WORKING PAPERS DA FEP

Investigação - Trabalhos em curso - nº 130, Maio de 2003

HEURISTICS FOR THE EARLY/TARDY SCHEDULING PROBLEM WITH RELEASE DATES

Jorge M. S. Valente

Rui A. F. S. Alves

FACULDADE DE ECONOMIA

UNIVERSIDADE DO PORTO

www.fep.up.pt

Heuristics for the early/tardy scheduling problem 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. We analyse the performance of several dispatch rules, a greedy procedure and a decision theory local search heuristic. The dispatch rules use a lookahead parameter whose value must be specified. We perform some experiments to determine an appropriate value for this parameter. The use of dominance rules to improve the solutions obtained by these heuristics is also considered. The computational results show that the use of the dominance rules can indeed improve the solution quality with little additional computational effort. To the best of our knowledge, this is the first analysis of heuristic performance for the early/tardy scheduling problem with release dates and no unforced idle time.

Keywords: scheduling, early/tardy, release dates, heuristics

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. O desempenho de várias heurísticas é analisado. Estas heurísticas incluem diversas *dispatch rules*, um procedimento *greedy* e uma heurística do tipo de pesquisa local combinada com teoria da decisão. As *dispatch rules* utilizam um

parâmetro *lookahead* cujo valor é necessário especificar. Alguns testes foram efectuados para determinar um valor apropriado para este parâmetro. A utilização de regras de dominância para melhorar as soluções obtidas pelas heurísticas for também considerada. Os resultados computacionais mostram que a utilização das regras de dominância permite de facto melhorar a qualidade das soluções com um reduzido esforço computacional.

Palavras-chave: sequenciamento, custos de posse e atraso, datas de disponibilidade, heuristicas

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 [5] and Landis [6]. 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 [7], the problem is strongly NP-hard. To the best of our knowledge, the only work in this problem is due to Valente and Alves [11]. They presented a branch-and-bound algorithm based on a decomposition of the problem into weighted earliness and weighted tardiness subproblems. Two lower bound procedures were presented for each subproblem, and the lower bound for the original problem is then simply the sum of the lower bounds for the two subproblems. 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 [8] and Liaw [9]. 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 [10] 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 [8]. 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 analyse the performance of several heuristics. We consider some heuristics originally presented for the problem with equal release dates, namely two dispatch rules developed in [10] and the dispatch rule and the greedy procedure proposed in [12]. We also consider a local search heuristic procedure based on the decision theory approach of Kanet and Zhou [4], though it can also be called a beam search algorithm with a beam width of one. The dispatch rules include a

lookahead parameter whose value must be specified. We perform some experiments to determine appropriate values for the lookahead parameter. We also consider the use of some dominance rules developed for the problem with equal release dates to improve the solution obtained by these heuristics. The computational results show that the use of the dominance rules can indeed improve the solution quality with little additional computational effort.

This paper is organized as follows. The heuristics are described in section 2. In section 3 we present the dominance rules that were used to improve the schedule obtained by the heuristics. The computational results are given in section 4. Finally, conclusions are provided in section 5.

2 The heuristics

In this section we describe the heuristics that were considered. The LIN-ET heuristic is one of the dispatch rules developed by Ow and Morton [10]. This heuristic uses the following priority index $I_j(t)$ to determine the job J_j to be scheduled at any instant t when both the machine and at least one unscheduled job are available:

$$I_j(t) = \begin{cases} W_j & \text{if} \quad s_j \leq 0 \\ W_j - \frac{s_j(H_j+W_j)}{k\overline{p}} & \text{if} \quad 0 \leq s_j \leq k\overline{p} \\ -H_j & \text{otherwise,} \end{cases}$$

where $W_j = w_j/p_j$, $H_j = h_j/p_j$, $s_j = d_j - t - p_j$ is the slack of job J_j at time t, $\overline{p}$ is the average processing time and k is a lookahead parameter. The EXP-ET dispatch rule was also presented by Ow and Morton, and uses the following priority index $I_j(t)$:

$$I_j(t) = \begin{cases} W_j & \text{if} \quad s_j \leq 0 \\ W_j \exp\left[-\frac{(H_j+W_j)}{H_j}(s_j/k\overline{p})\right] & \text{if} \quad 0 \leq s_j \leq \frac{W_j}{H_j+W_j}k\overline{p} \\ H_j^{-2}\left[W_j - \frac{(H_j+W_j)s_j}{k\overline{p}}\right]^3 & \text{if} \quad \frac{W_j}{H_j+W_j}k\overline{p} \leq s_j \leq k\overline{p} \\ -H_j & \text{otherwise,} \end{cases}$$

where W_j, H_j, s_j, $\overline{p}$ and k are as previously defined. The last dispatch rule, denoted by WPT-MS, was proposed by Valente and Alves [12] and uses the priority index:

$$I_j(t) = \begin{cases} W_j & \text{if} \quad s_j \leq 1 \\ W_j/s_j & \text{if} \quad 1 \leq s_j \leq \frac{W_j}{H_j+W_j} k\overline{p} \\ -H_j \left[1 - (k\overline{p} - s_j) / \left(k\overline{p} - \frac{W_j}{H_j+W_j} k\overline{p}\right)\right]^2 & \text{if} \quad \frac{W_j}{H_j+W_j} k\overline{p} \leq s_j \leq k\overline{p} \\ -H_j & \text{otherwise,} \end{cases}$$

where once more W_j, H_j, s_j, $\overline{p}$ and k are as previously defined. The dispatch rules use a priority function that starts at $-H_j$ when jobs are in no danger of being tardy ($s_j \geq k\overline{p}$), and then gradually increases to a maximum of W_j when jobs are on time or late ($s_j \leq 0$). Therefore, the rules reflects a priority that focuses on the tardiness cost of a job as its slack becomes small, while the earliness cost dominates when that slack is large. The dispatch heuristics differ only in the calculation of the job priorities for the intermediate values of the job slack. The choice of the lookahead parameter k should reflect the average number of jobs that may clash in the future each time a sequencing decision is to be made. The time complexity of the dispatch rules is $O(n^2)$.

The Greedy heuristic was presented by Valente and Alves, and can be described as follows. Let c_{xy}, with $x \neq y$, be the combined cost of scheduling jobs J_x and J_y, in this order, in the next two positions in the sequence. Let t be the current time, u be the number of yet unscheduled jobs, L a list with the indexes of those jobs and $P(j)$ the priority of job J_j. Also let L' be a list with the indexes of the unscheduled jobs that are available at time t. The steps of the heuristic are:

Step 1: Initialize $t = \min\{r_j : j = 1, \ldots, n\}$, $u = n$ and $L = \{1, 2, \ldots, n\}$.

Step 2: Determine $L' = \{j : j \in L \wedge r_j \leq t\}$.

Step 3: If $L' = \emptyset$, set $t = \min r_j, j \in L$ and go to step 2.

Step 4: If $\left|L'\right| = 1$

set $t = t + p_j, j \in L'$;

set $L = L \setminus \{j\}, j \in L'$;

stop if $u = 1$; otherwise set $u = u - 1$ and go to step 2.

Step 5: Set $P(j) = 0$, for all $j \in L'$.

Step 6: Determine c_{ij} for all $i, j \in L^{'}$, $i \neq j$.

Step 7: For all pairs $(i, j) \in L^{'}$, with $i \neq j$, do:

If $c_{ij} < c_{ji}$, set $P(i) = P(i) + 1$;

If $c_{ij} < c_{ji}$, set $P(j) = P(j) + 1$;

If $c_{ij} = c_{ji}$, set $P(i) = P(i) + 1$ and $P(j) = P(j) + 1$.

Step 8: Schedule job J_l for which $P(l) = \max\left\{P_j; j \in L^{'}\right\}$ and set $t = t + p_l$ and $L = L \setminus \{l\}$.

Step 9: Stop if $u = 1$; otherwise set $u = u - 1$ and go to step 2.

If $c_{ij} < c_{ji}$, it seems better to schedule job J_i in the next position rather than job J_j. The priority $P(j)$ of job J_j is therefore the number of times job J_j is the preferred job for the next position when it is compared with all other available unscheduled jobs. The Greedy heuristic selects, at each iteration, the job with the highest priority $P(j)$. Because of the $O(n^2)$ complexity of steps 6 and 7, the overall complexity of the heuristic is $O(n^3)$.

Finally, we consider a heuristic procedure, denoted as DTS, that can be described as follows. Let t, u, L and $L^{'}$ be as previously defined for the Greedy heuristic. The steps of the DTS procedure are:

Step 1: Initialize $t = \min\{r_j : j = 1, \ldots, n\}$, $u = n$ and $L = \{1, 2, \ldots, n\}$.

Step 2: Determine $L^{'} = \{j : j \in L \wedge r_j \leq t\}$.

Step 3: If $L^{'} = \emptyset$, set $t = \min r_j, j \in L$ and go to step 2.

Step 4: If $\left|L^{'}\right| = 1$

set $t = t + p_j, j \in L^{'}$;

set $L = L \setminus \{j\}, j \in L^{'}$;

stop if $u = 1$; otherwise set $u = u - 1$ and go to step 2.

Step 5: For each $j \in L^{'}$ do:

schedule j in the next position;

sequence the remaining jobs in L using a dispatch rule or other heuristic;

calculate the objective function value O_j of this partial schedule.

Step 6: Select job J_l with $O_l = \min\left\{O_j; j \in L^{'}\right\}$ to be scheduled next and set $t = t + p_l$ and $L = L \setminus \{l\}$.

Step 7: Stop if $u = 1$; otherwise set $u = u - 1$ and go to step 2.

The DTS procedure can be considered as a local search heuristic based on the decision theory approach of Kanet and Zhou [4]. The decision theory approach defines the alternative courses of action at each decision juncture, evaluates the consequences of each alternative according to a certain criterion, and then chooses the best alternative. In the DTS procedure we generate all possible scenarios by scheduling each of the currently available jobs next, and then sequence the remaining jobs using a dispatch rule or other similar heuristic. Each scenario is evaluated by calculating the objective function value O_j, and the job (and associated scenario) with the minimum O_j value is then chosen to be scheduled next. Alternatively, the DTS heuristic can also be called a beam search algorithm that performs a detailed evaluation at each node and has a beam width of one. The LIN-ET heuristic was chosen to sequence the remaining jobs in step 5, since it was the best-performing of the other heuristic procedures we analysed (see the computational results in section 4). The DTS procedure is then guaranteed to generate a sequence at least as good as that of the LIN-ET dispatch rule. In fact, the LIN-ET sequence is enumerated by the DTS algorithm, and will only be discarded if a superior schedule is found. Given the $O\left(n^3\right)$ complexity of step 5 when the LIN-ET heuristic is used, the overall complexity of the DTS procedure is $O\left(n^4\right)$.

3 Dominance rules

In this section we present the dominance rules that were used to improve the schedule generated by the heuristics. 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 [10] 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 [9] 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. After a heuristic has generated a schedule, these rules are applied as follows. 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).

4 Computational results

In this section we present the results from the computational tests. A set of problems with 15, 25, 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. Due to the large computational times that would be required, the DTS heuristic was not used on the 1000 job instances. 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, 25, 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

As we previously remarked, the effectiveness of the dispatch rules depends on the lookahead parameter k. We first performed extensive experiments to determine an appropriate value for k. An initial test was first conducted to determine the range where the best values of the lookahead parameter were concentrated. A more

detailed test was then performed in this range. In this test, we considered the values $\{0.2, 0.3, 0.4, \ldots 6.0\}$ and computed the objective function value for each k and each instance. These experiments showed that the best values of the lookahead parameter are in the range $[2.0, 3.0]$ for all dispatch rules. We recommend using a k of 2.5, since this value consistently provided good performance for all instance types.

In table 2 we present the average objective function value (mean ofv) for each heuristic, both before and after the application of the dominance rules (DR), and the average of the relative improvements in the objective function value (avg % imp), calculated as $\frac{H-H_{DR}}{H} * 100$, where H and H_{DR} are the objective function values of a heuristic before and after the use of the dominance rules, respectively. We also give the number of times each heuristic produces the best result when compared with the other heuristics, both before and after the application of the dominance rules. A test was also performed to determine if the differences between the heuristic objective function values before and after the dominance rules are statistically significant. Given that the heuristics were used on exactly the same problems, 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. Table 2 also includes the significance values (sig) of this test, i.e., the level of significance values above which the equal distribution hypothesis is to be rejected. In table 3 we give the average number of iterations (avg n^o iter) performed by the adjacent (adj) and non-adjacent (non adj) dominance rules, as well as the average percentage of the total objective function value improvement (avg % imp due) that is due to each rule.

From table 2 we can see that the use of the dominance rules improves the heuristic results, reducing the objective function value by an average that ranges from around one to four percent. The Wilcoxon test values also indicate that the differences in distribution between the heuristic results before and after the dominance rules are statistically significant. The average relative improvement in the objective function value is usually higher for the EXP-ET heuristic. The number of iterations of both dominance rules increases with the instance size and decreases with the processing time and penalty variability. These results are to be expected for the non-adjacent rule, since the probability of two jobs having the same processing time is higher when the instance size is large and the variability is low. The adjacent rule performs a larger number of iterations than the non-adjacent rule. The average percentage of the total objective function value improvement that is due to the non-adjacent rule increases with the instance size and decreases with the processing time and

			mean ofv				n^o times best	
var	n	Heur	bfr DR	aft DR	avg % imp	sig	bfr DR	aft DR
low	25	LIN-ET	1689	1654	2,6	0,000	7	49
		EXP-ET	1716	1656	4,4	0,000	6	55
		WPT-MS	1691	1660	2,4	0,000	17	58
		Greedy	1712	1698	0,8	0,000	49	33
		DTS	1618	1598	1,7	0,000	128	141
	100	LIN-ET	22472	22110	1,8	0,000	14	11
		EXP-ET	22673	22185	2,7	0,000	7	9
		WPT-MS	22604	22234	2,0	0,000	9	6
		Greedy	22912	22495	2,0	0,000	2	5
		DTS	21880	21502	2,2	0,000	149	151
	500	LIN-ET	502860	490980	2,4	0,000	30	16
		EXP-ET	503250	491235	2,5	0,000	11	11
		WPT-MS	503391	491640	2,4	0,000	14	11
		Greedy	505964	493125	2,7	0,000	3	5
		DTS	494483	485067	2,7	0,000	122	138
high	25	LIN-ET	141134	138252	2,7	0,000	8	43
		EXP-ET	143480	139099	3,9	0,000	8	41
		WPT-MS	147210	143906	3,0	0,000	7	23
		Greedy	143448	142518	0,9	0,000	44	35
		DTS	136266	134228	1,9	0,000	130	141
	100	LIN-ET	1722755	1704511	1,4	0,000	7	6
		EXP-ET	1731078	1704559	2,1	0,000	3	11
		WPT-MS	1769785	1749870	1,5	0,000	1	1
		Greedy	1749002	1738640	0,8	0,000	5	0
		DTS	1653346	1631417	1,8	0,000	164	162
	500	LIN-ET	37742317	37500648	0,7	0,000	3	1
		EXP-ET	37772697	37489037	0,9	0,000	2	4
		WPT-MS	37984997	37728441	0,8	0,000	2	3
		Greedy	38022178	37771122	0,7	0,000	0	1
		DTS	36657046	36432201	0,9	0,000	173	171

Table 2: Heuristic results: objective function value and statistical test

		low var				high var			
		avg n^o iter		avg % imp due		avg n^o iter		avg % imp due	
n	Heur	adj	non adj	adj	non adj	adj	non adj	adj	non adj
25	LIN-ET	2,3	1,2	96,0	4,0	2,2	1,0	100,0	0,0
	EXP-ET	2,5	1,2	97,6	2,4	2,3	1,0	100,0	0,0
	WPT-MS	2,2	1,2	95,2	4,8	2,0	1,0	99,9	0,1
	Greedy	1,4	1,2	74,7	25,3	1,4	1,0	96,8	3,2
	DTS	1,8	1,0	99,3	0,7	1,8	1,0	100,0	0,0
100	LIN-ET	4,0	2,2	76,9	23,1	3,4	1,3	94,0	6,0
	EXP-ET	4,5	2,4	79,7	20,3	3,6	1,3	95,4	4,6
	WPT-MS	4,0	2,4	74,3	25,7	3,1	1,4	93,0	7,0
	Greedy	3,5	2,6	59,2	40,8	2,4	1,3	93,3	6,7
	DTS	4,4	2,1	86,0	14,0	3,6	1,1	98,9	1,1
500	LIN-ET	8,2	5,1	31,7	68,3	5,8	2,3	72,4	27,6
	EXP-ET	8,2	4,9	39,2	60,8	6,2	2,3	75,7	24,3
	WPT-MS	8,1	5,0	31,1	68,9	4,9	2,3	70,9	29,1
	Greedy	7,7	5,7	24,3	75,7	4,9	2,4	70,3	29,7
	DTS	14,0	4,9	55,8	44,2	9,1	2,6	85,3	14,7

Table 3: Dominance rules: iterations and relative importance

penalty variability, which agrees once again with the higher probability of equal processing times. The number of adjacent rule iterations is usually lower for the Greedy heuristic, while the relative improvement due to the non-adjacent rule is higher. This suggests that the Greedy heuristic generates schedules that are more difficult to improve with adjacent interchanges, which agrees with the job selection criterion used by this procedure. From the objective function values and the number of times each heuristic is the best, we can also conclude the following. As expected, the DTS heuristic provided the best results for all instance types, both before and after the application of the dominance rules, and was usually two to four percent better than the best of the remaining procedures. The LIN-ET dispatch rule was the best of the other heuristics, since it provided the lowest average objective function value for nearly all instance types, both before and after the application of the dominance rules. This is a somewhat surprising result, since the EXP-ET heuristic provided better results than the LIN-ET in the computational tests performed by Ow and Morton for the problem with identical release dates. The Greedy heuristic was usually inferior to the dispatch rules, despite its higher computational complexity.

The effect of the α and β parameters on the relative objective function value improvement for the DTS heuristic is presented in table 4. The relative improvement

usually increases with α and β, with the clear exception of the ($\alpha = 0.75, \beta = 0.50$) parameter combination. In table 5 we present the runtimes (in seconds) for instances with 100 or more jobs. We can see that the dominance rules require little additional computational effort, and therefore their use is recommended, since they allow for significant improvements in objective function value. The DTS heuristic is fast for small and medium size instances, but it requires high computation times for larger instances. The dispatch rules are extremely fast even for the largest instances. The Greedy heuristic is noticeably slower than the dispatch rules for the larger instances, even though its computation time is still relatively low. The DTS heuristic is therefore recommended for small or medium size instances. For large instance sizes, however, it requires excessive computation times, and the LIN-ET heuristic should then be used.

				var			
			low			high	
n	Alfa	$\beta = 0.10$	$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.10$	$\beta = 0.25$	$\beta = 0.50$
25	0.25	0,3	1,1	2,1	0,3	0,9	2,7
	0.50	0,6	1,2	3,4	0,7	3,2	4,3
	0.75	1,7	3,1	1,6	1,7	1,4	1,7
100	0.25	0,5	1,2	3,4	0,3	1,0	1,9
	0.50	1,4	3,0	3,3	0,9	2,1	2,9
	0.75	2,0	3,1	2,0	2,6	3,0	1,3
500	0.25	0,5	1,9	3,4	0,2	0,5	1,2
	0.50	0,9	2,7	5,0	0,5	0,7	1,5
	0.75	3,2	5,0	1,3	1,2	1,5	0,5

Table 4: Relative improvement for the DTS heuristic

We also compared the heuristic results with the optimum objective function value for instances with 15 and 25 jobs. In table 6 we present the average of the relative deviations from the optimum (% dev), calculated as $\frac{H-O}{O} * 100$, where H and O are the heuristic and the optimum objective function values, respectively. The number of times each heuristic generates an optimum schedule (n^o opt) is also given. The average performance of the DTS heuristic is quite good. Even before the dominance rules are applied, the DTS procedure provides results that are two and four percent above the optimum for instances with 15 and 25 jobs, respectively. After the application of the dominance rules, these values are only one and two percent, respectively. When the dominance rules are used, the DTS heuristic also provides optimal solutions for over seventy five percent of the 15 job instances and

	var							
	low				high			
Heur	n=100	n=250	n=500	n=1000	n=100	n=250	n=500	n=1000
LIN-ET	0,000	0,001	0,003	0,010	0,000	0,001	0,003	0,011
EXP-ET	0,000	0,001	0,006	0,014	0,000	0,001	0,003	0,016
WPT-MS	0,001	0,002	0,004	0,018	0,000	0,001	0,005	0,017
Greedy	0,011	0,168	1,336	10,666	0,012	0,168	1,321	10,533
DTS	0,182	5,613	82,098	—	0,180	5,652	82,474	—
LIN-ET+DR	0,001	0,006	0,032	0,221	0,001	0,002	0,010	0,055
EXP-ET+DR	0,001	0,007	0,033	0,184	0,000	0,003	0,011	0,051
WPT-MS+DR	0,001	0,007	0,033	0,218	0,001	0,003	0,014	0,064
Greedy+DR	0,012	0,173	1,360	10,799	0,012	0,169	1,328	10,568
DTS+DR	0,182	5,618	82,120	—	0,180	5,653	82,481	—

Table 5: Runtimes (in seconds)

forty percent of the 25 job instances. The average performance of the LIN-ET heuristic, when followed by the application of the dominance rules, is also quite adequate, since its results are four (six) percent above the optimum for the 15 (25) job instances. This procedure also generates an optimal solution for around fifty (fifteen) percent of the 15 (25) job instances.

	low var				high var			
	n = 15		n = 25		n = 15		n = 25	
Heur	% dev	n^o opt	% dev	n^o opt	% dev	n^o opt	% dev	n^o opt
LIN-ET	6,8	30	9,2	3	7,6	35	8,9	2
EXP-ET	8,6	24	11,6	4	11,3	21	11,6	2
WPT-MS	7,5	31	9,3	6	15,2	23	15,3	3
Greedy	8,1	56	10,9	17	8,0	59	11,2	17
DTS	2,2	99	3,8	38	2,2	102	4,3	25
LIN-ET + DR	3,9	88	6,2	31	4,0	88	5,8	22
EXP-ET + DR	4,1	81	6,4	32	5,3	79	7,0	26
WPT-MS + DR	4,8	77	6,5	40	9,7	61	11,6	15
Greedy + DR	6,9	68	9,9	23	7,2	70	10,2	19
DTS + DR	0,7	139	2,0	84	1,1	137	2,2	64

Table 6: Comparison with optimum objective function values

In table 7 we present the effect of the α and β parameters on the relative deviation from the optimum for the DTS + DR heuristic. The performance seems to be worse when one of the parameters is set at its highest value and when both are at their middle values, but otherwise these parameters do not appear to have any other clear

and consistent effect on the heuristic performance.

		var					
		low			high		
n	Alfa	$\beta = 0.10$	$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.10$	$\beta = 0.25$	$\beta = 0.50$
15	0.25	0,7	0,4	1,8	0,2	0,5	0,8
	0.50	0,0	0,8	1,5	0,2	0,9	3,7
	0.75	0,4	0,3	0,0	1,3	1,8	0,3
25	0.25	0,2	1,5	2,1	0,2	1,9	3,6
	0.50	0,7	4,4	2,3	1,2	2,3	3,2
	0.75	2,1	1,5	3,1	3,0	3,2	1,1

Table 7: Relative deviation from the opimum for the DTS + DR heuristic

5 Conclusion

In this paper we considered the single machine earliness/tardiness scheduling problem with different release dates and no unforced idle time and analysed the performance of three dispatch rules, a greedy procedure and a local search heuristic based on the decision theory of Kanet and Zhou. The dispatch rules use a lookahead parameter whose value must be specified. Experiments were performed to determine an appropriate value for this parameter. We also considered the use of dominance rules to improve the solutions obtained by the heuristics. The computational results show that the use of the dominance rules is recommended, since they improve the solution quality of all heuristics with little additional computational effort. The decision theory DTS heuristic provides the best results, and is recommended for small or medium size instances. For large instances, however, it requires excessive computation times, and the LIN-ET dispatch rule then becomes the method of choice.

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] Kanet, J. J., and Zhou, Z. A decision theory approach to priority dispatching for job shop scheduling. *Production and Operations Management 2* (1993), 2–14.

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

[6] 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.

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

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

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

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

[11] Valente, J. M. S., and Alves, R. A. F. S. An exact approach for the early/tardy scheduling problem with release dates. Working Paper 129, Faculdade de Economia do Porto, Portugal, 2003.

[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.