Heuristics for the single machine scheduling problem with early and quadratic tardy penalties

Jorge M.S. Valente

LIAAD, Faculdade de Economia
Universidade do Porto,
Rua Dr. Roberto Frias,
Porto 4200-464, Portugal
Fax: +351-22-550-50-50
E-mail: jvalente@fep.up.pt

Abstract: This paper considers the single machine scheduling problem with linear earliness and quadratic tardiness costs, and no machine idle time. Several dispatching heuristics are proposed, and their performance is analysed on a wide range of instances. The heuristics include simple scheduling rules, as well as a procedure that takes advantage of the strengths of these rules. Linear early/quadratic tardy dispatching rules are also considered, as well as a greedy-type procedure. Extensive experiments are performed to determine appropriate values for the parameters required by some of the heuristics. The computational tests show that the best results are given by the linear early/quadratic tardy dispatching rule. This procedure is also quite efficient, and can quickly solve even very large instances.
[Received 15 December 2006; Revised 20 July 2007; Accepted 24 July 2007]

Keywords: heuristics; scheduling; single machine; early penalties; quadratic tardy penalties; no machine idle time; dispatching rules.

Reference to this paper should be made as follows: Valente, J.M.S. (2007) 'Heuristics for the single machine scheduling problem with early and quadratic tardy penalties', *European J. Industrial Engineering*, Vol. 1, No. 4, pp.431–448.

Biographical notes: Jorge M.S. Valente is an Assistant Professor of Operations Research at the Faculty of Economics, University of Porto, Portugal. He received a PhD in Management Science and an MS in Economics from the University of Porto. His current research interests include production scheduling, combinatorial optimisation, heuristic techniques and agent-based computational economics.

1 Introduction

This paper considers a single machine scheduling problem with linear earliness and quadratic 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. For a given schedule, the earliness and tardiness of J_j are defined as $E_j = \max\{0, d_j - C_j\}$ and

$T_j = \max\{0, C_j - d_j\}$, respectively, where C_j is the completion time of J_j. The objective is to find a schedule that minimises the sum of linear earliness and quadratic tardiness costs $\sum_{j=1}^{n}\left(E_j + T_j^2\right)$, subject to the constraint that no machine idle time is allowed.

Scheduling models with a single processor may appear to arise infrequently in practice. However, this scheduling environment does indeed occur in several activities (for a recent example in the chemical industry, see Wagner et al. (2002)). Moreover, the performance of many production systems is quite often dictated by the quality of the schedules for a single bottleneck machine. Models with a single processor are then 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.

Scheduling models with both earliness and tardiness penalties are compatible with the philosophy of Just-In-Time (JIT) production. The JIT production philosophy emphasises 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. Earliness/tardiness problems are also compatible with a recent trend in industry, namely supply chain management. This approach seeks to integrate the flow of materials from the suppliers to the customers, in order to improve the efficiency of the supply chain and to provide a better service to the end-user. The adoption of this approach has caused organisations to view early deliveries, in addition to tardy deliveries, as undesirable.

Linear earliness and quadratic tardiness costs are considered in this paper. On the one hand, early deliveries or early completions of jobs result in unnecessary inventory that ties up cash, as well as space and resources required to maintain and manage the inventory. These costs tend to be proportional to the quantity of inventory held, and therefore a linear penalty is used for early jobs.

On the other hand, late deliveries can result in lost sales and loss of goodwill, as well as disruptions and delays in stages further down the supply chain or production line. A quadratic penalty is considered for the tardy jobs, instead of the more usual linear tardiness or maximum tardiness functions. As described in Sun et al. (1999), the quadratic penalty may be preferable to these two other tardiness measures for the following reasons.

Firstly, the maximum tardiness measure does not distinguish between schedules where tardiness occurs for all jobs, or only one, as long as the maximum tardiness is the same. Secondly, when a linear tardiness is used, it is possible that a single or only a few jobs contribute the majority of the cost, without regard to how the overall tardiness is distributed. In fact, the linear tardiness criterion does not differentiate between sequences where all jobs are only a little tardy, or a single job is extremely late, as long as the total cost is equal. The quadratic penalty overcomes these problems, and provides a more robust performance measure. Moreover, a quadratic tardiness penalty is also appropriate in practice. Indeed, the tardiness of a job is an important attribute of service quality. Also, a customer's dissatisfaction tends to increase quadratically with the tardiness, as proposed in the loss function of Taguchi (1986).

In this paper, it is assumed that no machine idle time is allowed. This assumption is 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 to meet 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 set-up costs or times. Some specific examples of production settings where 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.

This problem has been previously considered by Valente (to appear). He proposed a lower bounding procedure based on a relaxation of the job completion times, as well as a branch-and-bound algorithm. The corresponding problem with inserted idle time was studied by Schaller (2004), who presented a timetabling procedure to optimally insert idle time in a given sequence, as well as a branch-and-bound procedure and simple and efficient heuristic algorithms.

The single machine early/tardy problem with linear earliness and tardiness costs $\sum_{j=1}^{n}\left(E_j + T_j\right)$ has also been previously considered by Garey et al. (1988), Kim and Yano (1994) and Schaller (2007). Garey et al. (1988) show that the problem is NP-hard, and propose a timetabling procedure. Kim and Yano (1994) present some properties of optimal solutions, and use them to develop both optimal and heuristic algorithms. Schaller (2007) develops a new lower bound and a new dominance condition, and also shows how to strengthen the lower bounds proposed by Kim and Yano (1994). The computational tests show that the new lower bounds improve the efficiency of a branch-and-bound algorithm.

Valente and Alves (to appear) presented several heuristics for the problem with quadratic earliness and tardiness costs and job-dependent penalties $\sum_{j=1}^{n}\left(h_j E_j^2 + w_j T_j^2\right)$, and no machine idle time. The minimisation 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 previously considered. Gupta and Sen (1983) presented a branch-and-bound algorithm and a heuristic rule for the problem with no idle time. Su and Chang (1998) and Schaller (2002) considered the insertion of idle time, and proposed timetabling procedures and heuristic algorithms. Sen et al. (1995) 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 (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. Also, a recent survey of multicriteria scheduling problems is given in Hoogeveen (2005). This survey also considers and reviews problems with earliness and tardiness penalties.

In this paper, several dispatching heuristics are proposed, and their performance is analysed on a large set of instances. Three simple but widely used scheduling rules are considered, and an adaptation of one of those rules to a quadratic tardiness objective function is proposed. A heuristic that tries to take advantage of the strengths of the best-performing of these simple rules is also developed. Modified versions of early/tardy dispatching procedures originally proposed for the weighted problem with fully linear costs are also presented. These heuristics have been suitably adapted, in order to take into account the quadratic tardiness cost, as well as the non-weighted nature of the considered problem. Finally, a greedy-type heuristic procedure is also presented. Extensive computational experiments are performed in order to determine appropriate values for the parameters required by some of the heuristics.

The remainder of this paper is organised as follows. The heuristics are described in Section 2. In Section 3, the computational results are presented. Finally, some concluding remarks are provided in Section 4.

2 The heuristics

2.1 Simple linear dispatching rules

Three simple scheduling rules are considered, namely the Longest Processing Time (LPT), Earliest Due Date (EDD) and Shortest Processing Time (SPT) heuristics. The LPT (SPT) rule schedules the jobs in non-increasing (non-decreasing) order of their processing times, while the EDD heuristic sequences the jobs in non-decreasing order of their due dates. These rules only require sorting, and their time complexity is then $O(n \log n)$.

These heuristics are considered for two major reasons. On the one hand, these rules are quite well-known and widely used in many production settings. Therefore, it seems reasonable to include them for comparison purposes.

On the other hand, these rules have some interesting properties for the related problem with a fully linear objective function $\sum_{j=1}^{n} \left(E_j + T_j\right)$. Indeed, the LPT heuristic is particularly adequate to problems where most jobs will be completed early. In fact, the LPT sequence is optimal if it does not contain any tardy jobs. Conversely, the SPT rule is optimal if it generates a schedule with no early jobs. Therefore, this rule is appropriate for problems where most jobs will be tardy. Finally, the EDD heuristic usually performs better than either the LPT or SPT rules when the number of early and tardy jobs is relatively balanced.

Therefore, each one of these simple rules can perform quite well, under the appropriate circumstances, for the problem with a completely linear objective function. For this reason, it seems appropriate to analyse their performance for the problem with a quadratic tardiness cost.

2.2 Simple quadratic dispatching rule

The SPT rule is locally optimal, under the appropriate conditions, for the linear total tardiness problem $\sum_{j=1}^{n} T_j$. In fact, if two adjacent jobs are always tardy, regardless of their order, it is optimal to schedule those jobs in SPT order. In this section, a dispatching rule derived from a local optimality condition for the quadratic tardiness problem $\sum_{j=1}^{n} T_j^2$ is presented. Therefore, this heuristic is an adaptation of the SPT rule to a quadratic objective function. This procedure can also be seen as an adaptation of the WPT_s_j_T dispatching rule proposed by Valente and Alves (to appear) for the problem with quadratic early/tardy costs and job-dependent penalties.

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

$$\left(\frac{1}{p_i}\right)\left[p_j + 2\left(t + p_i - d_i\right)\right] \geq \left(\frac{1}{p_j}\right)\left[p_i + 2\left(t + p_j - d_j\right)\right]$$

where job J_i immediately precedes job J_j and t is the start time of job J_i.

Proof: The condition can be established using simple interchange arguments. For the sake of brevity, the details are omitted.

Theorem 1 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 be interpreted as the priority of job J_i with respect to job J_j (job J_j with respect to job J_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. Therefore, the priority index of job J_j at time t, denoted as $I_j(t)$, can be calculated as:

$$I_j(t) = \left(\frac{1}{p_j}\right)\left[\overline{p} + 2\max\left(t + p_j - d_j, 0\right)\right]$$

At each iteration, the SPT_s_j dispatching rule selects the unscheduled job with the largest priority. The priority index of the SPT_s_j heuristic includes both a SPT component and a slack (s_j) related component (the slack of job J_j is defined as $s_j = d_j - t - p_j$). When a job is early, the SPT_s_j heuristic is equivalent to the SPT rule, since the priority of job J_j is then equal to $\left(1/p_j\right)\overline{p}$. When a job is tardy, however, the SPT ratio $\left(1/p_j\right)$ is modified by a slack-related component, and the priority increases with the job's tardiness.

The SPT_s_j dispatching heuristic is particularly suited to 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 SPT_s_j rule is essentially an adaptation of the SPT heuristic to a quadratic tardiness objective. The time complexity of the SPT_s_j heuristic is $O\left(n^2\right)$.

The following numerical example will be used to illustrate the proposed heuristics. Consider an instance with six jobs, with processing times 8, 10, 6, 4, 3 and 5, and due dates 15, 10, 9, 2, 12 and 17, respectively. In the first iteration, at time $t = 0$, the average processing time $\overline{p}$ of the remaining unscheduled jobs is 6.3333. The priorities of the six available jobs are 0.7917, 0.6333, 1.0556, 2.8533, 2.1111 and 0.9048. Job 4 has the largest priority, and is then selected for processing. After the subsequent iterations are performed, the final sequence 4-5-3-2-1-6, with objective function value 891, is then obtained.

2.3 The CS heuristic

Early computational tests were performed with the LPT, EDD, SPT and SPT_s_j dispatching rules. These tests showed that the SPT_s_j heuristic performed better than the SPT rule. Moreover, the preliminary tests also showed that the best results were given by the EDD (SPT_s_j) heuristic for problems where most jobs were early (tardy). The LPT heuristic was outperformed by the other procedures, even for instances where most jobs were early. In fact, the LPT heuristic focuses on minimising the earliness costs, and completely disregards the tardiness component of the objective function. This means that the LPT sequence may contain a few jobs that are quite tardy, even for instances where most jobs will indeed be early. Since the objective function penalty for tardiness is much higher than the penalty for earliness, the LPT sequence will have a large cost, even though it minimises the earliness component.

In this section, a heuristic (denoted as Critical Slack (CS)) that tries to take advantage of the strengths of the EDD and SPT_s_j rules is presented. At each iteration, the CS heuristic uses one of these two rules to choose the next job. Indeed, the CS procedure selects the rule that is expected to provide the best performance, given the characteristics of the current workload.

The CS heuristic classifies the current workload as non-tardy or tardy. When most jobs have large slacks, the current workload is classified as non-tardy. Conversely, a tardy load consists mainly of jobs with low slacks. At each iteration, the CS heuristic analyses the characteristics of the current set of unscheduled jobs, and classifies the workload as either non-tardy or tardy. Then, the CS procedure selects the EDD (SPT_s_j) rule when the load is non-tardy (tardy).

Two versions of the CS heuristic are considered. These versions share the same basic framework, and differ only in the criterion used to classify the workload as non-tardy or tardy. In both versions, a CS value crit_slack is first calculated. This critical value is calculated as crit_slack = slack_prop $\times$ n_U $\times$ $\overline{p}$, where n_U is the number of unscheduled jobs, and $0 \leq$ slack_prop < 1 is a user-defined parameter. Therefore, the critical slack value is then a proportion slack_prop of the total processing time of the currently unscheduled jobs.

The CS_AS version calculates the average slack $\overline{s}$ of the remaining unscheduled jobs. The workload is then classified as non-tardy (tardy) if $\overline{s} >$ crit_slack ($\overline{s} \leq$ crit_slack). In the CS_LP version, on the other hand, each job is first classified as non-tardy or tardy. A job is said to be non-tardy (tardy) if $s_j >$ crit_slack ($s_j \leq$ crit_slack). The proportion of non-tardy and tardy jobs is then calculated, and the current workload is classified as non-tardy (tardy) if the percentage of non-tardy (tardy) jobs is the largest. The time complexity of both versions of the CS heuristic is $O\left(n^2\right)$.

Consider the numerical example that was previously presented, and assume slack_prop = 0.2. In the first iteration, the critical slack value is equal to 7.6. In the CS_AS version, the average slack $\overline{s}$ of the remaining unscheduled jobs is 4.5. Since $\overline{s} <$ crit_slack, the load is classified as critical, and the SPT_s_j rule is used to select the next job. The priorities of the six unscheduled jobs are 0.7917, 0.6333, 1.0556, 2.8533, 2.1111 and 0.9048. Job 4 is selected for processing, since it has the largest priority. After the subsequent iterations are performed, the final sequence 4-5-3-2-1-6, with objective function value 891, is then obtained. The same final sequence is also generated by the CS_LP version.

2.4 *Linear early/quadratic tardy dispatching rules*

Ow and Morton (1989) developed two early/tardy dispatching rules, denoted as LINET and EXPET, for the fully linear problem with job-dependent earliness and tardiness penalties $\sum_{j=1}^{n}\left(h_j E_j + w_j T_j\right)$ (where h_j and w_j are the job-specific earliness and tardiness penalties, respectively). In this section, adaptations of these rules to the linear earliness and quadratic tardiness problem are proposed. Therefore, the heuristics proposed by Ow and Morton have been suitably modified, to take into account the quadratic tardiness cost, as well as the fact that $h_j = w_j = 1$. The proposed heuristics are denoted by EQTP_LIN and EQTP_EXP, where EQTP stands for Earliness and Quadratic Tardiness Penalties.

Both versions of the EQTP dispatching rule calculate a priority index for each remaining job every time the machine becomes available, and the job with the highest priority is selected to be processed next. Let $I_j(t)$ denote the priority index of job J_j at time t. The EQTP_LIN version uses the following priority index $I_j(t)$:

$$I_j(t) = \begin{cases} \left(1/p_j\right)\left[\overline{p} + 2\left(t + p_j - d_j\right)\right] & \text{if} \quad s_j \leq 0 \\ \left(\overline{p}/p_j\right) - \left(1/p_j\right)\left(\overline{p} + 1\right) s_j / k\overline{p} & \text{if} \quad 0 < s_j < k\overline{p} \\ -\left(1/p_j\right) & \text{otherwise} \end{cases}$$

where k is a lookahead parameter and s_j and $\overline{p}$ are as previously defined.

The EQTP_EXP rule instead uses the following priority index:

$$I_j(t) = \begin{cases} \left(1/p_j\right)\left[\overline{p} + 2\left(t + p_j - d_j\right)\right] & \text{if} \quad s_j \leq 0 \\ \left(\overline{p}/p_j\right)\exp\left[-\left(\overline{p}+1\right)s_j/k\overline{p}\right] & \text{if} \quad 0 < s_j < \left[\overline{p}/\left(\overline{p}+1\right)\right]k\overline{p} \\ \left(1/p_j\right)^{-2}\left[\left(\overline{p}/p_j\right) - \left(1/p_j\right)\left(\overline{p}+1\right)s_j/k\overline{p}\right]^3 & \text{if} \quad \left[\overline{p}/\left(\overline{p}+1\right)\right]k\overline{p} \leq s_j < k\overline{p} \\ -\left(1/p_j\right) & \text{otherwise} \end{cases}$$

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

The EQTP_LIN and EQTP_EXP dispatching rules assign a priority value of $-\left(1/p_j\right)$ to jobs that are in no danger of becoming tardy ($s_j \geq k\overline{p}$). This assures that two jobs that have large slacks will be scheduled in LPT order. Conversely, the SPT_s_j rule is used to calculate the priority value when a job is on time or late ($s_j \leq 0$). The EQTP_LIN and EQTP_EXP heuristics differ in the calculation of the job priorities for the intermediate values of the job slack. The priority decreases linearly as the job slack increases in the EQTP_LIN dispatching rule, while exponential and cubic functions are instead used in the EQTP_EXP heuristic.

The effectiveness of the EQTP_LIN and EQTP_EXP heuristics depends on the value of the lookahead parameter k. This parameter should reflect the number of competing critical jobs, that is, the number of jobs that may clash each time a sequencing decision is to be made (for details, see Ow and Morton, 1989). In the proposed implementation, the value of k is calculated dynamically at each iteration. Therefore, each time a scheduling decision has to be made, the characteristics of the current workload are used to determine an appropriate value for the lookahead parameter.

The following procedure is used to compute the value of the lookahead parameter k at each iteration. First, a critical slack value crit_slack is calculated, just as previously described for the CS heuristics. Then, each job is classified as critical if $0 < s_j \leq$ crit_slack, and non-critical otherwise. Therefore, a job is considered critical if it is not already tardy ($s_j > 0$), but is about to become tardy ($s_j \leq$ crit_slack). Finally, the lookahead parameter k is set equal to the number of critical jobs. The time complexity of the EQTP_LIN and EQTP_EXP dispatching rules is $O\left(n^2\right)$.

Again, consider the previous numerical example, and assume slack_prop $= 0.25$. In the first iteration, at time $t = 0$, the critical value crit_slack is equal to 9.5. Three jobs have a slack $0 < s_j \leq 9.5$, and the lookahead parameter is then set at $k = 3$. In the EQTP_LIN version, the priorities of the six available jobs are 0.4539, 0.6333, 0.8626, 2.5833, 0.9532 and 0.3534. Job 4 is selected for processing, since it has the largest priority. Once the remaining iterations are performed, the final sequence 4-5-3-2-1-6 (with an objective function value of 891) is obtained. The EQTP_EXP version, on the other hand, generates the sequence 4-2-5-3-1-6, with objective function value 938.

2.5 Greedy heuristic

In this section, a greedy-type procedure, denoted by Greedy, is presented. This heuristic is an adaptation of a procedure originally introduced by Fadlalla et al. (1994) for the mean tardiness problem, and later adapted to other problems (see, for instance, Valente and Alves, 2005; Volgenant and Teerhuis, 1999).

Two different versions of the Greedy heuristic are considered. These versions share the basic framework, and differ only slightly in the calculation of the job priorities. Let c_{xy}, with $x \neq y$, be the combined cost of scheduling jobs J_x and J_y, in this order, in the next two positions in the sequence, that is, 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. Also, let L be a list with the indexes of the yet unscheduled jobs and $P(j)$ the priority of job J_j. The steps of the Greedy_v1 version are:

Step 1. Initialisation:

Set $t = 0$ and $L = \{1, 2, \ldots, n\}$.

Step 2. Calculate the job priorities:

Set $P(j) = 0$, for all $j \in L$;

For all pairs of jobs $(i, j) \in L$, with $i < j$, do:

Calculate c_{ij} and c_{ji};

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 3. Select the next job:

Schedule job l for which $P(l) = \max\left\{P_j;\ j \in L\right\}$;

Set $t = t + p_l$ and $L = L \setminus \{l\}$.

Step 4. Stopping condition:

If $|L| = 1$, stop;

Else, go to step 2.

In the Greedy_v2 version, Step 2 is instead given by:

Step 2. Calculate the job priorities:

Set $P(j) = 0$, for all $j \in L$;

For all pairs of jobs $(i, j) \in L$, with $i < j$, do:

Calculate c_{ij}, c_{ji} and $|c_{ij} - c_{ji}|$;

If $c_{ij} < c_{ji}$

set $P(i) = P(i) + |c_{ij} - c_{ji}|$;

set $P(j) = P(j) - |c_{ij} - c_{ji}|$.

Else

set $P(i) = P(i) - |c_{ij} - c_{ji}|$;

set $P(j) = P(j) + |c_{ij} - c_{ji}|$.

If $c_{ij} < c_{ji}$, it seems better to schedule job J_i in the next position rather than job J_j. Conversely, it seems preferable to schedule job J_j next when $c_{ij} > c_{ji}$. In the Greedy_v1 version, 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 the other unscheduled jobs. In the

Greedy_v2 version, for all pairs of jobs (i, j), with $i < j$, the priority of the preferred job is instead increased by $|c_{ij} - c_{ji}|$, while the priority of the other job is decreased by that same value. The time complexity of both versions of the Greedy heuristic is $O(n^3)$.

Again, consider the numerical example. In the first iteration, the priorities of the six unscheduled jobs are equal to 2, 3, 4, 5, 0 and 1, in the Greedy_v1 version. In the Greedy_v2 version, these priorities are instead equal to −184, 4, −2, 392, −54 and −156. In both versions, job 4 has the largest priority, and is selected for processing. After the subsequent iterations are performed, the final sequence 4-3-5-2-1-6 (with objective function value 872) is then obtained, for both versions.

3 Computational results

In this section, the set of test problems used in the computational tests is first presented, and the preliminary computational experiments are described. These experiments are performed to determine appropriate values for the parameters required by the CS and EQTP heuristics. Moreover, the performance of the alternative versions of the CS, EQTP and Greedy heuristics is also analysed in these initial experiments in order to select the best-performing. Finally, the computational results are presented. The heuristic procedures are first compared, and the heuristic results are evaluated against optimum objective function values for some instance sizes.

The instances used in the computational tests are available online at http://www.fep.up.pt/docentes/jvalente/benchmarks.html. The objective function value provided by the EQTP_EXP heuristic, as well as the optimum objective function value (when available), can also be obtained at this address. Throughout this section, and in order to avoid excessively large tables, results will sometimes be presented only for some representative cases.

3.1 Experimental design

The computational tests are performed on a set of problems with 10, 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 was generated from one of the two uniform distributions [45, 55] and [1, 100], in order to obtain low (L) and high (H) variability, respectively, for the processing time values. For each job J_j, an integer due date d_j was 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 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 processing time 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 Greedy heuristic was only applied to instances with up to 500 jobs.

3.2 Parameter adjustment tests

In this section, the preliminary computational experiments are described. These initial experiments were performed to determine appropriate values for the parameters required

by the CS_AS, CS_LP, EQTP_LIN and EQTP_EXP dispatching rules. The performance of the alternative versions of the CS, EQTP and Greedy 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, 1000 and 2000 jobs, and contained five instances for each combination of instance size, processing time 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 to determine an appropriate value for the slack_prop parameter used by the CS_AS, CS_LP, EQTP_LIN and EQTP_EXP heuristics. The values $\{0.00, 0.05, 0.10, \ldots, 0.95\}$ were considered, and the objective function value was computed for each slack_prop value and each instance. An analysis of these results showed that a value of slack_prop$= 0.15$ provided the best performance for the CS_AS and CS_LP heuristics. For the EQTP_LIN and EQTP_EXP dispatching rules, the best results were given by slack_prop values in the range $[0.55, 0.95]$. The slack_prop parameter was then set at 0.60 for both the EQTP_LIN and the EQTP_EXP heuristics, since this value consistently provided good results for all instance types.

The slack_prop parameter is instance-dependent, so the best results can be achieved with different values when several instances are considered. Consequently, the values recommended above for this parameter will not provide the best possible performance for all instances. Nevertheless, the computational tests that were performed showed that the chosen values do consistently provide good results across all the instance types.

The slack_prop parameter is also problem- and shop-dependent. Therefore, other parameter values may be more appropriate for problems or shops whose characteristics are different from those of the considered test instances. For instance, production environments that use due date setting methods such as CON, SLACK or TWK may require different slack_prop values. In such environments, experiments should be performed to determine adequate values for the slack_prop parameter. These experiments are likely to be costly and/or difficult to perform, although this task is simplified by the fact that only one parameter has to be fine-tuned.

The performance of the alternative versions of the CS, EQTP and Greedy heuristics was also analysed in these preliminary computational experiments, in order to select the best-performing versions. Therefore, the following three ($h1$ versus $h2$) pairs of alternative heuristic versions were compared: (CS_AS versus CS_LP), (EQTP_EXP versus EQTP_LIN) and (Greedy_v1 versus Greedy_v2).

Table 1 presents the average relative improvement 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 $(h2\_\text{ofv} - h1\_\text{ofv}) / h2\_\text{ofv} \times 100$, where $h2$_ofv and $h1$_ofv are the objective function values of the appropriate heuristic versions.

The performance of the alternative versions of the CS heuristic is quite similar. In fact, the objective function values provided by these alternative versions is generally quite close, particularly for the medium and large size instances. The CS_AS version, however, usually provides better results than its CS_LP counterpart for a slightly larger number of instances.

The EQTP_EXP heuristic performs better than its EQTP_LIN alternative, particularly for instances with a high processing time variability. Indeed, the EQTP_EXP version provides on average a relative improvement in the objective function value of over 3% for instances with a high variability. For low variability instances, however, this improvement is under 1%. Also, the EQTP_EXP version gives better results for a larger number of the test instances.

Table 1 Heuristic version comparison

	n	*Low var*				*High var*			
		%imp	<	=	>	*%imp*	<	=	>
CS_AS	25	0.97	8.33	91.67	0.00	0.33	11.67	81.67	6.67
versus	50	0.12	13.33	82.50	4.17	0.02	16.67	67.50	15.83
CS_LP	100	0.03	9.17	83.33	7.50	−0.03	25.00	62.50	12.50
	250	0.07	6.67	76.67	16.67	0.13	15.83	62.50	21.67
	500	0.00	13.33	70.83	15.83	0.03	28.33	54.17	17.50
	1000	0.00	15.83	73.33	10.83	0.03	21.67	59.17	19.17
	2000	0.00	15.00	67.50	17.50	0.00	25.00	51.67	23.33
EQTP_EXP	25	0.41	43.33	55.83	0.83	3.81	65.83	25.00	9.17
versus	50	0.15	45.00	53.33	1.67	3.86	65.00	25.00	10.00
EQTP_LIN	100	0.10	48.33	49.17	2.50	3.73	65.00	23.33	11.67
	250	0.13	43.33	48.33	8.33	3.51	52.50	23.33	24.17
	500	0.08	40.83	45.83	13.33	3.37	49.17	25.00	25.83
	1000	0.07	42.50	45.83	11.67	3.03	49.17	25.00	25.83
	2000	0.07	40.83	45.00	14.17	2.81	49.17	25.00	25.83
Greedy_v1	25	0.05	54.17	45.83	0.00	−0.50	80.00	9.17	10.83
versus	50	0.24	64.17	34.17	1.67	4.19	85.00	0.83	14.17
Greedy_v2	100	0.01	78.33	20.83	0.83	5.70	83.33	0.00	16.67
	250	0.25	83.33	15.83	0.83	4.51	82.50	0.00	17.50
	500	0.10	87.50	12.50	0.00	5.51	82.50	0.00	17.50

The Greedy_v1 version clearly outperforms its Greedy_v2 alternative when the processing time variability is high. In fact, the Greedy_v1 heuristic provides a relative improvement in the objective function value of about 4–5% (with the exception of the instances with 25 jobs). For instances with low variability, the relative improvement given by the Greedy_v1 heuristic is below 1%. Also, for both low and high variability settings, the Greedy_v1 version gives better results for around 80% of the test instances. In the following sections, results will only be presented for the CS_AS, EQTP_EXP and Greedy_v1 versions.

3.3 Heuristic results

In this section, the computational results for the heuristic procedures are presented. Table 2 gives 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 EQTP_EXP heuristic, and are therefore presented as index numbers. More precisely, these values are calculated as heur_ofv/eqtp_exp_ofv × 100, where heur_ofv and eqtp_exp_ofv are the average objective function values of the appropriate heuristic and the EQTP_EXP dispatching rule, respectively.

The best results are given by the EQTP_EXP dispatching rule, closely followed by the CS_AS procedure. In fact, the EQTP_EXP heuristic not only provides the lowest average objective function value, but also gives the best results for a large percentage of the instances

(particularly for the largest instances, or when the variability of the processing times is low). The CS_AS procedure also performs quite well, providing an average objective function value that is quite close to the results given by the EQTP_EXP heuristic.

Table 2 Heuristic results

Var	*Heur*	*n = 25*		*n = 100*		*n = 500*		*n = 2000*	
		ofv	*%best*	*ofv*	*%best*	*ofv*	*%best*	*ofv*	*%best*
L	LPT	156.92	4.50	162.79	1.33	163.93	0.83	164.47	0.00
	EDD	100.81	8.67	100.93	3.25	100.95	2.75	100.95	3.67
	SPT	140.70	0.00	144.86	0.00	145.82	0.00	145.92	0.00
	SPT_sj	102.05	15.92	101.52	15.50	101.31	29.08	101.25	31.83
	CS_AS	100.10	32.67	100.05	23.42	100.05	27.83	100.04	24.58
	EQTP_EXP	100.00	65.83	100.00	71.25	100.00	86.58	100.00	91.75
	Greedy_v1	101.57	49.25	101.89	38.92	101.96	36.25	–	–
H	LPT	325.02	0.08	364.43	0.00	377.41	0.00	380.17	0.00
	EDD	134.93	4.83	139.77	0.75	141.89	1.67	142.08	1.83
	SPT	129.45	0.00	133.08	0.00	134.78	0.00	134.82	0.00
	SPT_sj	102.36	5.17	101.39	0.92	101.08	12.75	100.99	25.00
	CS_AS	100.13	20.92	100.25	10.42	100.29	18.08	100.29	23.50
	EQTP_EXP	100.00	43.75	100.00	55.25	100.00	53.92	100.00	91.83
	Greedy_v1	102.44	45.33	103.63	35.67	103.92	49.92	–	–

The SPT_s_j and the Greedy_v1 heuristics also provide good results. The Greedy_v1 procedure gives an average objective function value that is about 2–3% worse than the EQTP_EXP heuristic, but it nevertheless provides the best results for a significant number of instances. The SPT_s_j procedure performs well for medium and large instances, since it provides an average objective function value that is about 1% worse than the results given by the EQTP_EXP dispatching rule.

The simple LPT and SPT rules perform rather poorly, giving results that are substantially worse than those of the other heuristics. The linear SPT rule was clearly outperformed by its SPT_s_j quadratic counterpart, meaning that the modifications that were introduced in this linear rule, in order to adapt it to a quadratic objective function, have indeed significantly improved its performance. Therefore, it is certainly important to address the quadratic tardiness component of the cost function and develop a specific procedure, instead of simply using a heuristic appropriate for a linear tardiness objective function.

The EDD rule does provide an average objective function value that is quite close to the EQTP_EXP results for instances with a low processing time variability, but its performance is quite poor when the variability is high. The CS_AS heuristic performs considerably better than either of the EDD and SPT_s_j rules, particularly when the variability of the processing times is high. Consequently, a considerable performance improvement can be achieved by selectively using these two simple heuristics (i.e. by choosing at each iteration the rule that is expected to perform better, given the characteristics of the current job load).

Table 3 presents the effect of the T and R parameters on the average objective function value (calculated relative to the EQTP_EXP heuristic). This Table gives results for the best heuristics (the EQTP_EXP is omitted, since its values would all be equal

to 100) and for instances with 100 jobs. The SPT_s_j heuristic provides an average objective function value that is quite close to the results given by the EQTP_EXP dispatching rule for instances with a large tardiness factor T. The SPT_s_j rule, however, performs considerably worse when the tardiness factor is low. This result is to be expected, since the SPT_s_j heuristic is particularly well suited to problems where most jobs will be completed after their due dates, since it is derived from a local optimality condition for tardy jobs. When the tardiness factor T is high, most jobs will be tardy and the SPT_s_j rule indeed performs well. For low values of T, on the other hand, the proportion of tardy jobs is lower, and the performance of the SPT_s_j heuristic correspondingly deteriorates.

Table 3 Objective function value, relative to the EQTP_EXP heuristic, 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*
SPT_s_j	0.0	108.50	111.19	114.96	116.03	204.42	202.17	215.09	207.42
	0.2	160.11	376.49	315.94	184.66	131.46	794.73	565.84	467.52
	0.4	123.33	169.45	277.28	1325.44	119.35	156.01	280.71	1876.76
	0.6	108.34	115.34	115.49	101.34	104.68	112.71	115.49	109.19
	0.8	101.59	100.00	100.00	100.00	100.53	100.49	100.05	100.02
	1.0	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
CS_AS	0.0	102.82	102.78	102.73	102.68	135.53	133.37	131.02	126.46
	0.2	79.42	100.73	104.17	103.68	62.01	126.35	135.65	130.47
	0.4	100.57	100.04	100.01	100.13	99.97	100.41	102.19	109.45
	0.6	100.47	100.00	100.01	100.07	100.15	100.19	101.06	106.77
	0.8	100.42	100.00	100.01	100.02	100.15	100.11	100.60	101.93
	1.0	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Greedy_v1	0.0	100.35	100.61	100.85	101.82	100.35	100.98	101.82	103.93
	0.2	179.73	331.87	160.98	125.95	242.03	487.37	254.17	169.94
	0.4	138.45	175.98	236.52	986.42	183.55	217.47	337.38	1351.64
	0.6	114.44	117.69	114.65	100.50	132.66	132.32	125.99	101.88
	0.8	102.45	100.00	100.00	100.00	105.71	99.98	99.99	99.99
	1.0	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

The CS_AS heuristic is quite close to the EQTP_EXP dispatching rule for $T \geq 0.4$. However, the CS_AS heuristic is clearly outperformed when most jobs are early, particularly when the variability of the processing times is high. In fact, when the processing time variability is high (low), the CS_AS heuristic provides an average objective function value that is about 30% (3%) worse than the results given by the EQTP_EXP procedure when $T = 0.0$ or $T = 0.2$ and $R \geq 0.4$. The Greedy_v1 heuristic performs well for instances where most jobs are early ($T = 0.0$) or tardy ($T = 1.0$). The performance of the Greedy_v1 procedure deteriorates when the tardiness factor T takes on intermediate values. Therefore,

the Greedy_v1 procedure is less effective when there is a greater balance between the number of early and tardy jobs.

The heuristic runtimes (in sec) are presented in Table 4. The Greedy_v1 heuristic is computationally demanding, and therefore can only be used for small and medium size instances. The other heuristic procedures are quite fast, even for large instances. The simple LPT, EDD and SPT rules are the most efficient, since they only require sorting, which can be performed in $O(n \log n)$ time. The SPT_s_j and CS_AS procedures are also quite efficient, even with their higher $O(n^2)$ time complexity. The EQTP_EXP dispatching rule, even though it also requires $O(n^2)$ time, is more computationally demanding. Nevertheless, this heuristic is still extremely fast, being capable of solving even quite large instances with 2000 jobs in less than 0.3 sec on a personal computer. The EQTP_EXP is then the heuristic procedure of choice, since it not only provides the best results, but is also computationally efficient.

Table 4 Heuristic runtimes (in sec)

Var	*Heur*	*n = 100*	*n = 250*	*n = 500*	*n = 1000*	*n = 1500*	*n = 2000*
L	LPT	0.0000	0.0001	0.0002	0.0004	0.0006	0.0006
	EDD	0.0000	0.0000	0.0001	0.0002	0.0004	0.0004
	SPT	0.0000	0.0001	0.0002	0.0003	0.0004	0.0006
	SPT_sj	0.0001	0.0009	0.0030	0.0123	0.0267	0.0477
	CS_AS	0.0002	0.0009	0.0036	0.0139	0.0312	0.0544
	EQTP_EXP	0.0005	0.0041	0.0153	0.0600	0.1360	0.2412
	Greedy_v1	0.1887	2.9302	23.3965	–	–	–
H	LPT	0.0000	0.0001	0.0001	0.0004	0.0006	0.0008
	EDD	0.0000	0.0001	0.0001	0.0002	0.0003	0.0006
	SPT	0.0001	0.0001	0.0001	0.0003	0.0004	0.0007
	SPT_sj	0.0001	0.0007	0.0030	0.0123	0.0264	0.0464
	CS_AS	0.0001	0.0010	0.0037	0.0144	0.0326	0.0593
	EQTP_EXP	0.0007	0.0042	0.0160	0.0639	0.1433	0.2548
	Greedy_v1	0.1858	2.8757	22.9192	–	–	–

3.4 Comparison with optimum results

In this section, the heuristic results are compared with the optimum objective function values for instances with up to 20 jobs. The optimum results were obtained using the branch-and-bound algorithm proposed by Valente (to appear). Table 5 presents the average of the relative deviations from the optimum (%dev), calculated as $(H - O)/O \times 100$, 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 5, it can be seen that the heuristics are much closer to the optimum for instances with a low processing time variability, with the exception of the SPT procedure. The EQTP_EXP and CS_AS heuristics perform quite well for instances with a low variability, giving results that are 1–2% above the optimum. The other heuristics are far from the optimum, with the exception of the EDD rule. When the variability of the processing times is high, however, even the best-performing EQTP_EXP and CS_AS

heuristics are 10–20% above the optimum (though the deviation from the optimum decreases with the instance size for the EQTP_EXP procedure). For the Greedy_v1 heuristic, the average deviation from the optimum is quite large for instances with a high processing time range. However, this heuristic provides an optimum solution for a large number of instances. In fact, for some problem sizes, particularly when the variability is low, the Greedy_v1 procedure generates an optimum solution for over 50% of the instances.

Table 5 Comparison with optimum objective function values

Var	*Heur*	*n = 10*		*n = 15*		*n = 20*	
		%dev	*%opt*	*%dev*	*%opt*	*%dev*	*%opt*
L	LPT	641.48	7.08	1047.29	6.00	1404.28	4.33
	EDD	1.50	9.17	1.71	2.00	1.86	0.92
	SPT	586.77	0.00	835.35	0.00	1088.79	0.00
	SPT_sj	128.88	23.67	119.50	21.17	98.89	18.17
	CS_AS	1.61	30.33	1.53	24.25	1.66	20.58
	EQTP_EXP	1.78	45.58	2.14	34.50	1.83	28.17
	Greedy_v1	38.79	62.17	55.41	54.67	54.59	48.17
H	LPT	1659.80	2.17	2786.84	0.50	3755.46	0.50
	EDD	32.07	0.33	36.33	0.00	37.32	0.00
	SPT	589.29	0.00	810.08	0.00	1051.87	0.00
	SPT_sj	195.66	7.17	230.05	5.25	224.85	3.50
	CS_AS	13.48	8.33	14.47	5.75	14.84	3.42
	EQTP_EXP	22.14	22.25	16.45	11.92	11.96	8.67
	Greedy_v1	40.18	52.67	68.90	35.83	92.85	30.33

The effect of the T and R parameters on the relative deviation from the optimum is presented in Table 6. This table gives results for the best heuristics and for instances with 20 jobs. The SPT_s_j heuristic is quite close to the optimum for instances with a large tardiness factor T. The average relative deviation from the optimum, however, is substantially higher when the tardiness factor is low. This is to be expected, since the SPT_s_j heuristic is particularly suited to problems where most jobs will be tardy. When the tardiness factor T is high, most jobs will indeed be tardy and the SPT_s_j rule is then quite close to optimal. For low values of T, on the other hand, the number of tardy jobs is lower, and the average deviation of the SPT_s_j heuristic from the optimum correspondingly increases.

The CS_AS dispatching rule is quite close to the optimum when the tardiness factor is greater than or equal to 0.6. The performance of the CS_AS heuristic, however, is clearly inferior when most jobs are early, particularly when the variability is high. In fact, the CS_AS heuristic is about 30–50% above the optimum for high variability instances with $T = 0.0$ or $T = 0.2$. The effect of the T and R parameters on the average relative deviation from the optimum is similar for the EQTP_EXP and the Greedy_v1 heuristics. These procedures are much closer to the optimum when nearly all jobs are early ($T = 0.0$) or when there is a larger proportion of tardy jobs ($T \geq 0.6$). The performance of these dispatching rules deteriorates, particularly for the Greedy_v1 heuristic, when $T = 0.2$ or $T = 0.4$.

Table 6 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*
SPT_s$_j$	0.0	15.57	24.92	22.35	70.52	131.89	141.19	191.15	194.88
	0.2	103.34	312.48	261.33	382.97	125.65	620.08	1546.85	729.72
	0.4	22.38	62.02	168.47	873.41	22.86	64.63	185.59	1377.21
	0.6	8.82	16.02	17.08	9.92	5.60	12.68	17.66	22.90
	0.8	1.50	0.14	0.00	0.00	1.45	1.81	1.10	1.08
	1.0	0.00	0.00	0.00	0.00	0.05	0.07	0.09	0.11
CS_AS	0.0	2.90	2.98	3.00	2.57	37.15	33.52	36.63	31.75
	0.2	3.59	4.47	5.13	4.67	13.20	54.29	53.44	46.79
	0.4	1.35	0.66	1.30	4.43	2.73	3.51	8.13	19.82
	0.6	1.11	0.33	0.01	0.10	1.94	1.45	1.75	5.18
	0.8	1.10	0.08	0.00	0.01	1.22	1.31	0.91	1.20
	1.0	0.00	0.00	0.00	0.00	0.05	0.07	0.09	0.11
EQTP_EXP	0.0	0.19	0.09	0.08	0.11	0.66	1.64	2.65	2.64
	0.2	17.05	13.56	3.98	2.23	60.07	91.36	26.80	20.54
	0.4	0.10	0.13	0.42	5.92	6.34	4.57	13.49	44.53
	0.6	0.02	0.02	0.01	0.01	3.97	1.38	1.23	1.88
	0.8	0.01	0.01	0.00	0.00	1.68	0.82	0.30	0.25
	1.0	0.00	0.00	0.00	0.00	0.03	0.05	0.05	0.06
Greedy_v1	0.0	0.75	1.41	2.90	5.44	3.33	8.85	16.18	19.08
	0.2	74.84	173.11	170.09	198.99	127.37	265.48	287.64	371.61
	0.4	28.41	61.53	108.97	453.53	60.78	70.64	139.77	814.20
	0.6	9.88	10.13	8.51	0.91	18.66	17.10	5.51	1.05
	0.8	0.68	0.00	0.00	0.00	0.96	0.13	0.06	0.04
	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.01

4 Conclusion

This paper considered the single machine scheduling problem with linear earliness and quadratic tardiness costs, and no machine idle time. Several dispatching heuristics were proposed, and their performance was analysed on a wide range of instances. The heuristics included simple scheduling rules, as well as a procedure that takes advantage of the strengths of these rules. Linear early/quadratic tardy dispatching rules were also considered, as well as a greedy-type procedure. Extensive computational experiments were performed to determine adequate values for the parameters required by some of the heuristics.

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. Also, dispatching rules are often the only heuristic approach capable of generating solutions, within reasonable computation times, for large instances. Additionally, dispatching rules

are used by other heuristic procedures, for example, they are often used to generate the initial sequence required by local search or metaheuristic algorithms.

The best results were given by the EQTP_EXP dispatching rule. This heuristic provided the lowest average objective function values, and also obtained the best results for a large percentage of the instances. The performance of the EQTP_EXP procedure was quite good for instances with a low processing time range, since it provided results that are about 1–2% above the optimum. For instances with a high range, however, the deviation from the optimum exceeded 10%, though it decreased as the instance size increased.

The Greedy_v1 heuristic is computationally demanding, and therefore can only be used for small and medium size instances. The other heuristic procedures, however, were quite fast, and are capable of solving even very large instances in less than one second on a personal computer. The EQTP_EXP dispatching rule is then the heuristic procedure of choice, since it not only provided the best results, but is also computationally efficient.

The best of the proposed heuristics perform quite adequately for the problem with no idle time. These heuristics, however, might also be useful for the problem with inserted idle time. Indeed, the procedure developed by Schaller (2004) can be applied to optimally insert idle time in the sequences generated by the proposed heuristics. Therefore, investigating the performance of these heuristics for the problem with inserted idle time certainly seems an interesting possibility for future research.

Acknowledgements

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

References

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

Fadlalla, A., Evans, J.R. and Levy, M.S. (1994) 'A greedy heuristic for the mean tardiness sequencing problem', *Computers and Operations Research*, Vol. 21, pp.329–336.

Garey, M.R., Tarjan, R.E. and Wilfong, G.T. (1988) 'One-processor scheduling with symmetric earliness and tardiness penalties', *Mathematics of Operations Research*, Vol. 13, pp.330–348.

Gupta, S.K. and Sen, T. (1983) 'Minimizing a quadratic function of job lateness on a single machine', *Engineering Costs and Production Economics*, Vol. 7, pp.187–194.

Hoogeveen, H. (2005) 'Multicriteria scheduling', *European Journal of Operational Research*, Vol. 167, pp.592–623.

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

Kim, Y.D. and Yano, C.A. (1994) 'Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates', *Naval Research Logistics*, Vol. 41, pp.913–933.

Korman, K. (1994) 'A pressing matter', *Video*, pp.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.

Ow, P.S. and Morton, T.E. (1989) 'The single machine early/tardy problem', *Management Science*, Vol. 35, pp.177–191.

Schaller, J. (2002) 'Minimizing the sum of squares lateness on a single machine', *European Journal of Operational Research*, Vol. 143, pp.64–79.

Schaller, J. (2004) 'Single machine scheduling with early and quadratic tardy penalties', *Computers and Industrial Engineering*, Vol. 46, pp.511–532.

Schaller, J. (2007) 'A comparison of lower bounds for the single-machine early/tardy problem', *Computers and Operations Research*, Vol. 34, pp.2279–2292.

Sen, T., Dileepan, P. and Lind, M.R. (1995) 'Minimizing a weighted quadratic function of job lateness in the single machine system', *International Journal of Production Economics*, Vol. 42, pp.237–243.

Su, L-H. and Chang, P-C. (1998) 'A heuristic to minimize a quadratic function of job lateness on a single machine', *International Journal of Production Economics*, Vol. 55, pp.169–175.

Sun, X., Noble, J.S. and Klein, C.M. (1999) 'Single-machine scheduling with sequence dependent setup to minimize total weighted squared tardiness', *IIE Transactions*, Vol. 31, pp.113–124.

Taguchi, G. (1986) *Introduction to Quality Engineering*, Asian Productivity Organization, Tokyo, Japan.

Valente, J.M.S. (to appear) 'An exact approach for the single machine scheduling problem with linear early and quadratic tardy penalties', *Asia-Pacific Journal of Operational Research*.

Valente, J.M.S. and Alves, R.A.F.S. (2005) 'Improved heuristics for the early/tardy scheduling problem with no idle time', *Computers and Operations Research*, Vol. 32, pp.557–569.

Valente, J.M.S. and Alves, R.A.F.S. (to appear) 'Heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties', *Computers and Operations Research*.

Volgenant, A. and Teerhuis, E. (1999) 'Improved heuristics for the n-job single-machine weighted tardiness problem', *Computers and Operations Research*, Vol. 26, pp.35–44.

Wagner, B.J., Davis, D.J. and Kher, H. (2002) 'The production of several items in a single facility with linearly changing demand rates', *Decision Sciences*, Vol. 33, pp.317–346.