	5200	80.97	6878	73.34	221714	67.36
	0.75	7450	67.43	41185	64.29	56349	61.22
high	0.25	1327	84.25	5877	75.01	85467	66.47
	0.50	3321	79.23	10811	69.95	97747	67.20
	0.75	10341	69.14	15185	64.70	87495	59.20

In table 8, we present the average number of nodes generated by the branch-and-bound algorithm (NG), as well as the average percentage of these nodes that were eliminated by the three fathoming tests (%EL). 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 (%A) and the non-adjacent rule (%NA). In table 9, we present the *R* and *D* 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 WPTDR lower bound is used.

The proportion of nodes eliminated by the lower bound test is higher for the new and more accurate lower bounds, while the number of nodes generated is much lower. Only a very small percentage of nodes is eliminated by the non-adjacent rule. This result is most likely somewhat influenced by the order in which the two rules are applied, since the adjacent rule can eliminate nodes that would otherwise be fathomed by the non-adjacent dominance condition. The adjacent dominance rule, however, requires a much lower computational effort, and it's therefore more efficient to apply it before checking the non-adjacent rule. The proportion of nodes fathomed by the non-adjacent rule decreases with the variability of the processing times and increases with the instance size. 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 processing time variability 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. The number of nodes generated is also usually lower when the variability is high. The number of nodes usually increases with *R* and *D*, the only exception being the (R = 0.75, D = 0.50) parameter combination. The proportion of nodes fathomed by the lower bound test decreases with both *R* and D, and the importance of the adjacent rule becomes correspondingly higher, since the non-adjacent rule has only a marginal effect.

The number of nodes generated, and correspondingly the runtimes, tend to increase with the release date and due date ranges. Therefore, the problem usually becomes harder to solve with a branch-and-bound algorithm as R and D increase. As we previously remarked, there is a greater balance between the number of early and tardy jobs as D increases, so the problem should indeed be harder. An increase in the range of release dates, on the other hand, reduces the number of feasible schedules. This reduction is not very substantial, however, since we have restricted the value of R in order to avoid unforced idle time, and is more than compensated by the deterioration in the performance of the lower bound fathoming test. In fact, as we previously mentioned, the relative importance of the lower bound elimination test steadily decreases as both R and D increase.

7. CONCLUSIONS

In this paper, we considered the most effective lower bounding procedure for the single machine earliness/tardiness problem with release dates and no unforced idle time. This lower bounding method uses multiplier adjustment procedures that require an initial sequence. We investigated the sensitivity of the lower bounding procedure to the initial sequences, and experimented with different scheduling rules and dominance conditions.

The computational results show that the lower bounding procedure is sensitive to the choice of initial schedule, and tighter lower bound values can indeed be obtained through the use of better initial sequences. The new bounding procedures that use more sophisticated heuristics, and/or dominance rules, provided better lower bound values than the previously existing method. We also analysed the effectiveness of the lower bounding procedure when it is incorporated in a branch-and-bound algorithm. For the larger instances, all the new lower bounds provided much better results and were clearly superior to the existing procedure. Furthermore, the computational results showed that the new procedures are clearly more consistent, and the improvement they provide becomes larger as the instance difficulty increases.

ACKNOWLEDGEMENTS

The author would like to thank the anonymous referees for several comments that were used to improve this paper.

REFERENCES

1. Abdul-Razaq, T. and Potts, C.N. (1988). Dynamic programming state-space relaxation for single machine scheduling. *Journal of the Operational Research Society*, 39: 141-152.
2. Akturk, M.S. and Ozdemir, D. (2000). An exact approach to minimizing total weighted tardiness with release dates. *IIE Transactions*, 32: 1091-1101.
3. Akturk, M.S. and Ozdemir, D. (2001). A new dominance rule to minimize total weighted tardiness with unequal release dates. *European Journal of Operational Research*, 135: 394-412.
4. Belouadah, H., Posner, M.E., and Potts, C.N. (1992). Scheduling with release dates on a single machine to minimize total weighted completion time. *Discrete Applied Mathematics*, 36: 213-231.
5. Hassin, R. and Shani, M. (2005). Machine scheduling with earliness, tardiness and non-execution penalties. *Computers and Operations Research*, 32: 683-705.
6. Korman, K. (1994). A pressing matter. *Video*, February: 46-50.
7. Landis, K. (1993). *Group technology and cellular manufacturing in the Westvaco Los Angeles VH department.* Project Report in IOM 581, School of Business, University of Southern California, USA.
8. Lenstra, J.K., Rinnooy Kan, A.H.G., and Brucker, P. (1977). Complexity of machine scheduling problems. Annals *of Discrete Mathematics*, 1: 343-362.
9. Li, G. (1997). Single machine earliness and tardiness scheduling. *European Journal of Operational Research*, 96: 546-558.
10. Liaw, C.F. (1999). A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. *Computers and Operations Research*, 26: 679-693.
11. Ow, P.S. and Morton, T.E. (1989). The single machine early/tardy problem. *Management Science*, 35: 177-191.
12. Potts, C.N. and van Wassenhove, L. N. (1985). A branch-and-bound algorithm for the total weighted tardiness problem. *Operations Research*, 33: 363-377.
13. Rachamadugu, R.M.V. (1987). A note on the weighted tardiness problem. *Operations Research*, 35: 450-452.
14. Rachamadugu, R.M.V. and Morton, T.E. (1981). *Myopic Heuristics for the Single Machine Weighted Tardiness Problem.* Working Paper 28-81-82, Graduate School of Industrial Administration, Carnegie-Mellon University.
15. Smith, W.E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3: 59-66.
16. Valente, J.M.S. and Alves, R.A.F.S. (2003). *Heuristics for the Early/Tardy Scheduling Problem with Release Dates.* Working Paper 130, Faculdade de Economia da Universidade do Porto, Portugal.
17. Valente, J.M.S. and Alves, R.A.F.S. (2005a). An exact approach to early/tardy scheduling with release dates. *Computers and Operations Research*, 32: 2905-2917.
18. Valente, J.M.S. and Alves, R.A.F. S. (2005b). Improved heuristics for the early/tardy scheduling problem with no idle time. *Computers and Operations Research*, 32: 557-569.
19. Valente, J.M.S. and Alves, R.A.F.S. (2005c). Improved lower bounds for the early/tardy scheduling problem with no idle time. *Journal of the Operational Research Society*, 56: 604-612.