Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 35 (2008) 3696–3713
computers & operations research
www.elsevier.com/locate/cor

Heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties

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, Rua Dr. Roberto Frias, 4200-464 Porto, Portugal*

Available online 19 April 2007

Abstract

In this paper, we consider the single machine scheduling problem with quadratic earliness and tardiness costs, and no machine idle time. We propose several dispatching heuristics, and analyse their performance on a wide range of instances. The heuristics include simple and widely used scheduling rules, as well as adaptations of those rules to a quadratic objective function. We also propose heuristic procedures that specifically address both the earliness and the tardiness penalties, as well as the quadratic cost function. Several improvement procedures were also analysed. These procedures are applied as an improvement step, once the heuristics have generated a schedule.

The computational experiments show that the best results are provided by the heuristics that explicitly consider both early and tardy costs, and the quadratic objective function. Therefore, it is indeed important to specifically address the quadratic feature of the cost function, instead of simply using procedures originally developed for a linear objective function. The heuristics are quite fast, and are capable of quickly solving even very large instances. The use of an improvement step is recommended, since it usually improves the solution quality with little additional computational effort.

Keywords: Scheduling; Single machine; Early/tardy; Quadratic penalties; Dispatching rules

1. Introduction

In this paper, we consider a single machine scheduling problem with quadratic earliness and tardiness costs, and no machine idle time. Formally, the problem can be 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, $j = 1, 2, \ldots, n$, requires a processing time p_j and should ideally be completed on its due date d_j. Also, let h_j and w_j denote the earliness and tardiness penalties of job J_j, respectively. 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 the weighted quadratic earliness and tardiness costs $\sum_{j=1}^{n}(h_j E_j^2 + w_j T_j^2)$, subject to the constraint that no machine idle time is allowed.

* Corresponding author. Fax: +351 22 550 50 50.
E-mail address: jvalente@fep.up.pt (J.M.S. Valente).

0305-0548/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2007.04.006

Scheduling models with a single processor may appear to arise infrequently in practice. However, this scheduling environment actually occurs in several activities (for a specific example in the chemical industry, see [1]). Furthermore, the performance of many production systems is quite often dictated by the quality of the schedules for a single bottleneck machine. Also, the analysis of single machine problems provides insights that prove valuable for scheduling more complex systems. In fact, the solution procedures for some complex systems often require solving single machine subproblems.

Scheduling models with both earliness and tardiness penalties are compatible with the philosophy of just-in-time (JIT) production. The JIT production philosophy emphasizes producing goods only when they are needed, and therefore takes up the view that both earliness and tardiness should be discouraged. Therefore, an ideal schedule is 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.

In this paper, we consider quadratic earliness and tardiness penalties, instead of the more usual linear objective function. Therefore, deliveries that are quite early or tardy are more heavily penalized. On the one hand, this avoids schedules in which a single or only a few jobs contribute the majority of the cost, without regard to how the overall cost is distributed. On the other hand, in several practical settings, the penalties of non-conformance with the due dates do indeed increase in severity in a non-linear fashion.

We assume that no machine idle time is allowed. This assumption is actually appropriate for many production settings. Indeed, when the capacity of the machine is limited when compared with the demand, the machine must be kept running in order to satisfy the customers' 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. Furthermore, 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 [2] and Landis [3]. 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.

This problem has been previously considered by Valente [4]. He presented a lower bounding procedure, as well as a branch-and-bound algorithm. The corresponding problem with no idle time and weighted linear earliness and tardiness costs $\sum_{j=1}^{n}(h_j E_j + w_j T_j)$ has also been considered by several authors, and both exact and heuristic approaches have been proposed. Among the exact approaches, lower bounds and branch-and-bound algorithms were presented by Abdul-Razaq and Potts [5], Li [6], Liaw [7] and Valente and Alves [8]. Among the heuristics, several dispatching rules and beam search algorithms were presented by Ow and Morton [9] and Valente and Alves [10,11].

Linear early/tardy problems with inserted idle time have also been previously analysed. Sourd and Kedad-Sidhoum [12] considered a general version of the weighted problem, and presented a lower bounding procedure and a branch-and-bound algorithm. Some more specific versions have also been analysed in [13] (tolerance windows), [14] (identical processing times) and [15] (common due date).

The non-weighted linear problem $\sum_{j=1}^{n}(E_j + T_j)$ with inserted idle time has been considered by Kim and Yano [16] and Ventura and Radhakrishnan [17]. Kim and Yano presented a lower bound and a branch-and-bound algorithm, while Ventura and Radhakrishnan proposed a lagrangean relaxation procedure that provides both lower and upper bounds. The common due date version of this problem was analysed by Sundararaghavan and Ahmed [18].

Problems with a related quadratic objective function have also been previously considered. Schaller [19] analysed the single machine problem with inserted idle time and a linear earliness and quadratic tardiness $\sum_{j=1}^{n}(E_j + T_j^2)$ objective function. He presented a timetabling procedure to optimally insert idle time in a given sequence, as well as branch-and-bound and heuristic algorithms.

The minimization of the quadratic lateness $\sum_{j=1}^{n} L_j^2$, where the lateness of J_j is defined as $L_j = C_j - d_j$, has also been considered. Gupta and Sen [20] presented a branch-and-bound algorithm and a heuristic rule for the problem with no idle time. Su and Chang [21] and Schaller [22] considered the insertion of idle time, and proposed timetabling procedures and heuristic algorithms. Sen et al. [23] presented a branch-and-bound algorithm for the weighted problem $\sum_{j=1}^{n} w_j L_j^2$ where idle time is allowed only prior to the start of the first job. Baker and Scudder [24] provide an excellent survey of scheduling problems with earliness and tardiness penalties, while Kanet and Sridharan [25] give a review of scheduling models with inserted idle time.

In this paper, we propose and analyse several dispatching heuristics. We consider simple but widely used scheduling rules, and also propose adaptations of those rules to the quadratic objective function. Several heuristics that simultane-

ously consider the earliness and tardiness penalties, as well as the quadratic cost function, are also presented. Extensive computational experiments were performed in order to determine appropriate values for the parameters required by some of the heuristic algorithms. We also propose several improvement procedures. These procedures are applied as an improvement step after the heuristics have generated a schedule.

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

2. The heuristics

2.1. Simple linear dispatching rules

We consider three simple scheduling rules, namely the weighted longest processing time (WLPT), weighted shortest processing time (WSPT) and earliest due date (EDD) heuristics. The WLPT (WSPT) rule schedules the jobs in non-increasing order of their p_j/h_j (w_j/p_j) ratios, while the EDD heuristic sequences the jobs in non-decreasing order of their due dates. Since they only involve sorting, the time complexity of these rules is $O(n \log n)$.

These heuristics are included for two main reasons. On the one hand, these rules are quite well known and widely used in practice, so it seems sensible to include them for comparison purposes. On the other hand, these heuristics have been used before for the problem with a linear objective function. Indeed, these rules have some interesting properties for the linear earliness/tardiness problem.

The WLPT rule is particularly suited to problems where most jobs will be completed early. In fact, Ow and Morton [9] proved that the WLPT heuristic is optimal if it results in a schedule with no tardy jobs. Conversely, the WSPT rule is appropriate for problems where most jobs will be tardy. Indeed, Ow and Morton also proved that the WSPT sequence is optimal if it does not contain any early jobs. Finally, the EDD rule performs better than either the WLPT or WSPT heuristics when there is a greater balance between the number of early and tardy jobs. Therefore, each one of these simple rules performs quite well, under the appropriate circumstances, for the linear early/tardy problem. For that reason, it seems appropriate to analyse their performance for the problem with a quadratic cost function.

2.2. Simple quadratic dispatching rules

The WLPT and WSPT rules are locally optimal, under the appropriate conditions, for the problem with a linear objective function. In fact, if two adjacent jobs are always early, regardless of their order, those jobs should be scheduled in WLPT order. Conversely, when two adjacent jobs are necessarily tardy, regardless of their order, it is optimal to schedule those jobs in WSPT order. In this section, we present two dispatching rules that are derived from local optimality conditions for the problem with quadratic costs. Therefore, these heuristics are essentially an adaptation of the WLPT and WSPT rules to a quadratic objective function.

Theorem 1. *Consider any two adjacent jobs i and j that are always early, regardless of their order. In an optimal sequence, all such adjacent pairs of jobs must satisfy the following condition*:

$$\frac{h_i}{p_i}[p_j - 2(d_i - t - p_i)] \geqslant \frac{h_j}{p_j}[p_i - 2(d_j - t - p_j)],$$

where job i immediately precedes job j, and t is the start time of job i.

Proof. The condition can be established using simple interchange arguments. For the sake of brevity, we omit the details. □

Theorem 1 provides a local optimality condition for two adjacent jobs that are always early, regardless of their order. The left (right) side of this expression can be interpreted as the priority of job i with respect to job j (job j with respect to job i) at time t. A dispatching rule priority index can then be derived by comparing the priority of each job with an average job with processing time $\overline{p}$, where $\overline{p}$ is the average processing time of the remaining unscheduled jobs.

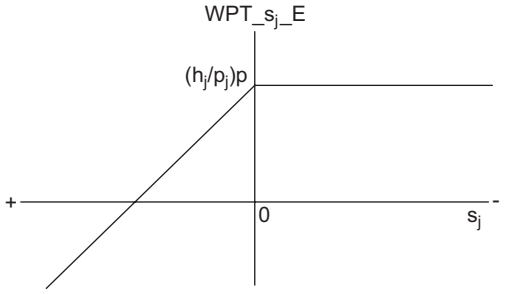

Fig. 1. WPT_s_j_E priority index.

Therefore, the priority index of job j at time t, denoted as $I_j(t)$, can be calculated as

$$I_j(t) = \frac{h_j}{p_j}[\overline{p} - 2\max(d_j - t - p_j, 0)].$$

At each iteration, the WPT_s_j_E dispatching rule selects the unscheduled job with the largest priority. The priority index of the WPT_s_j_E heuristic includes both a weighted processing time (WPT) component, and a slack (s_j) component. In fact, the slack of job j is defined as $s_j = d_j - t - p_j$. When a job is tardy, the priority index of the WPT_s_j_E heuristic is just equal to $(h_j/p_j)\overline{p}$. When a job is early, however, this ratio is modified by a slack-related component, and the priority decreases as the job's slack becomes larger (see Fig. 1).

This dispatching heuristic is particularly suited to problems where most jobs will be completed before their due dates, since it is derived from a local optimality condition for early (E) jobs. The WPT_s_j_E rule is then an adaptation of the WLPT heuristic to a quadratic objective function.

Theorem 2. *Consider any two adjacent jobs i and j that are always tardy, regardless of their order. In an optimal sequence, all such adjacent pairs of jobs must satisfy the following condition*:

$$\frac{w_i}{p_i}[p_j + 2(t + p_i - d_i)] \geqslant \frac{w_j}{p_j}[p_i + 2(t + p_j - d_j)],$$

where job i immediately precedes job j, and t is the start time of job i.

Proof. The condition can be established using simple interchange arguments. For the sake of brevity, we omit the details. □

Theorem 2 provides a local optimality condition for two adjacent jobs that are always tardy, regardless of their order. The left (right) side of this expression can again be interpreted as the priority of job i with respect to job j (job j with respect to job i) at time t. A dispatching rule priority index can once more be derived by comparing the priority of each job with an average job with processing time $\overline{p}$. Therefore, the priority index of job j at time t can be calculated as

$$I_j(t) = \frac{w_j}{p_j}[\overline{p} + 2\max(t + p_j - d_j, 0)].$$

At each iteration, the WPT_s_j_T dispatching rule selects the unscheduled job with the largest priority. Again, the priority index of the WPT_s_j_T heuristic includes both weighted processing time and slack components. In fact, the WPT_s_j_T heuristic is actually equivalent to the WSPT rule when a job is early, since the priority of job j is then equal to $(w_j/p_j)\overline{p}$. When a job is tardy, however, the WSPT ratio is modified by a slack-related component, and the priority increases with the job's tardiness (see Fig. 2).

The WPT_s_j_T dispatching heuristic is appropriate for problems where most jobs will be completed after their due dates, since it is derived from a local optimality condition for tardy jobs. Therefore, the WPT_s_j_T rule can be seen as an adaptation of the WSPT heuristic to a quadratic objective function. The time complexity of both the WPT_s_j_E and WPT_s_j_T dispatching rules is $O(n^2)$.

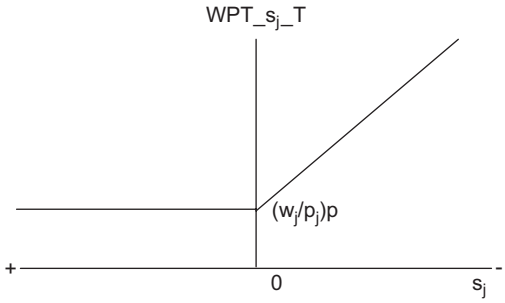

Fig. 2. WPT_s_j_T priority index.

2.3. The ECTL heuristic

We performed preliminary computational tests that showed that the new WPT_s_j_E and WPT_s_j_T dispatching heuristics indeed performed better than the WLPT and WSPT rules. These tests also showed that the WPT_s_j_E heuristic is particularly suited to problems where most jobs will be early, while the WPT_s_j_T rule performs well for problems where most jobs will be late. Moreover, the preliminary tests indicated that the EDD heuristic is superior to both the WPT_s_j_E and WPT_s_j_T dispatching rules when there is a greater balance between the number of early and tardy jobs. Therefore, each one of these three heuristics can perform quite well, under the appropriate circumstances.

In this section, we present a heuristic (denoted as ECTL) that tries to take advantage of the strengths of the WPT_s_j_E, EDD and WPT_s_j_T rules. At each iteration, the ECTL heuristic selects the next job using one of these three rules. The choice of dispatching rule is based on the characteristics of the current workload. In fact, at each iteration the ECTL heuristic chooses the rule that is expected to perform better, given the characteristics of the current set of unscheduled jobs.

The workload for a weighted quadratic earliness/tardiness problem can be classified into one of three major categories. When most jobs have large slacks, the current workload can be described as *early*. Conversely, a *late* load consists mainly of jobs with negative slack, i.e., jobs that are already late. Finally, we say the load is *critical* when several jobs have relatively low slacks, and are at risk of becoming late. At each iteration, the ECTL heuristic classifies the current workload as early, critical or tardy (in fact, ECTL stands for early-critical-tardy load). Then, the WPT_s_j_E (EDD/WPT_s_j_T) rule is selected if the load is early (critical/tardy).

We considered two versions of the ECTL procedure. These versions share this basic framework, and differ mainly in the criterion used to classify a workload as early, critical or tardy. In both versions, 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 unscheduled 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 currently unscheduled jobs.

The ECTL_AS version calculates the average slack $\overline{s}$ of the remaining unscheduled jobs. The current workload is then classified as early (critical/tardy) if $\overline{s} > \max\_slack$ ($\overline{s} \in [0, \max\_slack]/\overline{s} < 0$). In the ECTL_LP version, on the other hand, each job is classified as early, critical or tardy. A job is said to be early (critical/tardy) if $s_j > \max\_slack$ ($s_j \in [0, \max\_slack]/s_j < 0$). The ECTL_LP version then calculates the proportion of early, critical and tardy jobs. The current workload is classified as early (critical/tardy) if the percentage of early (critical/tardy) jobs is the largest. The time complexity of both versions of the ECTL heuristic is $O(n^2)$.

2.4. Early/tardy dispatching rules

The preliminary computational tests showed, as mentioned before, that the priority index of the WPT_s_j_E heuristic is indeed appropriate for early jobs. Conversely, these tests also showed that the WPT_s_j_T heuristic provides adequate priority values for tardy jobs. Therefore, each one of these two priority indexes can perform well, under the appropriate circumstances.

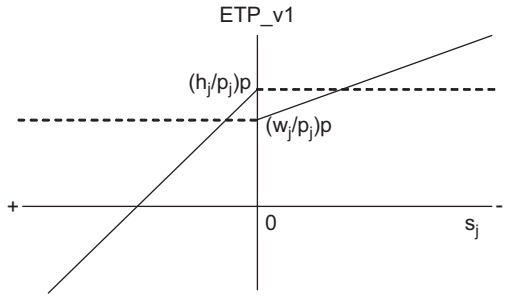

Fig. 3. ETP_v1 priority index.

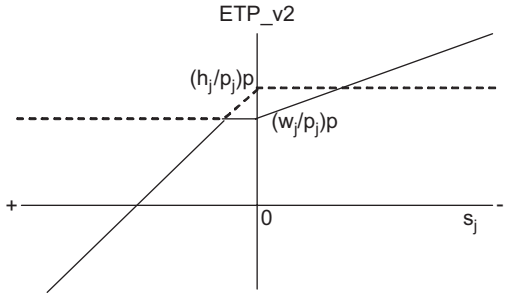

Fig. 4. ETP_v2 priority index.

In this section, we present a heuristic (denoted by ETP) that uses the priority indexes of both the WPT_s_j_E and the WPT_s_j_T rules, in order to take advantage of their respective strengths. At each iteration, the priority of each job is calculated using one of these two priority indexes (in fact, ETP stands for early/tardy priority). The choice of priority index is based on the characteristics of each job. More specifically, the priority index is chosen according to the value of each job's slack. At each iteration, the ETP heuristic then selects the unscheduled job with the largest priority.

We considered four versions of the ETP procedure. The ETP_v1 version chooses the WPT_s_j_T (WPT_s_j_E) priority index for a given job j when $s_j \leqslant 0$ ($s_j > 0$). The ETP_v2 version is nearly identical. In fact, the ETP_v2 procedure also uses the WPT_s_j_T priority index when a job is tardy or on time. When $s_j > 0$, however, the ETP_v2 priority value is set equal to the minimum of the WPT_s_j_E and WPT_s_j_T priority indexes. In the ETP_v1 procedure, the priority value decreases at time $t = 0$ when $(h_j/p_j)\overline{p} > (w_j/p_j)\overline{p}$ (see Fig. 3). In the ETP_v2 version, the priority index was slightly modified, in order to assure that the priority value is always non-increasing in the job's slack (see Fig. 4).

The other two versions of the ETP procedure introduce a lookahead parameter k. Also, let T_j^0 denote the priority value of job j when $s_j = 0$ and the WPT_s_j_T priority index is used. Moreover, let $E_j^{k\overline{p}}$ be the priority of job j when $s_j = k\overline{p}$ and the WPT_s_j_E priority index is used. Therefore, we have $T_j^0 = (w_j/p_j)\overline{p}$ and $E_j^{k\overline{p}} = (h_j/p_j)[\overline{p} - 2k\overline{p}]$. In the remaining two versions of the ETP heuristic, the priority index of job j at time t is then calculated as

$$I_j(t) = \begin{cases} (w_j/p_j)[\overline{p} + 2(t + p_j - d_j)] & \text{if } s_j \leqslant 0, \\ T_j^0 - s_j(T_j^0 - E_j^{k\overline{p}})/k\overline{p} & \text{if } 0 < s_j < k\overline{p}, \\ (h_j/p_j)[\overline{p} - 2(d_j - t - p_j)] & \text{if } s_j \geqslant k\overline{p}. \end{cases}$$

The priority is provided by the WPT_s_j_E priority index when a job is in no danger of becoming tardy ($s_j \geqslant k\overline{p}$). Conversely, the WPT_$s_j$_T is used to calculate the priority value when a job is on time or late ($s_j \leqslant 0$). Finally, when a job is about to become tardy ($0 < s_j < k\overline{p}$), the priority is given by a linear interpolation between the priority values associated with $s_j = 0$ and $s_j = k\overline{p}$ (see Fig. 5). This approach is similar to the one used in the LINET heuristic presented by Ow and Morton [9] for the problem with a linear objective function.

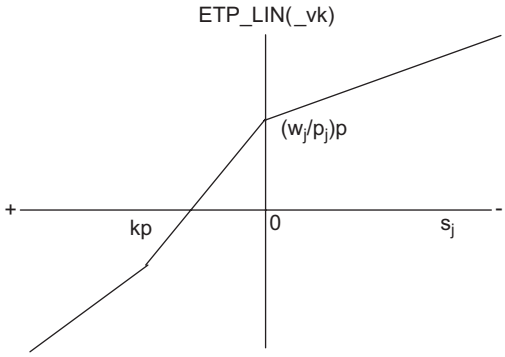

Fig. 5. ETP_LIN(_vk) priority index.

The effectiveness of these two final versions of the ETP dispatching heuristic 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 (for details, see [9]). We considered two different approaches for setting the value of k. In the first approach, denoted as ETP_LIN, k is set at a fixed value. Therefore, the same value of k is used for all instances, and also for all the iterations the heuristic performs for a given instance.

In the second approach, denoted as ETP_LIN_vk, 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 job can be classified as early, critical or tardy, just as previously mentioned in the discussion of the ECTL procedure. 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 the ETP_LIN_vk version. A critical interval of slack values $[0, \max\_slack]$ is first calculated, just as previously described for the ECTL heuristic. A job J_j is classified as critical if $s_j \in [0, \max\_slack]$, and non-critical otherwise. The proportion of critical jobs $prop\_crit$ is then calculated. 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 time complexity of all four versions of the ETP heuristic is $O(n^2)$.

3. The improvement procedures

In this section, we describe several improvement procedures. These procedures are applied as an improvement step after the heuristics have generated a schedule. We consider three simple improvement procedures: adjacent pairwise interchange (API), 3-swaps (3SW) and largest cost insertion (LCI). We also analyse four additional procedures that use a combination of the API, 3SW and LCI methods. These procedures will be denoted as A_L, L_A, 3_L and L_3.

The API procedure, at each iteration, considers in succession all adjacent job positions. A pair of adjacent jobs is then swapped if such an interchange improves the objective function value. This process is repeated until no improvement is found in a complete iteration (i.e., until the sequence is locally optimal and cannot be further improved by adjacent swaps).

The 3SW procedure is similar, but it considers three consecutive job positions instead of an adjacent pair of jobs. All possible permutations of these three jobs are then analysed, and the best configuration is selected. Once more, the procedure is applied repeatedly until no improvement is possible.

The LCI method selects at each iteration the job with the largest objective function value. The selected job is then removed from its position i in the schedule, and inserted at position j, for all $j \neq i$. The best insertion is then performed if it improves the objective function value. This process is repeated until no improving move is found.

The A_L method uses both the API and the LCI procedures as follows. First, the API procedure is applied, as described above. Then, the LCI method is used. These two steps are repeated while the schedule is improved by the LCI procedure (i.e., while the LCI method modifies the sequence generated by the API procedure).

The L_A method also uses both the API and LCI procedures. However, the order in which these two procedures are used is different. Indeed, the L_A procedure applies the LCI method before the API improvement procedure. Finally, the 3_L (L_3) method is quite similar to the A_L (L_A) procedure, since it merely replaces the API improvement procedure with the 3SW method.

4. Computational results

In this section, we first present the set of test problems used in the computational tests, and then describe the preliminary computational experiments. These experiments were performed to determine adequate values for the parameters required by the ECTL and ETP heuristics. Moreover, the performance of the alternative versions of the ECTL and ETP heuristics was also analysed in these initial computational experiments, in order to select the best-performing of those versions. The computational results are then presented. We first compare the heuristic procedures, and then analyse the effectiveness of the improvement step. Finally, we evaluate the heuristic results against 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 10, 15, 20, 25, 30, 40, 50, 75, 100, 250, 500, 750 and 1000 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 [45, 55] and [1, 100], to create low (L) and high (H) variability, respectively. For each job J_j, an integer due date d_j is generated from the uniform distribution $[P(1 - T - R/2), P(1 - T + R/2)]$, where P is the sum of the processing times of all jobs, T is the tardiness factor, set at 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0, and R is the range of due dates, set at 0.2, 0.4, 0.6 and 0.8.

For each combination of problem size n, processing time and penalty variability (var), T and R, 50 instances were randomly generated. Therefore, a total of 1200 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.

4.2. Parameter adjustment tests

In this section, we describe the preliminary computational experiments. These initial experiments were performed to determine appropriate values for the parameters required by the ECTL and ETP dispatching rules. The performance of the alternative versions of the ECTL and ETP heuristics was also analysed, in order to select the best-performing versions. A separate problem set was used to conduct these preliminary experiments. This test set included instances with 25, 50, 100, 250, 500 and 1000 jobs, and contained five instances for each combination of instance size, variability, T and R. The instances in this smaller test set were generated randomly just as previously described for the full problem set.

Extensive computational tests were performed in order to determine adequate values for the parameters used by the ECTL, ETP_LIN and ETP_LIN_vk heuristics. The ECTL_AS and ECTL_LP dispatching rules require a value for the *slack_prop* parameter. We considered the values $\{0.20, 0.25, \ldots, 0.80\}$, and computed the objective function value for each *slack_prop* value and each instance. An analysis of these results showed that the best values for *slack_prop* were in the range [0.25, 0.35]. We then decided to set *slack_prop* at 0.30, since this value provided adequate performance for all instance types.

The ETP_LIN heuristic requires a value for the lookahead parameter k. We considered the values $\{0.5, 0.6, \ldots, 9.0\}$, and the objective function value was then computed for each value of k and each instance. The best performance was achieved with k values in the range [6.0, 9.0]. The lookahead parameter k was then set at 7.0, since this value consistently provided good results.

Table 1
Heuristic version comparison

	n	Low var				High var			
		%imp	<	=	>	%imp	<	=	>
ECTL_AS	25	0.054	11.67	83.33	5.00	0.602	25.00	60.83	14.17
vs	50	0.562	19.17	68.33	12.50	0.190	30.00	50.83	19.17
ECTL_LP	100	1.099	24.17	64.17	11.67	0.196	29.17	46.67	24.17
	250	0.789	25.00	50.83	24.17	0.857	34.17	45.83	20.00
	500	0.991	23.33	57.50	19.17	0.482	38.33	36.67	25.00
	1000	1.085	21.67	59.17	19.17	0.581	37.50	37.50	25.00
ETP_v2	25	0.000	0.00	100.00	0.00	0.149	4.17	94.17	1.67
vs	50	0.000	0.00	100.00	0.00	−0.051	15.00	83.33	1.67
ETP_v1	100	0.000	0.00	100.00	0.00	0.134	15.00	82.50	2.50
	250	0.000	0.00	100.00	0.00	−0.002	5.00	92.50	2.50
	500	0.000	0.00	100.00	0.00	0.000	3.33	96.67	0.00
	1000	0.000	0.00	100.00	0.00	0.003	1.67	97.50	0.83
ETP_LIN_vk	25	0.001	6.67	90.83	2.50	0.129	24.17	70.00	5.83
vs	50	0.000	4.17	93.33	2.50	−0.023	25.00	65.00	10.00
ETP_LIN	100	0.000	1.67	97.50	0.83	0.002	26.67	63.33	10.00
	250	0.000	0.00	100.00	0.00	−0.003	25.83	65.00	9.17
	500	0.000	0.00	100.00	0.00	0.000	10.00	90.00	0.00
	1000	0.000	0.00	100.00	0.00	0.000	6.67	93.33	0.00

The ETP_LIN_vk dispatching heuristic requires a value for the parameters k_L, k_H and *slack_prop*. The following values were considered for these parameters:

k_L : {0.5, 1.0, 1.5, 2.0},

k_H : {7.0, 7.5, . . . , 11.0},

slack_prop : {0.20, 0.25, 0.30, . . . , 0.80}.

We then applied the ETP_LIN_vk heuristic to the test set instances for each combination of these parameter values. A thorough analysis of these results was conducted in order to determine parameter values that provided an adequate performance across all instance types. The following values were then selected for these parameters: $k_L = 0.5$, $k_H = 8.5$ and *slack_prop* = 0.25.

The performance of the alternative versions of the ECTL and ETP heuristics was also analysed in these preliminary computational experiments, in order to select the best-performing versions. The ECTL_AS version was then compared with its ECTL_LP alternative. Also, for the ETP heuristic, we compared ETP_v1 with ETP_v2, while the ETP_LIN fixed lookahead parameter version was evaluated against its dynamic ETP_LIN_vk counterpart. Therefore, we compared the following three ($h1$ vs $h2$) pairs of alternative heuristic versions: (ECTL_AS vs ECTL_LP), (ETP_v2 vs ETP_v1) and (ETP_LIN_vk vs ETP_LIN).

In Table 1, we present the average of the relative improvements in objective function value provided by the $h1$ heuristic over its $h2$ counterpart (%imp), as well as the percentage number of times version $h1$ performs better (<), equal (=) or worse (>) than version $h2$. The relative improvement given by version $h1$ is calculated as $(h2\_of\,v - h1\_of\,v)/h2\_of\,v \times 100$, where $h2\_of\,v$ and $h1\_of\,v$ are the objective function values of the appropriate heuristic versions.

The ECTL_AS version performs better than its ECTL_LP counterpart. The ECTL_AS heuristic provides a positive relative improvement, and gives better results for a somewhat larger number of instances. The performance of the alternative versions of the ETP heuristic, however, is quite similar, particularly for the instances with a low processing time and penalty variability. In fact, when the variability is low, the ETP_v2 and ETP_v1 heuristics provide the same results for all instances, while the ETP_LIN_vk and ETP_LIN procedures only differ for a few of the smaller instances. For problems with a high variability, the average objective function value provided by the alternative versions is nearly identical. However, the ETP_v2 (ETP_LIN_vk) provides better results than its ETP_v1 (ETP_LIN) counterpart for a

Table 2
Heuristic results

Var	Heur	$n = 25$		$n = 100$		$n = 500$		$n = 1000$	
		ofv	%best	ofv	%best	ofv	%best	ofv	%best
L	EDD	101.75	0.42	101.83	0.00	101.86	0.00	101.86	0.00
	WLPT	154.45	0.00	157.01	0.00	157.72	0.00	157.78	0.00
	WSPT	154.01	0.00	156.79	0.00	157.89	0.00	157.71	0.00
	WPT_sj_E	115.31	31.67	116.92	30.67	117.66	31.17	117.73	32.50
	WPT_sj_T	127.47	30.33	128.01	30.50	128.29	29.58	128.16	30.08
	ECTL_AS	100.02	67.33	100.04	48.17	100.05	45.08	100.06	45.50
	ETP_v2	100.00	93.92	100.00	97.17	100.00	100.00	100.00	100.00
	ETP_LIN_vk	100.00	93.33	100.00	96.00	100.00	100.00	100.00	100.00
H	EDD	212.60	0.00	223.35	0.00	227.32	0.00	228.44	0.00
	WLPT	308.12	0.00	329.15	0.00	334.84	0.00	336.63	0.00
	WSPT	299.42	0.00	326.88	0.00	334.81	0.00	337.11	0.00
	WPT_sj_E	163.66	25.08	168.78	26.67	170.48	29.33	171.06	30.25
	WPT_sj_T	277.46	13.92	299.07	16.17	303.36	24.58	304.78	25.00
	ECTL_AS	104.16	40.92	104.66	37.67	104.68	38.92	104.73	38.50
	ETP_v2	100.29	71.83	100.08	74.17	100.01	88.42	100.00	93.42
	ETP_LIN_vk	100.00	75.17	100.00	75.67	100.00	88.75	100.00	96.75

somewhat larger number of instances. Therefore, in the following sections we will only present results for the better performing ECTL_AS, ETP_v2 and ETP_LIN_vk versions.

4.3. Heuristic results

In this section, we present the computational results for the heuristic procedures. In Table 2, we give the average objective function value (ofv) for each heuristic, as well as the percentage number of times a heuristic provides the best result when compared with the other heuristics (%best). The average objective function values are calculated relative to the ETP_LIN_vk heuristic, and are therefore presented as index numbers. More precisely, these values are calculated as $heur\_of\,v/etp\_lin\_vk\_of\,v * 100$, where $heur\_of\,v$ and $etp\_lin\_vk\_of\,v$ are the average objective function values of the appropriate heuristic and the ETP_LIN_vk dispatching rule, respectively.

The best results are given by the ETP_v2 and ETP_LIN_vk dispatching rules. In fact, these heuristics not only provide the lowest average objective function value, but also obtained the best results for a quite large percentage of the instances. Indeed, for some problem sizes with low processing time and penalty variability, these heuristics actually achieve the best results for all the instances. The performance of the ETP_v2 and ETP_LIN_vk rules is quite similar, since both the average objective function values and the percentage of best results are quite close.

The ECTL_AS heuristic also provides an adequate performance, although it is outperformed by the ETP dispatching rules, particularly for the instances with a high variability. In fact, all the heuristic procedures are much closer to the average objective function value of the ETP_LIN_vk heuristic when the variability is low. The ECTL_AS heuristic provides the best results for around 40% of the test instances. Also, the ECTL_AS procedure is quite close to the ETP_LIN_vk heuristic when the variability is low. For instances with a high variability, however, the ECTL_AS heuristic gives an average objective function value that is about 3–4% worse than the ETP rules.

The simple linear WLPT and WSPT rules, as well as the quadratic WPT_s_j_E and WPT_s_j_T dispatching heuristics, perform rather poorly. In fact, these simple rules provide results that are substantially worse than those of the ECTL and ETP heuristics. The EDD rule does provide an average objective function value that is close to the ETP results for instances with a low variability, but its performance is quite poor when the variability is high. Therefore, ignoring earliness and/or tardiness penalties is quite costly in terms of solution quality.

The quadratic WPT_s_j_E (WPT_s_j_T) rule clearly outperforms its WLPT (WSPT) linear counterpart. Hence, the modifications that were introduced in these linear rules, in order to adapt them to a quadratic objective function, have indeed significantly improved their performance. Therefore, it is certainly important to specifically address the

Table 3
Heuristic runtimes (in seconds)

Heur	Low var			High var		
	$n = 500$	$n = 750$	$n = 1000$	$n = 500$	$n = 750$	$n = 1000$
EDD	0.0002	0.0002	0.0003	0.0001	0.0002	0.0004
WLPT	0.0019	0.0028	0.0039	0.0017	0.0029	0.0039
WSPT	0.0019	0.0027	0.0039	0.0017	0.0026	0.0042
WPT_sj_E	0.0032	0.0072	0.0121	0.0030	0.0074	0.0121
WPT_sj_T	0.0023	0.0070	0.0120	0.0027	0.0067	0.0116
ECTL_AS	0.0039	0.0100	0.0178	0.0042	0.0099	0.0176
ETP_v2	0.0107	0.0241	0.0429	0.0110	0.0246	0.0438
ETP_LIN_vk	0.0100	0.0232	0.0409	0.0101	0.0242	0.0420

quadratic component of the cost function, and develop specific procedures, instead of simply using the heuristics previously developed for the linear objective function.

The ECTL_AS heuristic performs significantly better than either of the EDD, WPT_s_j_E or WPT_s_j_T rules. Consequently, a considerable performance improvement can indeed be achieved by selectively using these three simple heuristics, i.e., by choosing at each iteration the rule that is expected to perform better, given the characteristics of the current workload.

In Table 3, we present the heuristic runtimes (in seconds). The heuristic procedures are quite fast, even for the largest instances. The simple linear EDD, WLPT and WSPT rules are the most efficient, since they only require sorting, which can be performed in $O(n \log n)$ time. The quadratic WPT_s_j_E and WPT_s_j_T dispatching heuristics require some additional computational time, as one would expect, given their $O(n^2)$ time complexity. The ECTL and (particularly) the ETP heuristics are computationally more demanding. Nonetheless, even these dispatching rules are quite fast, and capable of solving even quite large instances in less than one second.

4.4. Improvement step results

In this section, we present the computational results for the improvement procedures. In Table 4, we give the average of the relative improvements in the objective function value provided by the improvement procedures. For each heuristic, the relative improvement is calculated as $(ofv - imp\_ofv)/ofv * 100$, where ofv and imp_ofv are the objective function values before and after the application of the appropriate improvement procedure, respectively. In Table 5, we present the percentage number of times the improvement procedures improve the heuristic solution.

From Tables 4 and 5, we can see that the improvement step is usually successful in reducing the heuristic objective function value, particularly for the instances with a high processing time and penalty variability. In fact, both the relative improvement and the percentage of improved instances are larger when the variability is high. Also, the effectiveness of the improvement step decreases with the quality of the schedule generated by the heuristic procedures. Indeed, the relative improvement is quite marginal for the best-performing ETP_v2 and ETP_LIN_vk dispatching rules. Moreover, for these heuristics the improvement procedures only improve a quite small percentage of the largest instances with a low variability. However, the improvement step does reduce the objective function value for most of the high variability problems.

The results provided by the ECTL_AS heuristic are improved by around 7–9% for instances with high processing time and penalty variability. For the low variability instances, the relative improvement is minor, but the improvement step nevertheless reduces the objective function value for over 50% of the test problems. Finally, the worst-performing EDD, WLPT, WSPT, WPT_s_j_E and WPT_s_j_T rules benefit the most from the improvement procedures, and both the relative improvement and the percentage of improved instances are indeed quite large for these heuristics. In fact, these simple rules were quite far in solution quality from the other heuristics, and therefore had a much larger room for improvement.

The LCI procedure is the least effective of the improvement steps. The other procedures are rather close, and provide quite similar relative improvement values. Therefore, the improvement steps that combine the LCI and the API (3SW) procedures do not actually give a noticeably superior performance when compared with the standalone API (3SW)

Table 4
Improvement procedures results—relative improvement

Var	n	Heur	API	3SW	LCI	A_L	L_A	3_L	L_3
L	100	EDD	1.70	1.70	0.18	1.70	1.70	1.70	1.70
		WLPT	42.71	42.71	32.96	42.71	42.71	42.71	42.71
		WSPT	42.70	42.70	32.90	42.70	42.70	42.70	42.70
		WPT_sj_E	14.92	14.92	12.15	14.92	14.92	14.92	14.92
		WPT_sj_T	26.11	26.11	22.42	26.11	26.11	26.11	26.11
		ECTL_AS	0.27	0.27	0.11	0.27	0.27	0.27	0.27
		ETP_v2	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		ETP_LIN_vk	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	1000	EDD	1.72	1.72	0.06	1.72	1.72	1.72	1.72
		WLPT	43.05	43.05	32.55	43.05	43.05	43.05	43.05
		WSPT	43.04	43.04	32.48	43.04	43.04	43.04	43.04
		WPT_sj_E	15.90	15.90	12.91	15.90	15.90	15.90	15.90
		WPT_sj_T	25.49	25.49	21.54	25.49	25.49	25.49	25.49
		ECTL_AS	0.43	0.43	0.21	0.43	0.43	0.43	0.43
		ETP_v2	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		ETP_LIN_vk	0.00	0.00	0.00	0.00	0.00	0.00	0.00
H	100	EDD	55.43	57.53	12.18	55.50	55.33	57.54	57.57
		WLPT	64.75	64.79	48.21	64.79	63.76	64.83	64.59
		WSPT	59.54	64.00	48.70	61.41	62.65	64.13	64.20
		WPT_sj_E	31.29	33.36	22.10	32.35	33.24	33.63	33.69
		WPT_sj_T	53.19	55.38	46.91	53.83	54.77	55.43	55.42
		ECTL_AS	6.42	8.95	0.67	6.86	6.84	9.27	9.21
		ETP_v2	0.04	0.22	0.30	0.39	0.40	0.54	0.55
		ETP_LIN_vk	0.04	0.16	0.30	0.38	0.39	0.47	0.47
	1000	EDD	58.06	59.07	3.75	58.06	58.08	59.07	59.09
		WLPT	65.33	65.34	42.47	65.35	65.07	65.35	65.32
		WSPT	64.31	65.32	42.31	64.45	64.60	65.33	65.30
		WPT_sj_E	34.40	34.60	20.67	34.48	34.56	34.67	34.66
		WPT_sj_T	54.63	55.17	42.44	54.69	54.78	55.18	55.16
		ECTL_AS	8.55	9.62	0.56	8.69	8.63	9.70	9.67
		ETP_v2	0.00	0.02	0.07	0.08	0.08	0.10	0.10
		ETP_LIN_vk	0.00	0.02	0.07	0.08	0.08	0.10	0.09

method. The application of the improvement step virtually eliminates the performance gap between the heuristics, as far as solution quality is concerned. Indeed, after the improvement step, the objective function values are quite close for all heuristics. In fact, the improvement procedures provide a higher improvement for the worst-performing heuristics, so the objective function values are then quite similar after the application of the improvement step.

In Table 6, we present the effect of the T and R parameters on the relative objective function value improvement for the 3SW method and instances with 100 jobs. This effect is significantly different for the several heuristic procedures. For the EDD heuristic, the relative improvement has a somewhat similar magnitude across all instance types.

The relative improvement in the objective function value, however, is clearly increasing with the tardiness factor T for the WLPT and WPT_s_j_E rules. For these heuristics, the improvement is actually quite minor when T is equal to 0.0 or 0.2. However, the relative improvement then increases significantly for higher values of the tardiness factor. This is to be expected, since most jobs will be completed before their due dates when T is equal to 0.0 or 0.2, and the WLPT and WPT_s_j_E rules indeed perform better when most jobs are early. For higher values of the tardiness factor, however, most jobs will be tardy, and the performance of the WLPT and WPT_s_j_E heuristics deteriorates, so there is more room for improvement.

Conversely, the relative improvement is decreasing with T for the WSPT and WPT_s_j_T rules. In fact, these heuristics perform better when most jobs are tardy. As the tardiness factor decreases, the number of early jobs becomes higher, and the potential for improvement consequently increases. For the ECTL_AS and ETP heuristics, the relative improvement

Table 5
Improvement procedures results—improved instances

Var	n	Heur	API	3SW	LCI	A_L	L_A	3_L	L_3
L	100	EDD	100.00	100.00	89.58	100.00	100.00	100.00	100.00
		WLPT	100.00	100.00	99.92	100.00	100.00	100.00	100.00
		WSPT	100.00	100.00	100.00	100.00	100.00	100.00	100.00
		WPT_sj_E	77.25	79.42	50.75	77.25	77.25	79.42	79.42
		WPT_sj_T	79.58	80.92	60.33	79.58	79.58	80.92	80.92
		ECTL_AS	67.67	70.25	1.58	67.67	67.67	70.25	70.25
		ETP_v2	49.00	54.50	0.50	49.00	49.00	54.50	54.50
		ETP_LIN_vk	48.92	54.50	0.42	48.92	48.92	54.50	54.50
	1000	EDD	100.00	100.00	96.08	100.00	100.00	100.00	100.00
		WLPT	100.00	100.00	100.00	100.00	100.00	100.00	100.00
		WSPT	100.00	100.00	100.00	100.00	100.00	100.00	100.00
		WPT_sj_E	67.67	67.67	50.00	67.67	67.67	67.67	67.67
		WPT_sj_T	70.00	70.00	59.92	70.00	70.00	70.00	70.00
		ECTL_AS	54.83	54.83	4.00	54.83	54.83	54.83	54.83
		ETP_v2	2.50	3.42	0.00	2.50	2.50	3.42	3.42
		ETP_LIN_vk	2.50	3.42	0.00	2.50	2.50	3.42	3.42
H	100	EDD	100.00	100.00	98.58	100.00	100.00	100.00	100.00
		WLPT	100.00	100.00	99.50	100.00	100.00	100.00	100.00
		WSPT	100.00	100.00	99.58	100.00	100.00	100.00	100.00
		WPT_sj_E	99.92	99.92	67.58	100.00	100.00	100.00	100.00
		WPT_sj_T	99.75	99.83	72.83	99.75	99.75	99.83	99.83
		ECTL_AS	99.83	99.83	21.08	99.92	99.92	99.92	99.92
		ETP_v2	99.58	99.75	17.92	99.67	99.67	99.83	99.83
		ETP_LIN_vk	99.67	99.75	17.58	99.75	99.75	99.83	99.83
	1000	EDD	100.00	100.00	99.83	100.00	100.00	100.00	100.00
		WLPT	100.00	100.00	100.00	100.00	100.00	100.00	100.00
		WSPT	100.00	100.00	100.00	100.00	100.00	100.00	100.00
		WPT_sj_E	84.00	84.92	63.58	84.00	84.00	84.92	84.92
		WPT_sj_T	81.58	83.42	71.00	81.67	81.67	83.42	83.42
		ECTL_AS	76.42	77.33	17.25	76.42	76.42	77.33	77.33
		ETP_v2	60.17	67.75	12.92	61.17	61.17	68.00	68.00
		ETP_LIN_vk	60.25	67.83	13.00	61.33	61.33	68.00	68.00

is higher for the intermediate values of T, and then decreases as the tardiness factor approaches its extreme values. Actually, when T is equal to 0.0 or 1.0, the improvement step provides little or no improvement. This result is to be expected, since these heuristics consider both earliness and tardiness costs, and are more likely to be closer to the optimum for the extreme T values. In fact, for a tardiness factor of 0.0 (1.0) most jobs will be early (late), and the early/tardy problem is then easier. For the intermediate values of T, there is a greater balance between the number of early and tardy jobs, and the problem becomes much harder.

In Table 7, we present the improvement procedures runtimes (in seconds). The API and 3SW methods, as well as the A_L and 3_L improvement steps, are computationally quite efficient, even for the largest instances. The LCI, L_A and L_3 improvement procedures, however, are somewhat more demanding. The computation time required by the improvement steps is also significantly different for the several heuristics. In fact, the computation time of the improvement procedures is quite minor for the best-performing ETP and ECTL_AS heuristics. For the worst-performing EDD, WLPT, WSPT, WPT_s_j_E and WPT_s_j_T rules, however, the computational effort is significantly higher.

The use of an improvement step, particularly the 3SW or API methods, is therefore recommended, since it usually improves the solution quality with little additional computational effort. The ETP heuristics, followed by the improvement step, are then the procedures of choice. After the application of the improvement step, the objective function values are indeed quite close for all heuristics, as previously mentioned. However, the overall computation time required by

Table 6
3SW relative improvement for instances with 100 jobs

Heur	T	Low var				High var			
		$R = 0.2$	$R = 0.4$	$R = 0.6$	$R = 0.8$	$R = 0.2$	$R = 0.4$	$R = 0.6$	$R = 0.8$
EDD	0.0	2.91	1.73	1.19	0.99	58.67	54.38	51.11	46.30
	0.2	2.91	1.75	1.28	0.99	60.22	62.03	59.23	56.06
	0.4	2.80	1.78	1.24	1.01	61.18	62.32	61.85	60.87
	0.6	2.76	1.76	1.17	0.90	61.36	60.17	62.45	62.63
	0.8	2.96	1.73	1.22	0.95	61.50	61.59	58.62	54.37
	1.0	2.93	1.75	1.22	0.92	56.34	53.30	49.51	44.57
WPT_sj_E	0.0	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.00
	0.2	0.04	0.00	0.00	0.00	0.74	0.06	0.04	0.03
	0.4	2.31	4.27	4.45	0.57	26.74	26.00	16.69	5.21
	0.6	9.35	24.23	44.32	63.75	52.13	61.60	71.84	82.14
	0.8	11.35	27.58	39.04	47.07	54.47	62.95	68.17	70.39
	1.0	8.93	16.84	24.14	29.78	48.88	50.06	51.52	51.00
WPT_sj_T	0.0	16.03	23.22	29.80	35.15	74.39	73.32	74.66	74.05
	0.2	23.57	36.50	46.71	52.21	78.74	83.20	85.25	86.03
	0.4	32.25	54.61	70.91	81.78	80.88	88.34	92.93	96.25
	0.6	20.35	38.35	43.50	18.25	66.53	76.93	83.82	83.04
	0.8	3.29	0.12	0.00	0.00	15.33	10.25	3.17	1.99
	1.0	0.00	0.00	0.00	0.00	0.01	0.02	0.03	0.00
ECTL_AS	0.0	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.00
	0.2	0.00	0.00	0.00	0.00	0.17	0.05	0.04	0.03
	0.4	0.02	0.01	0.01	0.21	5.11	6.09	5.25	4.46
	0.6	0.07	0.05	0.10	5.75	12.77	19.25	31.12	48.73
	0.8	0.02	0.04	0.09	0.18	9.69	18.90	23.45	27.14
	1.0	0.00	0.00	0.00	0.01	0.01	0.37	0.50	1.73

the heuristic procedure together with the improvement step is lower for the ETP dispatching rules. The ETP_v2 and ETP_LIN_vk procedures require a somewhat higher computation time, but the improvement step is much faster for these heuristics. In fact, the ETP heuristics generate better sequences, so the improvement step does not have as much room for improvement. The higher computational effort of the heuristic procedure is then more than compensated by the lower computation time required by the improvement step.

4.5. Comparison with optimum results

In this section, we compare the heuristic results with the optimum objective function values determined in [4] for instances with up to 20 jobs; results obtained after the application of the 3SW improvement procedure are indicated by appending "+ 3SW" to the heuristic identifiers. In Table 8, 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.

The results given in Table 8 show that the problem is much harder when the processing time and penalty variability is high. Indeed, the heuristics are significantly closer to the optimum for instances with a low variability. The ETP and ECTL_AS procedures perform extremely well for problems with low variability, even before the application of the improvement step. In fact, these heuristics are not only quite close to the optimum, but also provide an optimum solution for over 50% of the test instances. The other heuristics, however, are far from the optimum (with the exception of the EDD rule). After the application of the 3SW improvement procedure, all the heuristics perform exceptionally well, since they only fail to provide an optimum solution for about 2–3% of the instances.

When the variability is high, and before the application of the 3SW procedure, the best-performing ETP heuristics are on average about 7–9% above the optimum. The other procedures perform much worse, and even the ECTL_AS

Table 7
Improvement procedures runtimes (in seconds)

Var	n	Heur	API	3SW	LCI	A_L	L_A	3_L	L_3
L	500	EDD	0.015	0.056	0.175	0.037	0.213	0.077	0.253
		WLPT	0.094	0.330	3.851	0.115	3.917	0.352	4.026
		WSPT	0.102	0.359	3.852	0.124	3.917	0.382	4.030
		WPT_sj_E	0.035	0.124	1.225	0.056	1.262	0.145	1.307
		WPT_sj_T	0.052	0.183	1.937	0.073	1.974	0.204	2.026
		ECTL_AS	0.001	0.004	0.023	0.020	0.043	0.023	0.047
		ETP_v2	0.000	0.002	0.022	0.022	0.045	0.023	0.047
		ETP_LIN_vk	0.001	0.002	0.022	0.022	0.045	0.024	0.048
	1000	EDD	0.061	0.218	0.939	0.146	1.086	0.303	1.241
		WLPT	0.377	1.322	29.631	0.462	29.874	1.406	30.307
		WSPT	0.410	1.436	29.866	0.496	30.110	1.523	30.585
		WPT_sj_E	0.139	0.492	9.388	0.224	9.526	0.578	9.698
		WPT_sj_T	0.206	0.723	14.924	0.291	15.079	0.809	15.269
		ECTL_AS	0.003	0.013	0.115	0.088	0.205	0.098	0.216
		ETP_v2	0.001	0.005	0.088	0.087	0.175	0.090	0.182
		ETP_LIN_vk	0.001	0.004	0.088	0.087	0.175	0.090	0.182
H	500	EDD	0.055	0.208	0.394	0.072	0.465	0.223	0.614
		WLPT	0.084	0.303	1.758	0.107	1.831	0.329	1.965
		WSPT	0.095	0.387	1.774	0.141	1.848	0.411	2.008
		WPT_sj_E	0.032	0.131	0.797	0.058	0.839	0.155	0.893
		WPT_sj_T	0.068	0.266	1.620	0.094	1.668	0.288	1.757
		ECTL_AS	0.005	0.026	0.027	0.030	0.050	0.050	0.073
		ETP_v2	0.001	0.003	0.022	0.024	0.041	0.027	0.045
		ETP_LIN_vk	0.001	0.003	0.022	0.025	0.041	0.027	0.045
	1000	EDD	0.225	0.830	2.018	0.294	2.306	0.898	2.906
		WLPT	0.337	1.224	12.648	0.438	12.917	1.320	13.490
		WSPT	0.399	1.564	12.439	0.551	12.753	1.657	13.436
		WPT_sj_E	0.134	0.532	5.748	0.238	5.905	0.633	6.126
		WPT_sj_T	0.279	1.059	11.454	0.389	11.667	1.154	12.050
		ECTL_AS	0.022	0.105	0.132	0.128	0.226	0.206	0.307
		ETP_v2	0.001	0.006	0.095	0.103	0.165	0.107	0.175
		ETP_LIN_vk	0.001	0.006	0.095	0.103	0.166	0.108	0.175

heuristic gives results that are around 20% above the optimum. After the improvement step, however, the heuristic performance is quite satisfactory. Indeed, the heuristic procedures not only achieve a 4–5% average deviation from the optimum, but also provide an optimum solution for over 60% of the instances.

These results also further validate the effectiveness of the improvement procedures. As mentioned in the previous section, both the relative improvement and the number of improved instances were marginal for the ETP heuristics on instances with low processing time and penalty variability. However, it can now be seen that in this case there was indeed little room for improvement. In fact, the ETP heuristics are already quite close to the optimum for instances with low variability, even before the application of the improvement step. For all the other cases, the improvement step not only significantly reduces the average deviation from the optimum objective function values, but also greatly increases the number of instances for which an optimum solution is found.

In Table 9, we present the effect of the T and R parameters on the relative deviation from the optimum for instances with 20 jobs. Table 9 provides the relative deviation values before the application of the improvement step. In fact, after the application of the improvement procedures, the relative deviation is similar for all parameter combinations. The results in Table 9 show that the effect of these parameters on the relative deviation from the optimum is significantly different for the several heuristic procedures. Also, these results are in line with those reported in the previous section for the improvement step.

The relative deviation from the optimum is relatively similar across all instance types for the EDD rule. However, the tardiness factor T has a significant effect for the other heuristics. The WLPT and WPT_s_j_E rules are quite close to

Table 8
Comparison with optimum objective function values

Var	Heur	$n = 10$		$n = 15$		$n = 20$	
		%dev	%opt	%dev	%opt	%dev	%opt
L	EDD	1.499	11.50	1.612	3.00	1.630	1.08
	WLPT	139.038	0.00	144.043	0.00	157.997	0.00
	WSPT	146.252	0.00	148.582	0.00	152.951	0.00
	WPT_sj_E	15.167	26.08	17.168	21.00	18.749	18.50
	WPT_sj_T	73.892	23.58	67.701	20.83	65.305	16.92
	ECTL_AS	0.090	67.42	0.085	51.00	0.110	38.42
	ETP_v2	0.045	71.50	0.027	59.17	0.019	49.50
	ETP_LIN_vk	0.047	70.83	0.029	58.50	0.018	50.00
	EDD + 3SW	0.005	98.75	0.003	98.08	0.001	96.67
	WLPT + 3SW	0.007	98.58	0.002	98.17	0.002	96.92
	WSPT + 3SW	0.008	98.17	0.007	96.92	0.003	96.33
	WPT_sj_E + 3SW	0.009	98.17	0.003	97.83	0.002	97.08
	WPT_sj_T + 3SW	0.003	98.92	0.006	97.17	0.001	96.08
	ECTL_AS + 3SW	0.007	98.50	0.001	98.08	0.002	96.75
	ETP_v2 + 3SW	0.007	98.50	0.002	97.92	0.002	96.58
	ETP_LIN_vk + 3SW	0.007	98.50	0.002	97.67	0.003	96.50
H	EDD	145.355	0.08	159.342	0.08	156.084	0.00
	WLPT	400.889	0.08	465.861	0.00	515.314	0.00
	WSPT	516.345	0.33	550.789	0.00	580.541	0.00
	WPT_sj_E	81.311	11.92	89.703	6.17	98.283	3.75
	WPT_sj_T	458.122	6.67	478.932	4.58	488.957	2.58
	ECTL_AS	21.686	19.25	19.687	10.58	21.134	5.83
	ETP_v2	11.485	27.33	9.199	15.33	8.573	9.08
	ETP_LIN_vk	8.103	27.75	7.263	15.50	6.850	8.83
	EDD + 3SW	7.361	77.17	8.387	65.08	9.897	58.00
	WLPT + 3SW	3.227	84.67	3.724	75.92	4.424	70.08
	WSPT + 3SW	13.871	71.67	16.570	57.42	13.714	52.42
	WPT_sj_E + 3SW	6.175	77.42	7.233	66.42	7.517	59.42
	WPT_sj_T + 3SW	10.167	76.83	8.497	66.25	9.493	59.58
	ECTL_AS + 3SW	5.535	78.67	5.542	69.00	6.569	62.83
	ETP_v2 + 3SW	4.690	80.75	5.168	70.67	5.892	64.83
	ETP_LIN_vk + 3SW	3.735	81.83	4.445	70.92	5.179	65.75

the optimum when the tardiness factor is low, but their performance substantially deteriorates for the larger T values. This is to be expected, since the WLPT and WPT_s_j_E rules are indeed more suited to problems where most jobs are early. Conversely, the WSPT and WPT_s_j_T heuristics focus on the tardiness cost, and these rules do indeed perform much better for the higher values of the tardiness factor T.

The effect of the parameter T is similar for the ECTL_AS and ETP procedures. These dispatching rules are quite close to the optimum for the extreme values of T, and their performance deteriorates for the intermediate values of the tardiness factor. Again, this result is to be expected, since these heuristics consider both earliness and tardiness costs. In fact, for the intermediate values of T there is a greater balance between the number of early and tardy jobs, and the problem is then much harder.

5. Conclusion

In this paper, we considered the single machine scheduling problem with quadratic earliness and tardiness costs and no machine idle time. We proposed several dispatching heuristics, and analysed their performance on a wide range of instances. The heuristics included simple but widely used scheduling rules, as well as adaptations of those rules to a

Table 9
Relative deviation from the optimum for instances with 20 jobs

Heur	T	Low var				High var			
		$R = 0.2$	$R = 0.4$	$R = 0.6$	$R = 0.8$	$R = 0.2$	$R = 0.4$	$R = 0.6$	$R = 0.8$
EDD	0.0	3.14	1.96	1.30	0.94	167.97	125.60	106.13	93.92
	0.2	2.52	1.64	1.31	0.95	194.49	187.18	159.06	134.09
	0.4	3.24	1.72	1.00	0.94	229.61	245.12	206.55	193.06
	0.6	2.33	1.37	0.94	0.81	216.99	184.92	191.01	140.83
	0.8	2.76	1.65	1.10	0.85	173.79	146.38	136.72	115.97
	1.0	2.83	1.70	1.15	0.96	121.06	108.23	96.16	71.19
WPT_sj_E	0.0	0.01	0.00	0.00	0.00	0.08	0.15	0.14	0.16
	0.2	0.12	0.02	0.01	0.02	2.81	2.01	1.13	0.83
	0.4	2.41	4.83	6.89	5.60	63.90	96.54	59.02	77.26
	0.6	10.48	29.11	54.49	83.58	165.78	254.11	275.41	324.83
	0.8	10.77	30.80	51.06	67.70	150.29	151.84	173.94	184.38
	1.0	9.28	18.47	27.08	37.24	88.47	90.35	102.82	92.53
WPT_sj_T	0.0	20.34	31.96	44.50	59.65	346.75	290.41	299.55	317.05
	0.2	31.50	57.02	94.97	131.16	467.39	627.33	718.25	705.74
	0.4	46.12	120.29	247.58	460.68	536.53	1068.49	1554.60	3115.02
	0.6	25.82	48.14	79.09	65.35	181.58	274.56	530.44	626.13
	0.8	2.87	0.27	0.00	0.00	15.46	20.86	15.24	22.22
	1.0	0.01	0.00	0.00	0.00	0.18	0.22	0.49	0.44
ECTL_AS	0.0	0.01	0.00	0.00	0.00	0.08	0.15	0.14	0.16
	0.2	0.05	0.01	0.01	0.02	1.38	1.63	1.12	0.83
	0.4	0.07	0.04	0.06	0.44	23.49	32.35	26.61	38.97
	0.6	0.11	0.07	0.11	1.29	54.31	61.30	66.28	82.53
	0.8	0.04	0.04	0.07	0.16	24.46	28.64	30.80	27.98
	1.0	0.01	0.00	0.00	0.01	0.18	0.30	2.07	1.46

quadratic objective function. We also proposed several heuristics that specifically address not only the early and tardy penalties, but also the quadratic cost function.

Dispatching heuristics are widely used in practice and, in fact, most real scheduling systems are either based on dispatching rules, or at least use them to some degree. For large instances, dispatching procedures are sometimes the only heuristic approach capable of generating solutions within reasonable computation times. Dispatching rules are also used by other heuristic procedures, e.g., they are often used to generate the initial sequence required by local search or metaheuristic algorithms.

Extensive experiments were performed to determine appropriate values for the parameters required by some of the heuristics. We also proposed several improvement procedures. These procedures are used as an improvement step, once a schedule has been generated by the heuristics.

The best results were given by the ETP heuristics. These procedures are quite close to the optimum for the low variability instances, and provide results that are about 7–9% above the optimum when the variability is high. The modifications that were introduced in existing rules, in order to adapt them to a quadratic objective function, significantly improved the heuristic performance. Therefore, it is indeed important to specifically address the quadratic feature of the cost function, and develop specific heuristics, instead of simply using procedures originally developed for a linear objective function. The dispatching heuristics were quite fast, and are capable of solving even very large instances in less than one second on a personal computer.

The use of an improvement step, particularly the 3SW or API methods, is recommended, since it usually improved the solution quality with little additional computational effort. After the application of the improvement step, the heuristic procedures are optimal for nearly all instances with a low variability, and give results that are only 4–5% above the optimum when the variability is high. The ETP heuristics, followed by the improvement step, are then the procedures of choice. In fact, after the application of the improvement step, the objective function values are quite close for all

heuristics. However, the overall computation time required by the heuristic procedure together with the improvement step is lower for the ETP dispatching rules.

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

[1] Wagner BJ, Davis DJ, Kher H. The production of several items in a single facility with linearly changing demand rates. Decision Sciences 2002;33:317–46.
[2] Korman K. A pressing matter. Video 1994; 46–50.
[3] 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.
[4] Valente JMS. An exact approach for single machine scheduling with quadratic earliness and tardiness penalties. Working Paper 238, Faculdade de Economia, Universidade do Porto, Portugal, 2007.
[5] Abdul-Razaq T, Potts CN. Dynamic programming state-space relaxation for single machine scheduling. Journal of the Operational Research Society 1988;39:141–52.
[6] Li G. Single machine earliness and tardiness scheduling. European Journal of Operational Research 1997;96:546–58.
[7] Liaw CF. A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. Computers and Operations Research 1999;26:679–93.
[8] Valente JMS, Alves RAFS. Improved lower bounds for the early/tardy scheduling problem with no idle time. Journal of the Operational Research Society 2005;56:604–12.
[9] Ow PS, Morton TE. The single machine early/tardy problem. Management Science 1989;35:177–91.
[10] Valente JMS, Alves RAFS. Improved heuristics for the early/tardy scheduling problem with no idle time. Computers and Operations Research 2005;32:557–69.
[11] Valente JMS, Alves RAFS. Filtered and recovering beam search algorithms for the early/tardy scheduling problem with no idle time. Computers and Industrial Engineering 2005;48:363–75.
[12] Sourd F, Kedad-Sidhoum S. The one machine problem with earliness and tardiness penalties. Journal of Scheduling 2003;6:533–49.
[13] Lakshminarayan S, Lakshmanan R, Papineau RL, Rochete R. Optimal single-machine scheduling with earliness and tardiness penalties. Operations Research 1978;26:1079–82.
[14] Verma S, Dessouky M. Single-machine scheduling of unit-time jobs with earliness and tardiness penalties. Mathematics of Operations Research 1998;23:930–43.
[15] Feldmann M, Biskup D. Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. Computers and Industrial Engineering 2003;44:307–23.
[16] Kim YD, Yano CA. Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates. Naval Research Logistics 1994;41:913–33.
[17] Ventura JA, Radhakrishnan S. Single machine scheduling with symmetric earliness and tardiness penalties. European Journal of Operational Research 2003;144:598–612.
[18] Sundararaghavan PS, Ahmed MU. Minimizing the sum of absolute lateness in single-machine scheduling. Naval Research Logistics Quarterly 1984;31:325–33.
[19] Schaller J. Single machine scheduling with early and quadratic tardy penalties. Computers and Industrial Engineering 2004;46:511–32.
[20] Gupta SK, Sen T. Minimizing a quadratic function of job lateness on a single machine. Engineering Costs and Production Economics 1983;7: 187–94.
[21] Su LH, Chang PC. A heuristic to minimize a quadratic function of job lateness on a single machine. International Journal of Production Economics 1998;55:169–75.
[22] Schaller J. Minimizing the sum of squares lateness on a single machine. European Journal of Operational Research 2002;143:64–79.
[23] Sen T, Dileepan P, Lind MR. Minimizing a weighted quadratic function of job lateness in the single machine system. International Journal of Production Economics 1995;42:237–43.
[24] Baker KR, Scudder GD. Sequencing with earliness and tardiness penalties: a review. Operations Research 1990;38:22–36.
[25] Kanet JJ, Sridharan V. Scheduling with inserted idle time: problem taxonomy and literature review. Operations Research 2000;48:99–110.