Available online at www.sciencedirect.com
ScienceDirect
Int. J. Production Economics 106 (2007) 261–274
international journal of production economics
www.elsevier.com/locate/ijpe

Heuristics for the early/tardy scheduling problem with release dates

Jorge M.S. Valente[a,*], Rui A.F.S. Alves[b]

[a]LIACC/NIAAD - Faculdade de Economia, Universidade do Porto, Rua Dr. Roberto Frias, 4200-464 Porto, Portugal
[b]Faculdade de Economia, Universidade do Porto, Portugal

Received 19 August 2003; accepted 28 June 2006
Available online 14 August 2006

Abstract

In this paper, we consider the single machine earliness/tardiness scheduling problem with release dates and no unforced idle time. We analyse the performance of a varied set of heuristics. This set includes simple scheduling rules, early/tardy dispatching heuristics, a greedy procedure and a decision theory heuristic. Two different approaches are considered to calculate a lookahead parameter used in the early/tardy dispatching heuristics, and extensive experiments were performed to determine an appropriate value for this parameter. We also propose an improvement procedure that uses some dominance rules to improve the solution obtained by the heuristics.

The computational results show that the use of the improvement step is recommended, since it reduces the objective function value with little additional computational effort. The best results were given by the decision theory heuristic, but this procedure is computationally expensive and therefore limited to small and medium size instances. For large instances, one of the early/tardy dispatching heuristics is then the heuristic of choice.

Keywords: Scheduling; Early/tardy; Release dates; Heuristics

1. Introduction

In this paper, we consider a single machine scheduling problem with release dates and due dates, earliness and tardiness costs, and no unforced machine idle time. Scheduling models with a single processor may appear to arise infrequently in practice. However, the performance of many production systems is quite often dictated by the quality of the schedules for a single bottleneck machine. Therefore, models with a single processor are most useful in practice for scheduling such a machine. Also, the analysis of single machine problems provides insights that prove valuable for scheduling more complex systems. In fact, multiple processor systems can sometimes be relaxed to a single machine problem, or a sequence of such problems. Furthermore, the solution procedures for some complex systems, such as job shop environments, often require solving single machine subproblems.

The different job due dates can represent the delivery dates committed to the costumers, or the time a certain part or component is required by a stage further down the production or assembly line. Scheduling problems with different release dates are also appealing, since in most real production

*Corresponding author. Tel.: +351 22 557 11 00; fax: +351 22 550 50 50.
E-mail address: jvalente@fep.up.pt (J.M.S. Valente).

0925-5273/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2006.06.006

settings the orders are released to the shop floor over time.

Scheduling models with both earliness and tardiness penalties are compatible with the philosophy of just-in-time (JIT) production, which emphasizes producing goods only when they are needed. In a JIT setting, early jobs must be held in inventory until their due dates, while jobs that finish late may cause a costumer, or a further stage in the production line, to delay or shutdown operations. Therefore, an ideal schedule in one in which all jobs are completed exactly on their due dates. Scheduling models with both early and tardy costs are then compatible with the JIT philosophy, since jobs are indeed scheduled to finish as close as possible to their due dates.

The earliness penalty, in addition to a holding cost for parts or finished products, may also represent deterioration in the production of perishable goods, as well as the cost of completing a project early in critical path analyses, as suggested by Sidney (1977). The tardiness penalty can represent rush shipping costs, lost sales and loss of goodwill, as well as disruptions and delays in stages further down the production line. We consider a general model with different penalties for earliness and tardiness. Furthermore, the penalties may be different for each job, reflecting the fact that each order may have different costumer priorities, as well as distinct storage costs.

We assume that no unforced machine idle time is allowed, so the machine is idle only when no unscheduled jobs are available. This assumption is appropriate for many production settings. When the capacity of the machine is limited when compared with the demand, the machine must be kept running in order to satisfy the costumers' orders. Idle time must also be avoided for machines with high operating costs, since the cost of keeping the machine running is then higher than the earliness cost incurred by completing a job before its due date. The assumption of no idle time is also justified when starting a new production run involves high setup costs or times. Some specific examples of production settings where the no idle time assumption is appropriate have been given by Korman (1994) and Landis (1993). More specifically, Korman considers the Pioneer Video Manufacturing (now Deluxe Video Services) disc factory at Carson, California, while Landis analyses the Westvaco envelope plant at Los Angeles.

The assumption of no unforced idle time is compatible with the existence of different release dates, as long as the forced idle time caused by the presence of 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 machine idleness cost is higher than the earliness cost.

The problem we consider can be formally stated as follows. A set of n independent jobs $\{J_1, J_2, \ldots, J_n\}$ has to be scheduled 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 preemptions are not allowed. Job J_j, $j = 1, 2, \ldots, n$, 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 minimizes the sum of weighted earliness and weighted tardiness costs $\sum_{j=1}^{n} (h_j E_j + w_j T_j)$ 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.

As a generalization of weighted tardiness scheduling (Lenstra et al., 1977), the problem is strongly NP-hard. Valente and Alves (2005a) 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 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 (1988), Li (1997) and Liaw (1999). Among the heuristics, Ow and Morton (1989) developed several 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). Baker and Scudder (1990) provide an excellent survey of scheduling problems with earliness and tardiness penalties, while Kanet and Sridharan (2000) give a review of scheduling models with inserted idle time that complements our focus on a problem with no machine idle time.

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 (2000) developed lower bounding procedures and a branch-and-bound algorithm. A recent survey of the state-of-the-art in weighted and unweighted tardiness scheduling can be found in Sen et al. (2003).

In this paper, we analyse the performance of a varied set of heuristics. This set includes simple and widely used scheduling rules, as well as early/tardy dispatching heuristics originally developed for the problem with equal release dates. We also consider a greedy procedure, as well as a heuristic method based on the decision theory approach of Kanet and Zhou (1993). The early/tardy dispatching rules include some parameters whose value must be specified, and extensive experiments were performed to determine appropriate values for these parameters. We also propose an improvement procedure that uses some dominance rules developed for the problem with equal release dates to improve the solution obtained by the heuristics.

The remainder of the paper is organized as follows. The heuristics are described in Section 2. In Section 3, we present the dominance rules and the improvement procedure that was used to improve the schedule obtained by the heuristics. The computational results are given in Section 4, and some concluding remarks are provided in Section 5.

2. The heuristics

2.1. Simple dispatching rules

We considered three simple but widely used scheduling rules, namely the earliest due date (EDD), weighted longest processing time (WLPT) and weighted shortest processing time (WSPT) heuristics. The EDD rule sequences the jobs in non-decreasing order of their due dates, and is frequently used for problems with due date related criteria. The WLPT heuristic schedules the jobs in non-decreasing order of their h_j/p_j ratios. This rule is suited to problems where most jobs will be completed early. In fact, Ow and Morton (1989) have proved that if the WLPT heuristic results in a schedule with no tardy jobs, then it is optimal for the early/tardy problem with identical release dates (see Lemma 2 in Ow and Morton, 1989). The WSPT heuristic sequences the jobs in non-increasing order of their w_j/p_j ratios. The WSPT heuristic is appropriate for problems where most jobs will be tardy. Indeed, it has also been proved by Ow and Morton (1989) that if the WSPT heuristic produces a schedule with no early jobs, then it is optimal for the early/tardy problem with identical release dates (see Lemma 1 in Ow and Morton, 1989). The time complexity of these rules for problems with different release dates is $O(n^2)$.

2.2. Early/tardy dispatching rules

The EDD, WLPT and WSPT rules are simple and widely used, and may perform well under some circumstances. However, the EDD rule ignores the unit penalties, while the WLPT (WSPT) heuristic only considers the earliness (tardiness) penalty. Therefore, a superior performance may be obtained using heuristics that specifically take into account both the earliness and the tardiness costs. We now present three early/tardy dispatching heuristics originally proposed for the problem with identical release dates.

The LINET dispatching heuristic was developed by Ow and Morton (1989), and 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:

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

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 EXPET rule was also presented by Ow and Morton, and uses the following priority index $I_j(t)$:

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

where W_j, H_j, s_j, $\overline{p}$ and k are as previously defined.

The last dispatching rule, denoted by WPTMS, was proposed by Valente and Alves (2005b) and uses the priority index:

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

where once more W_j, H_j, s_j, $\overline{p}$ and k are as previously defined.

These dispatching rules use a priority function that starts at $-H_j$ when jobs are in no danger of being tardy ($s_j \geqslant k\overline{p}$), and then gradually increases to a maximum of W_j when jobs are on time or late ($s_j \leqslant 0$). Therefore, the rules reflect 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 dispatching heuristics differ in the calculation of the job priorities for the intermediate values of the job slack. In the LINET heuristic, the priority decreases linearly as the job slack increases. Ow and Morton (1989) also experimented with other functions, and obtained good results with the exponential and cubic functions used in the EXPET rule. Valente and Alves (2005b) also tried different approaches for the calculation of the intermediate priority values, and the rectangular–hyperbolic and quadratic functions used in the WPTMS heuristic provided the best results for the problem with identical release dates. The borderline limit that separates the first and second conditions is also different in the WPTMS dispatching rule. This heuristic uses the condition $s_j \leqslant 1$ (instead of $s_j \leqslant 0$) to ensure that the maximum priority value is W_j even when the data is non-integer (the second condition would give a value larger than W_j when $0 < s_j < 1$).

The priority function of the early/tardy dispatching rules requires the average processing time $\overline{p}$. These rules can be implemented so that $\overline{p}$ is the average processing time of all jobs, and $\overline{p}$ is therefore calculated only once at the initialization step of the algorithms. However, it is also possible to implement these heuristics so that $\overline{p}$ represents the average processing time of the remaining available unscheduled jobs at time t. In this latter alternative, the value of $\overline{p}$ is updated dynamically at each algorithm iteration. We performed some preliminary tests with these two implementations, and the results were quite similar. We then selected the dynamic implementation, and therefore $\overline{p}$ is calculated at each iteration as the average processing time of the unscheduled and available jobs.

The effectiveness of these dispatching heuristics depends on the value of the lookahead parameter k. This parameter is related to the number of competing critical jobs, i.e., the number of jobs that may clash each time a sequencing decision is to be made. We considered two different approaches for setting the value of k. In the first approach, k is set at a fixed value, and therefore the same value of k is used for all instances, and also for all the iterations the heuristic performs for a given instance. This is the most usual method found in the literature for setting the value of the lookahead parameter in similar heuristics. The versions that use this first method will be denoted simply as LINET, EXPET and WPTMS.

In the second approach, the value of k is calculated dynamically at each iteration. Whenever a scheduling decision must be made, the characteristics of the current workload are used to determine an appropriate value for the lookahead parameter. At each time t, each unscheduled and available job can be classified into one of three major categories. When a job has a large slack, that job can be described as *early*. Conversely, a *late* job has a negative slack, i.e., that job is already late. Finally, we say a job is *critical* when it has a relatively low slack and is at risk of becoming late.

When most jobs are quite early or tardy, the problem is relatively easy, and the early and late jobs can therefore be jointly described as *non-critical*. If the workload consists only or mostly of non-critical jobs, a low value of k, denoted as k_L, is appropriate, since very few jobs will clash at a

sequencing decision. When most jobs are critical, the problem is harder, because several jobs have a low slack and will soon become late. The number of competing critical jobs is higher, and a higher value of k, denoted as k_H, is therefore adequate.

The following procedure is then used to calculate an appropriate value for k in this second approach. A critical interval of slack values $[0, \max\_slack]$ is first calculated. The upper limit in this interval is calculated as $\max\_slack = slack\_prop * n_U * \overline{p}$, where n_U is the number of currently unscheduled and available jobs, and $0 < slack\_prop < 1$ is a user-defined parameter. Therefore, the upper limit $\max\_slack$ is calculated as a proportion $slack\_prop$ of the total processing time of the unscheduled jobs available at time t. A job J_j is then said to be critical if $s_j \in [0, \max\_slack]$, and non-critical otherwise.

The proportion of critical jobs $prop\_crit$ is then calculated, where $prop\_crit$ is the number of critical jobs divided by the number of unscheduled and available jobs. Finally, the value of the lookahead parameter k is then set at $k = prop\_crit \times k_H + (1 - prop\_crit) \times k_L$. Hence, the value of k is a weighted average of k_H and k_L, with the weights given by the proportions of critical and non-critical jobs, respectively. The versions that use a variable value for the lookahead parameter k will be denoted as LINET_vk, EXPET_vk and WPTMS_vk.

Extensive computational tests were performed to determine adequate values for the fixed value of k in the first approach, and for the parameters $slack\_prop$, k_L and k_H in the dynamic procedures. These tests, and the values that were selected, are described in the computational results section. The time complexity of both versions of the early/tardy dispatching rules is $O(n^2)$.

2.3. Greedy heuristic

We now present a greedy procedure that was originally proposed by Valente and Alves (2005b) for the problem with identical release dates. This heuristic, denoted as Greedy, can be described as follows. Let t be the current time and 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, i.e., c_{xy} is the sum of the costs of J_x and J_y when they are completed at times $t + p_x$ and $t + p_x + p_y$, respectively. Let L be a list with the indexes of the yet unscheduled 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, i.e., $L' = \{j : j \in L \wedge r_j \leqslant t\}$. The steps of the greedy heuristic are:

Step 1: Initialize $t = \min\{r_j : j = 1, \ldots, n\}$ and $L = \{1, 2, \ldots, n\}$.
Step 2: Determine L'.
Step 3: If $L' = \emptyset$, set $t = \min r_j, j \in L$ and go to step 2.
Step 4: If $|L'| = 1$
set $t = t + p_j$, $j \in L'$;
set $L = L \backslash \{j\}$, $j \in L'$;
stop if $|L| = 1$; otherwise, 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 $P(l) = \max\{P_j; j \in L'\}$ and set $t = t + p_l$ and $L = L \backslash \{l\}$.
Step 9: Stop if $|L| = 1$; otherwise, 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 $O(n^3)$.

We now present a numerical example of the Greedy heuristic. Consider the following four jobs $J_1 = \{8, 1, 4, 0, 15\}$, $J_2 = \{6, 2, 6, 4, 17\}$, $J_3 = \{4, 5, 2, 6, 22\}$ and $J_4 = \{7, 2, 1, 8, 20\}$, where the job data is given as $\{p_j, h_j, w_j, r_j, d_j\}$. In the first iteration, we set $t = 0$ and $L = \{1, 2, 3, 4\}$ in step 1, and $L' = \{1\}$ in step 2. Since $L' \neq \emptyset$, no action is taken in step 3. In step 4, J_1 is the only available unscheduled job, and therefore it is selected for the first position in the schedule. The current time t is updated to $t = t + p_1 = 0 + 8 = 8$, we set $L = \{2, 3, 4\}$ and then advance to the next iteration at step 2.

In the second iteration, we set $L' = \{2, 3, 4\}$ in step 2. No action is performed in steps 3 and 4, and in step 5 we simply set $P_2 = P_3 = P_4 = 0$. In step 6, we calculate the c_{ij} costs. Since jobs 2 and 3 are completed early when scheduled in this order in the next two positions, we have $c_{23} = h_2(d_2 - t - p_2) + h_3(d_3 - t - p_2 - p_3) = 2 * 3 + 5 * 4 = 26$. Similarly, we calculate $c_{32} = 56$, $c_{24} = 7$, $c_{42} = 34$, $c_{34} = 52$

and $c_{43} = 25$. In step 7, P_2 is set at 2, since $c_{23} < c_{32}$ and $c_{24} < c_{42}$. The priority of job 4 is set at 1, because $c_{34} > c_{43}$. In step 8, we have $P_2 = 2$, $P_3 = 0$ and $P_4 = 1$, so we schedule job 2 next, update t to 14 and set $L = \{3, 4\}$. Since there are still two unscheduled jobs, in step 9 we return to step 2 and start a new iteration.

In iteration 3, we calculate $c_{34} = 25$ and $c_{43} = 7$, and schedule job 4 in the next position. Since there is only one unscheduled job remaining, the procedure is stopped in step 9. The final sequence is then 1–2–4–3, with a total cost of 20.

2.4. Decision theory heuristic

We also propose a heuristic procedure based on the decision theory approach of Kanet and Zhou (1993). This heuristic is denoted as DTS and can be described as follows. Let t be the current time and L a list with the indexes of the yet unscheduled jobs. Also, let L' be a list with the indexes of the unscheduled jobs that are available at time t. The steps of the DTS procedure are:

Step 1: Initialize $t = \min\{r_j : j = 1, \ldots, n\}$ and $L = \{1, 2, \ldots, n\}$.
Step 2: Determine L'.
Step 3: If $L' = \emptyset$, set $t = \min r_j, j \in L$ and go to step 2.
Step 4: If $|L'| = 1$
set $t = t + p_j$, $j \in L'$;
set $L = L \backslash \{j\}$, $j \in L'$;
stop if $|L| = 1$; otherwise, 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 $O_l = \min\{O_j; j \in L'\}$ to be scheduled next and set $t = t + p_l$ and $L = L \backslash \{l\}$.
Step 7: Stop if $|L| = 1$; otherwise, 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. This 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 dispatching 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 viewed as a beam search algorithm that performs a detailed evaluation at each node and has a beam width of one. The LINET_vk heuristic was chosen to sequence the remaining jobs in step 5, since it was the best-performing of the other heuristic procedures 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 LINET_vk dispatching rule. In fact, the LINET_vk sequence is enumerated by the DTS algorithm, and will only be discarded if a superior schedule is found. Given the $O(n^3)$ complexity of step 5 when the LINET_vk heuristic is used, the overall complexity of the DTS procedure is $O(n^4)$.

3. The improvement procedure

In this section, we propose an improvement procedure that uses two dominance rules developed for the problem with equal release dates to improve the solution obtained by the heuristics. Ow and Morton (1989) presented a dominance condition for adjacent jobs, while Liaw (1999) developed a rule that applies to non-adjacent jobs with identical processing times. These rules 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 proved (see Theorem 1 in Ow and Morton, 1989) 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:

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

with Ω_{xy} defined as

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

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 showed (see Theorem 5 in Liaw, 1999) 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:

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

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

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

where s_x and t are defined as before.

When different release dates are allowed, the previous conditions must still be satisfied whenever $r_j \leqslant t$. 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, the improvement procedure applies these rules 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. The adjacent condition is applied repeatedly until it cannot find an improving swap. Then, Liaw's non-adjacent rule is used. Once again, if a pair of jobs violates the rule those jobs are swapped, and the condition is applied until no improving swap is found. The above two steps are repeated while the non-adjacent rule detects an improvement (i.e., while it modifies the sequence generated by the adjacent rule).

4. Computational results

In this section, we first describe the set of problems used in the computational tests, and the experiments that were performed to determine adequate values for the parameters required by the early/tardy dispatching rules. We then compare the heuristic procedures, and analyse the effectiveness of the improvement step. Finally, we compare the heuristic results with optimum objective function values for some instance sizes. Throughout this section, and in order to avoid excessively large tables, we will sometimes present results only for some representative cases.

4.1. Experimental design

The computational tests were performed on a set of problems with $15, 20, 25, 30, 40, 50, 75, 100, 250, 500, 750, 1000, 1500$ and 2000 jobs. These problems were 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 (L) and high (H) variability, respectively. For each job J_j, an integer release date r_j was generated from the uniform distribution $[0, \alpha\sum_{j=1}^{n} p_j]$, 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 $\alpha = 1.00$ 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 $[0, \beta\sum_{j=1}^{n} p_j]$, 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 $d_j = (r_j + p_j) + s_j^d$. For each combination of instance size n, processing time and penalty variability (var), α and β, 50 instances were randomly generated. Therefore, a total of 450 instances were generated for each combination of problem size and variability. All the algorithms were coded in Visual C++ 6.0 and executed on a Pentium IV—2.8 GHz personal computer. Due to the large computational times that would be required, the DTS heuristic was only applied to instances with up to 250 jobs.

4.2. Parameter adjustment tests

We performed extensive computational tests in order to determine appropriate values for the parameters used by the early/tardy dispatching rules. A separate problem set was used to conduct these tests. This test set included instances with $25, 50, 100, 250, 500, 1000$ and 2000 jobs, and contained five instances for each combination of instance size, variability, α and β. The instances in this smaller test set were generated randomly just as previously described for the full problem set.

The versions that use a fixed lookahead parameter require a value for k. An initial test was conducted to determine the range where the best fixed values of the lookahead parameter were concentrated. A more detailed test was then performed in this range. In this second test, we considered the values $\{0.2, 0.3, 0.4, \ldots, 9.0\}$, and computed the objective function value for each

value of k and each instance. An analysis of these results showed that the best fixed values of k were usually in the range [2.0, 3.0] for all three dispatching rules. We then decided to set k at 2.5, since this value consistently provided good results for all instance types.

The dynamic versions require a value for the parameters *slack_prop*, k_L and k_H. The value of k_L was set at 0.5, since the experiments performed for the fixed value versions showed that this value provided a good performance for instances with mostly early or tardy jobs. Some additional preliminary experiments also confirmed that setting $k_L = 0.5$ was indeed an adequate choice. We then considered the following values for the parameters *slack_prop* and k_H:

$$slack\_prop : \{0.20, 0.25, 0.30, \ldots, 0.80\},$$

$$k_H : \{3.0, 3.5, 4.0, \ldots, 7.0\}.$$

The dynamic versions were applied to all the test set instances for each combination of these two parameter values. A thorough analysis of these results was then conducted in order to determine parameter values that provided an adequate performance across all instance types. We then decided to set $slack\_prop = 0.70$ for all the dispatching rules. The parameter k_H was set at 3.0 for the EXPET_vk and WPTMS_vk heuristics, and at 3.5 for the LINET_vk rule.

4.3. Heuristic results

We now present the computational results for the heuristic procedures. In Table 1, we present the average objective function value (ofv) for each heuristic, as well as the percentage number of times a heuristic gives the best result when compared with the other heuristics (%best). The average objective function values are calculated relative to the LINET_vk heuristic, and therefore are presented as index numbers. More precisely, these values are calculated as $heur\_ofv/linet\_vk\_ofv * 100$, where *heur_ofv* and *linet_vk_ofv* are the average objective function values of the appropriate heuristic and the LINET_vk dispatching rule, respectively.

The best results, as expected, were given by the DTS heuristic. This procedure provided a 4–5% improvement over the best of the remaining heuristics, and obtained the best objective function value for a quite large percentage of the instances. Among the other procedures, the LINET_vk and LINET dispatching rules are then the best performing heuristics, closely followed by the EXPET(_vk) algorithms. This is a somewhat surprising result, since the EXPET heuristic provided better results than the LINET in the computational tests performed by Ow and Morton for the problem with identical release dates. The WPTMS(_vk) rules are close to the other two early/tardy dispatching procedures for the instances with a low processing time and penalty variability. However, they are 2–4% worse than the LINET_vk heuristic for small and medium size instances with high variability.

The Greedy procedure is inferior to the early/tardy dispatching rules, despite its higher computational complexity. The simple EDD, WLPT and WSPT rules perform rather poorly, giving results that are substantially worse than those of the early/tardy heuristics. Therefore, ignoring the earliness and/or tardiness penalties is quite costly in terms of solution quality.

In Table 2, we present a comparison between the fixed and variable lookahead parameter versions of the early/tardy dispatching heuristics. This table gives the average of the relative improvements provided by the dynamic versions (%imp), as well as the percentage number of times the dynamic versions perform better ($<$), equal ($=$) or worse ($>$) than their fixed value counterparts. The improvement given by the dynamic versions is calculated as $(fixed - variable)/fixed * 100$, where *fixed* and *variable* are the objective function values of the versions with a fixed and a dynamic value for the lookahead parameter, respectively.

We also performed a test to determine if the difference between the fixed and variable lookahead parameter versions is statistically significant. Given that the heuristics were used on exactly the same problems, a paired-samples test is appropriate. Since not all the hypothesis of the paired-samples t-test were met, the non-parametric Wilcoxon test was selected. The significance values of this test (sig), i.e., the confidence level values above which the equal distribution hypothesis is to be rejected, are also provided in Table 2.

The LINET_vk and EXPET_vk heuristics perform better than their fixed value counterparts. These dynamic versions provide a positive, although sometimes minor, relative improvement, and they give better results for most of the test instances. The Wilcoxon test values also indicate that the difference in distribution between the fixed and dynamic versions is statistically significant. The comparison

Table 1
Heuristic results

var	heur	$n = 25$		$n = 100$		$n = 500$		$n = 1000$	
		ofv	%best	ofv	%best	ofv	%best	ofv	%best
L	EDD	174.97	0.00	212.95	0.00	236.80	0.00	243.98	0.00
	WLPT	304.15	0.00	387.68	0.00	441.17	0.00	460.31	0.00
	WSPT	161.96	0.44	200.65	0.00	229.94	0.00	239.39	0.00
	EXPET	102.14	2.44	101.05	0.44	100.35	5.11	100.21	7.56
	EXPET_vk	101.08	4.00	100.62	1.11	100.15	14.44	100.07	18.22
	LINET	100.71	3.56	100.56	1.78	100.22	17.78	100.16	12.22
	LINET_vk	100.00	6.22	100.00	0.22	100.00	32.44	100.00	35.78
	WPTMS	101.10	7.33	100.70	2.44	100.31	10.44	100.22	11.11
	WPTMS_vk	101.02	8.00	100.68	0.67	100.21	14.00	100.11	10.67
	Greedy	102.25	19.11	102.15	0.44	100.71	5.78	100.47	4.44
	DTS	96.32	82.67	96.44	92.89	–	–	–	–
H	EDD	184.42	0.00	236.59	0.00	271.83	0.00	278.07	0.00
	WLPT	333.76	0.00	446.67	0.00	519.27	0.00	535.49	0.00
	WSPT	163.64	0.00	216.51	0.00	251.64	0.00	260.06	0.00
	EXPET	102.53	3.78	101.30	1.33	100.34	9.78	100.16	15.78
	EXPET_vk	101.30	3.78	100.66	1.11	100.20	15.78	100.13	9.78
	LINET	100.45	5.11	100.55	1.56	100.18	23.33	100.06	22.00
	LINET_vk	100.00	3.56	100.00	0.00	100.00	44.22	100.00	43.33
	WPTMS	104.68	2.67	102.94	0.67	101.00	1.33	100.47	4.22
	WPTMS_vk	104.89	2.67	103.04	0.22	101.05	0.00	100.51	0.22
	Greedy	102.09	18.89	102.35	0.00	100.94	5.56	100.50	4.67
	DTS	95.73	78.00	95.57	95.11	–	–	–	–

Table 2
Fixed and variable lookahead parameter comparison

heur	n	var L					var H				
		%imp	sig	$<$	$=$	$>$	%imp	sig	$<$	$=$	$>$
EXPET_vk	15	0.94	0.00	35.33	46.89	17.78	1.36	0.00	39.56	46.22	14.22
	25	1.07	0.00	59.56	18.89	21.56	1.28	0.00	65.11	16.00	18.89
	50	1.27	0.00	74.44	2.00	23.56	1.40	0.00	76.67	2.00	21.33
	100	0.46	0.00	71.56	0.00	28.44	0.78	0.00	78.67	0.00	21.33
	250	0.44	0.00	74.89	0.00	25.11	0.30	0.00	73.56	0.00	26.44
	500	0.25	0.00	73.11	0.00	26.89	0.12	0.00	68.67	0.00	31.33
	1000	0.22	0.00	72.67	0.00	27.33	0.01	0.00	62.67	0.00	37.33
	2000	0.17	0.00	68.00	0.00	32.00	0.04	0.00	63.78	0.22	36.00
LINET_vk	15	0.54	0.00	34.67	48.00	17.33	0.28	0.00	35.56	46.22	18.22
	25	0.61	0.00	54.00	21.11	24.89	0.24	0.00	54.89	18.44	26.67
	50	0.82	0.00	66.89	2.89	30.22	0.72	0.00	65.33	2.22	32.44
	100	0.57	0.00	68.22	0.22	31.56	0.50	0.00	67.33	0.00	32.67
	250	0.34	0.00	66.67	0.00	33.33	0.29	0.00	65.56	0.00	34.44
	500	0.26	0.00	66.89	0.00	33.11	0.18	0.00	64.22	0.00	35.78
	1000	0.20	0.00	70.00	0.00	30.00	0.05	0.00	59.56	0.00	40.44
	2000	0.16	0.00	73.11	0.44	26.44	0.02	0.03	55.56	0.00	44.44
WPTMS_vk	15	0.08	0.08	18.44	70.67	10.89	−0.17	0.94	9.56	83.56	6.89
	25	−0.09	0.00	34.44	48.22	17.33	−0.26	0.43	18.89	67.33	13.78
	50	0.38	0.00	55.11	15.78	29.11	0.14	0.08	32.67	46.44	20.89
	100	−0.06	0.00	64.67	3.56	31.78	−0.15	0.69	44.00	22.44	33.56
	250	0.04	0.01	61.56	0.00	38.44	−0.14	0.03	46.22	8.00	45.78
	500	0.11	0.01	59.11	0.00	40.89	−0.10	0.00	44.67	2.00	53.33
	1000	0.14	0.00	62.00	0.22	37.78	−0.06	0.00	41.78	0.89	57.33
	2000	0.13	0.00	61.11	0.44	38.44	−0.04	0.00	37.56	3.78	58.67

Table 3
Heuristic runtimes (in seconds)

var	heur	$n = 100$	$n = 250$	$n = 500$	$n = 1000$	$n = 1500$	$n = 2000$
L	EDD	0.00	0.00	0.00	0.00	0.00	0.00
	WLPT	0.00	0.01	0.02	0.10	0.22	0.39
	WSPT	0.00	0.00	0.02	0.06	0.13	0.23
	EXPET	0.00	0.00	0.00	0.01	0.01	0.02
	EXPET_vk	0.00	0.00	0.00	0.02	0.03	0.07
	LINET	0.00	0.00	0.00	0.00	0.01	0.02
	LINET_vk	0.00	0.00	0.00	0.01	0.03	0.05
	WPTMS	0.00	0.00	0.00	0.01	0.01	0.03
	WPTMS_vk	0.00	0.00	0.00	0.02	0.03	0.06
	Greedy	0.00	0.07	0.57	4.51	15.14	35.87
	DTS	0.20	7.16	–	–	–	–
H	EDD	0.00	0.00	0.00	0.00	0.00	0.00
	WLPT	0.00	0.01	0.02	0.09	0.21	0.38
	WSPT	0.00	0.00	0.01	0.06	0.13	0.22
	EXPET	0.00	0.00	0.00	0.01	0.01	0.03
	EXPET_vk	0.00	0.00	0.00	0.02	0.04	0.07
	LINET	0.00	0.00	0.00	0.00	0.01	0.02
	LINET_vk	0.00	0.00	0.00	0.01	0.03	0.05
	WPTMS	0.00	0.00	0.00	0.01	0.01	0.03
	WPTMS_vk	0.00	0.00	0.00	0.02	0.04	0.06
	Greedy	0.00	0.07	0.56	4.43	14.91	35.34
	DTS	0.20	7.00	–	–	–	–

between the two versions, however, is not so clear for the WPTMS(_vk) heuristics. For instances with low processing time and penalty variability, the dynamic version is usually superior, while the precise opposite occurs for instances with high variability. The Wilcoxon test significance values usually indicate a statistically significant difference in distribution between the two versions, with the exception of small and medium size instances with high processing time and penalty variability.

In Table 3, we present the heuristic runtimes (in seconds). The DTS heuristic is computationally demanding, and therefore can only be used for small or medium size instances. The dispatching procedures are extremely fast, even for the largest instances. The Greedy heuristic is noticeably slower than the dispatching rules, but it can still solve large instances within reasonable computation times. The DTS heuristic is then recommended for small or medium size instances. For large instances, however, it requires excessive computation times, and the LINET_vk heuristic should then be used.

4.4. Improvement step results

We now present the computational results for the improvement step. In Table 4, we give the average of the relative improvements in the objective function value, calculated as $(ofv - imp\_ofv)/ofv * 100$, where *ofv* and *imp_ofv* are the objective function values of the appropriate heuristic before and after the application of the improvement step, respectively. A test was also performed to determine if the differences between the heuristic objective function values before and after the improvement step are statistically significant. The Wilcoxon test was once again chosen, and its significance values were always equal to 0.00.

From Table 4, we can see that the improvement step is effective in reducing the objective function value. The Wilcoxon test values also indicate that the differences in distribution between the heuristic results before and after the improvement step are statistically significant. The relative improvement is higher for the worst performing heuristics, but even the DTS procedure can benefit from a 1% to 1.5% improvement for most instance sizes. The results given by the early/tardy dispatching rules are also improved by around 2–3% (1–2%) for instances with low (high) processing time and penalty variability. The EDD, WLPT and WSPT rules benefit the most from the improvement step. These simple procedures were quite far in solution quality from the other heuristics, and therefore had a much larger room for improvement.

Table 4
Improvement procedure results

var	heur	$n = 25$	$n = 50$	$n = 100$	$n = 250$	$n = 500$	$n = 1000$	$n = 2000$
L	EDD	38.35	45.35	49.72	53.86	55.87	58.04	59.13
	WLPT	57.53	59.73	60.68	61.38	60.24	59.05	57.41
	WSPT	23.70	27.98	30.70	35.40	37.41	38.45	38.89
	EXPET	4.37	3.61	2.90	2.49	2.57	3.06	3.39
	EXPET_vk	2.87	2.27	2.09	2.04	2.27	2.86	3.20
	LINET	2.80	2.40	2.26	2.21	2.42	2.98	3.33
	LINET_vk	2.33	1.88	1.84	2.00	2.20	2.85	3.17
	WPTMS	2.37	2.08	1.87	2.11	2.34	2.97	3.34
	WPTMS_vk	1.87	1.58	1.62	1.91	2.18	2.84	3.18
	Greedy	0.99	1.50	1.94	2.31	2.57	3.19	3.44
	DTS	0.99	1.25	1.57	1.42	–	–	–
H	EDD	40.63	47.29	51.69	54.15	55.54	56.58	58.65
	WLPT	59.06	59.40	57.25	54.52	54.28	55.40	56.76
	WSPT	19.95	19.65	18.87	18.61	20.72	25.19	30.42
	EXPET	4.40	3.41	2.26	1.20	0.96	0.83	1.03
	EXPET_vk	3.05	2.07	1.34	0.73	0.67	0.69	0.94
	LINET	3.01	2.27	1.54	0.83	0.74	0.72	0.97
	LINET_vk	2.48	1.79	1.15	0.69	0.64	0.67	0.93
	WPTMS	2.86	2.21	1.39	0.83	0.76	0.72	0.99
	WPTMS_vk	2.76	2.05	1.26	0.75	0.75	0.72	0.99
	Greedy	0.65	0.87	0.88	0.66	0.70	0.76	1.01
	DTS	1.05	1.29	1.19	0.82	–	–	–

Table 5
Relative improvement for the LINET_vk heuristic

n	α	var L			var H		
		$\beta = 0.10$	$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.10$	$\beta = 0.25$	$\beta = 0.50$
25	0.25	0.72	1.41	3.40	0.67	2.10	3.51
	0.50	0.82	2.17	4.38	0.90	2.74	4.05
	0.75	1.81	4.06	2.20	1.44	4.97	1.91
50	0.25	0.49	1.17	2.52	0.52	1.39	2.34
	0.50	0.96	2.07	4.05	0.85	1.94	3.27
	0.75	1.91	2.68	1.05	1.95	2.77	1.09
100	0.25	0.59	2.55	2.37	0.47	0.73	1.62
	0.50	0.77	1.75	3.33	0.51	1.10	1.97
	0.75	1.72	2.58	0.87	1.53	1.91	0.54
500	0.25	1.35	3.31	4.65	0.29	0.99	1.16
	0.50	1.34	2.96	3.23	0.26	0.97	0.79
	0.75	1.20	1.51	0.26	0.53	0.67	0.16
1000	0.25	1.86	3.89	7.06	0.45	1.14	1.49
	0.50	1.99	3.72	4.19	0.36	0.93	0.68
	0.75	1.50	1.27	0.15	0.41	0.45	0.08

The effect of the α and β parameters on the relative objective function value improvement is presented in Table 5 for the LINET_vk heuristic. The relative improvement is usually non-decreasing with the due date slack range β, particularly when α is lower than 0.75. The improvement is usually lower when β is equal to 0.10 and the range of release dates α is set at 0.25 or 0.50.

Table 6
Improvement procedure runtimes (in seconds)

var	heur	$n = 250$	$n = 500$	$n = 750$	$n = 1000$	$n = 1500$	$n = 2000$
L	EDD	0.00	0.02	0.05	0.09	0.23	0.44
	WLPT	0.01	0.04	0.10	0.18	0.47	0.88
	WSPT	0.01	0.03	0.07	0.13	0.37	0.77
	EXPET	0.00	0.01	0.02	0.04	0.11	0.27
	EXPET_vk	0.00	0.01	0.02	0.03	0.09	0.21
	LINET	0.00	0.01	0.02	0.04	0.13	0.27
	LINET_vk	0.00	0.01	0.02	0.04	0.11	0.22
	WPTMS	0.00	0.01	0.02	0.04	0.13	0.31
	WPTMS_vk	0.00	0.01	0.02	0.04	0.11	0.22
	Greedy	0.00	0.01	0.02	0.04	0.11	0.23
	DTS	0.00	–	–	–	–	–
H	EDD	0.00	0.01	0.03	0.05	0.12	0.24
	WLPT	0.01	0.03	0.08	0.14	0.34	0.63
	WSPT	0.00	0.01	0.03	0.06	0.16	0.32
	EXPET	0.00	0.00	0.01	0.01	0.03	0.10
	EXPET_vk	0.00	0.00	0.01	0.01	0.03	0.05
	LINET	0.00	0.00	0.01	0.01	0.04	0.11
	LINET_vk	0.00	0.00	0.01	0.01	0.03	0.06
	WPTMS	0.00	0.00	0.01	0.01	0.04	0.11
	WPTMS_vk	0.00	0.00	0.01	0.01	0.03	0.06
	Greedy	0.00	0.00	0.01	0.01	0.03	0.06
	DTS	0.00	–	–	–	–	–

In Table 6, we give the computation time (in seconds) required by the improvement step. The additional computational effort is not very significant, since even for the largest instances the improvement step is executed in less than one second. Therefore, the use of the improvement step is recommended, since it is effective in reducing the objective function value.

The improvement step does not change the performance hierarchy among the various heuristics. The best results are once again given by the DTS procedure, followed by the early/tardy dispatching heuristics, and the simple EDD, WLPT and WSPT rules still perform poorly. However, the difference in performance between these heuristics has been somewhat reduced, since the relative improvement is higher for the worst performing heuristics.

4.5. Comparison with optimum results

The heuristic results were also compared with the optimum objective function values for instances with up to 30 jobs; results obtained after the application of the dominance rule-based improvement step are indicated by appending "+DR" to the heuristic identifiers. In Table 7, we present the average of the relative deviations from the optimum (%dev), calculated as $(H - O)/O * 100$, where H and O are the heuristic and the optimum objective function values, respectively. The percentage number of times each heuristic generates an optimum schedule (%opt) is also given.

From Table 7, we can see that all the heuristics are somewhat closer to the optimum for problems with a low processing time and penalty variability. The average performance of the DTS heuristic is quite good. Even before the application of the improvement step, the DTS procedure provides results that are 2–3% above the optimum. After the improvement step, the deviation is reduced to 1–2%. When the improvement step is applied, the DTS heuristic also provides optimal solutions for around 60% (35%) of the 20 (30) job instances. The average performance of the LINET_vk heuristic, when followed by the improvement step, is also quite adequate, since its results are 4–6% above the optimum.

In Table 8, we present the effect of the α and β parameters on the relative deviation from the optimum for the LINET_vk + DR heuristic. The relative deviation appears to increase with the due date slack range β, particularly when α is lower than 0.75. The heuristic performance is worst when β is

Table 7
Comparison with optimum objective function values

heur	var L				var H			
	$n = 20$		$n = 30$		$n = 20$		$n = 30$	
	%dev	%opt	%dev	%opt	%dev	%opt	%dev	%opt
EDD	77.87	0.00	95.78	0.00	95.38	0.00	114.69	0.00
WLPT	236.33	0.00	286.69	0.00	280.75	0.00	326.17	0.00
WSPT	82.57	0.44	106.42	0.00	87.66	0.22	110.99	0.00
EXPET	10.31	3.33	11.44	0.44	11.93	2.22	12.16	0.22
EXPET_vk	8.08	5.56	10.19	0.89	10.36	3.33	11.23	0.00
LINET	8.10	4.89	9.11	0.67	8.72	3.78	9.71	0.22
LINET_vk	6.86	5.78	8.37	1.78	7.97	6.67	9.23	0.22
WPTMS	7.92	6.89	9.62	0.89	15.24	4.00	15.39	0.00
WPTMS_vk	7.19	6.67	9.56	1.56	15.44	4.22	15.98	0.44
Greedy	8.93	16.44	11.83	4.22	10.42	17.56	12.47	2.44
DTS	1.84	45.11	2.94	20.22	2.25	42.67	3.16	19.78
EDD + DR	8.81	19.56	11.87	7.11	14.19	15.11	15.93	1.56
WLPT + DR	34.80	4.22	46.40	0.22	42.21	4.00	65.46	0.00
WSPT + DR	30.17	10.67	41.99	2.00	40.59	8.89	59.04	0.67
EXPET + DR	4.74	27.11	6.62	11.78	6.24	21.78	7.18	5.78
EXPET_vk + DR	4.44	28.89	6.91	12.22	6.33	21.56	8.07	6.22
LINET + DR	4.51	28.67	6.01	14.67	5.06	27.78	6.64	7.11
LINET_vk + DR	3.84	30.44	5.84	14.44	4.93	28.67	6.66	8.44
WPTMS + DR	5.01	25.33	6.89	11.33	10.63	16.44	12.08	2.22
WPTMS_vk + DR	4.88	27.33	7.46	10.89	11.10	16.89	12.91	3.11
Greedy + DR	7.86	21.56	10.51	6.00	9.41	22.00	11.66	4.00
DTS + DR	0.93	66.67	1.77	37.78	1.23	62.22	2.02	34.67

Table 8
Relative deviation from the optimum for the LINET_vk + DR heuristic

n	α	var L			var H		
		$\beta = 0.10$	$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.10$	$\beta = 0.25$	$\beta = 0.50$
15	0.25	1.38	3.36	9.77	1.14	3.08	7.46
	0.50	0.93	3.63	5.08	1.46	2.64	5.61
	0.75	1.67	4.01	6.93	3.32	4.60	4.83
20	0.25	1.33	4.89	5.95	1.20	5.10	9.12
	0.50	2.39	4.56	5.97	3.03	6.06	6.70
	0.75	2.77	4.09	2.62	3.40	5.07	4.74
25	0.25	0.93	3.10	9.48	1.19	3.59	12.57
	0.50	2.33	4.59	11.74	3.66	4.55	10.90
	0.75	3.09	4.96	5.14	5.88	5.48	4.70
30	0.25	1.28	6.04	9.97	1.67	4.32	12.25
	0.50	2.27	7.22	10.28	2.55	6.89	14.67
	0.75	4.81	5.43	5.29	4.19	8.08	5.35
40	0.25	2.18	5.37	13.24	2.54	7.03	14.88
	0.50	4.07	6.99	14.82	5.70	8.78	11.66
	0.75	4.27	8.09	6.24	6.34	6.07	7.31

equal to 0.50 and α is equal to 0.25 or 0.50. The heuristics are usually closer to the optimum when β is equal to 0.10 and the range of release dates α is set at 0.25 or 0.50. These results are similar to those reported for the improvement step, and seem to indicate that the problem is harder when the due date slack range is high and the release dates are not widely spread. This is to be expected, since the number of jobs that can be scheduled in a given position is larger when the release dates are only slightly scattered. Also, most jobs will likely be tardy when β is low, and as β increases there will be a greater balance between the number of early and tardy jobs, so the problem becomes much harder.

5. Conclusion

In this paper, we considered the single machine scheduling problem with earliness and tardiness penalties, release dates and no unforced idle time. We analysed the performance of a large and varied set of heuristics on a wide range of instances, and provided recommendations on which heuristic to use. This set of heuristics included simple but widely used scheduling rules, as well as early/tardy dispatching heuristics. A greedy-type procedure was also considered, as well as a heuristic method based on the decision theory approach of Kanet and Zhou.

Two different approaches were used to calculate a lookahead parameter required by the early/tardy dispatching heuristics. The first is the most usual method found in the literature, and it uses a fixed value, while the second calculates the parameter value dynamically. Extensive experiments were performed to determine appropriate values for the parameters used in these early/tardy dispatching rules. We also proposed an improvement procedure that uses some dominance rules to improve the solution obtained by the heuristics.

The computational results show that the use of the improvement step is recommended, since it is effective in reducing the objective function value, and the additional computational effort is not very significant. The dynamic versions of the LINET and EXPET dispatching rules outperformed their fixed value counterparts, while the WPTMS_vk heuristic only provided better results that the fixed value version for instances with a low processing time and penalty variability. The best results were given by the decision theory DTS heuristic, and this procedure is recommended for small or medium size problems. For large instances, however, the DTS procedure requires excessive computation times, and the LINET_vk dispatching rule then becomes the heuristic of choice.

Acknowledgement

The authors would like to thank the anonymous referees for several, and most useful, comments and suggestions that were used to improve this paper.

References

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

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

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

Baker, K.R., Scudder, G.D., 1990. Sequencing with earliness and tardiness penalties: a review. Operations Research 38, 22–36.

Kanet, J.J., Sridharan, V., 2000. Scheduling with inserted idle time: problem taxonomy and literature review. Operations Research 48, 99–110.

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

Korman, K., 1994. A pressing matter. Video, 46–50.

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.

Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P., 1977. Complexity of machine scheduling problems. Annals of Discrete Mathematics 1, 343–362.

Li, G., 1997. Single machine earliness and tardiness scheduling. European Journal of Operational Research 96, 546–558.

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.

Ow, P.S., Morton, T.E., 1989. The single machine early/tardy problem. Management Science 35, 177–191.

Sen, T., Sulek, J.M., Dileepan, P., 2003. Static scheduling research to minimize weighted and unweighted tardiness: a state-of-the-art survey. International Journal of Production Economics 83, 1–12.

Sidney, J., 1977. Optimal single-machine scheduling with earliness and tardiness penalties. Operations Research 25, 62–69.

Valente, J.M.S., Alves, R.A.F.S., 2005a. An exact approach to early/tardy scheduling with release dates. Computers and Operations Research 32, 2905–2917.

Valente, J.M.S., 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.