Computers & Operations Research 39 (2012) 2223–2231

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Dispatching heuristics for the single machine weighted quadratic tardiness scheduling problem ☆

Jorge M.S. Valente [a,*], Jeffrey E. Schaller [b]

[a] LIAAD—INESC Porto LA, Faculdade de Economia da Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4200-464 Porto, Portugal
[b] Department of Business Administration, Eastern Connecticut State University, 83 Windham St., Willimantic, CT 06226-2295, USA

ARTICLE INFO

Available online 17 November 2011

Keywords:
Scheduling
Single machine
Weighted quadratic tardiness
Heuristics
Dispatching rules

ABSTRACT

In this paper, we consider the single machine scheduling problem with weighted quadratic tardiness costs. Several efficient dispatching rules are proposed. These include existing heuristics for the linear problem, as well as procedures suitably adapted to the quadratic objective function. Also, both forward and backward scheduling procedures are considered.

The computational results show that the heuristics that specifically take into account the quadratic objective significantly outperform their linear counterparts. Also, the backward scheduling approach proves to be superior, and the difference in performance is even more noticeable for the harder instances.

The best of the backward scheduling heuristics is both quite efficient and effective. Indeed, this procedure can quickly generate a schedule even for large instances. Also, its relative deviation from the optimum is usually rather low, and it performs adequately even for the more difficult instances.

1. Introduction

This paper considers the single machine scheduling problem with weighted quadratic tardiness costs. Formally, the problem can be stated as follows. A set of n independent jobs $\{1, 2, \ldots, n\}$ has to be scheduled on a single machine that can handle only one job at a time. The machine is assumed to be continuously available from time zero onwards, and preemptions are not allowed. Job j, $j=1, 2, \ldots, n$, requires a processing time p_j, has a weight w_j and should ideally be completed on its due date d_j. For a given schedule, the tardiness of job j is defined as $T_j=\max\{0, C_j-d_j\}$, where C_j is the completion time of job j. The objective is then to find a schedule that minimizes the sum of the weighted squared tardiness values $\sum_{j=1}^{n} w_j T_j^2$.

Single machine scheduling environments may appear to arise infrequently in practice, but they actually occur in several practical settings; a specific example in the chemical industry is given in [1]. Scheduling models with a single machine are also useful for problems with multiple processors. Indeed, the performance of many production systems is frequently determined by the quality of the schedules for a single bottleneck machine. Furthermore, results and insights obtained for single machine problems can often be applied to more complex scheduling environments, such as parallel machines, flow shops or job shops.

The objective function considers squared tardiness costs. Tardiness is a widely used performance measure in scheduling problems, since tardy deliveries can result in contractual penalties, lost sales and loss of customer goodwill. Squared tardiness is used in this paper, since a customer's dissatisfaction tends to increase quadratically with the tardiness, as proposed in the loss function of Taguchi [2].

A weighted quadratic tardiness objective, to the best of our knowledge, has only been considered in a limited number of papers. Hoitomt et al. [3] developed a Lagrangian relaxation based solution procedure for scheduling jobs with simple precedence constraints on parallel machines, and demonstrated it with three examples. Several heuristics were presented by Sun et al. [4] for the single machine problem with release dates and sequence dependent setup times. Thomalla [5] considered the job shop scheduling problem with alternative processing plans, and compared a lagrangean relaxation based lower bound and heuristics with other methods for three small examples. Recently, Schaller and Valente [6] proposed several dominance rules, as well as branch-and-bound algorithms, which incorporate these rules, for the problem considered in this paper.

The complexity of the single machine weighted quadratic tardiness problem is, again to the best of our knowledge, still open. However, and given existing complexity results, it seems

☆This work has been partially financed by FCT (Fundação para a Ciência e Tecnologia), in the context of research project PTDC/EGE-GES/099741/2008.
* Corresponding author. Tel.: +351 225 571 100; fax: +351 225 505 050.
E-mail addresses: jvalente@fep.up.pt (J.M.S. Valente), schallerj@easternct.edu (J.E. Schaller).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.11.005

most likely that the problem is hard. Indeed, the corresponding total weighted tardiness problem (i.e. the corresponding linear problem) is strongly NP-hard [7,8].

Two single machine streams of research that are related to the considered problem are models with a quadratic performance measure and the total weighted tardiness problem. Among tardiness-related quadratic performance measures, the quadratic lateness problem has been studied by Gupta and Sen [9], Sen et al. [10], Su and Chang [11], Schaller [12] and Soroush [13,14]. Also, the linear earliness and squared tardiness problem was considered by Schaller [15], Valente [16–18] and Valente and Schaller [19]. A large number of papers have been published on the total weighted tardiness problem. Exact methods have been surveyed and compared in [20], while several heuristic methods were analyzed in [21]. A more recent literature review of both exact and heuristic procedures is provided by Sen et al. [22].

This paper presents and analyses efficient dispatching rules. Dispatching procedures are widely used in practice; in fact, most real scheduling systems are either based on these heuristics, or at least use them to some degree. Moreover, dispatching rules are sometimes the only approach capable of generating a solution within reasonable computation times for large instances. Furthermore, dispatching heuristics are also frequently used by other procedures, e.g. they are often used to generate the initial sequence required by metaheuristics. Both forward and backward scheduling heuristics are considered, and the proposed procedures are compared among themselves, as well as with optimal solutions for small problem sizes.

As previously mentioned, both exact and heuristic methods have been proposed for the related linear earliness and squared tardiness problem. Though this problem may seem quite similar to the squared tardiness problem considered in this paper, there actually exist key differences.

On the one hand, both objective functions are relevant in practice. The inclusion of the linear earliness is adequate when an early job has to be held in inventory, thereby incurring a holding cost. However, for some organizations and/or items, inventory holding costs might be insignificant when compared with the tardiness costs, or may even be totally non-existent (e.g. when the products are computer tasks or processes, which when completed will simply be stored in memory or in a hard drive). In these cases, an objective function involving only a squared tardiness component is more appropriate.

On the other hand, removing the linear earliness component from the objective function actually has a large effect on the design of effective algorithms. The backward scheduling approach used in this paper was certainly influenced by the results obtained for the linear earliness and squared tardiness problem [19]. However, the priority indices of both forward and backward dispatching rules are quite different for the two problems. Indeed, the absence of earliness costs means that the heuristics can focus solely on the tardiness. This leads to a significant change in the choice and usage of components in the development of the priority indices. As such, the priority functions that perform well for the two problems are quite distinct.

The remainder of this paper is organized as follows. In Section 2, the heuristic procedures are described. The computational results are reported in Section 3. Finally, some concluding remarks are provided in Section 4.

2. The heuristic procedures

In this section, the considered dispatching rules are described. Dispatching heuristics build the schedule one job at a time. That is, at each iteration they select the next job to be added to the current partial sequence. This selection is typically done by calculating a priority index for each unscheduled job, and choosing the job with the largest priority index.

The proposed heuristics include both forward and backward scheduling procedures. In forward scheduling, the schedule is built from the beginning, i.e. the selected job at a given iteration is added to the end of the current partial sequence. Backward scheduling procedures, on the other hand, build the schedule from the end, so the chosen job is instead scheduled at the beginning of the current partial sequence.

2.1. Forward scheduling dispatching rules

The considered forward dispatching rules include the best performing procedures for the linear problem, as well as modified versions of these procedures, suitably adapted to the quadratic tardiness objective. The priority indices of the forward heuristics are given in Table 1. The previously undescribed notation is defined as follows. Let t^F denote the current time in the forward scheduling dispatching rules, i.e. the time at which the next job to be scheduled will start. The slack of job j when forward scheduling is used is defined as $s_j^F = d_j - t^F - p_j$. Finally, p is the average processing time of the currently unscheduled jobs and k is a parameter. Further details concerning k will be provided in the remainder of this section.

The earliest due date (EDD) rule is widely used for scheduling problems involving due dates. This heuristic schedules jobs in non-decreasing order of their due dates, which corresponds to using a priority index equal to $-d_j$. The weighted shortest processing time (WSPT) rule sorts the jobs in non-increasing order of w_j/p_j. This rule provides an optimal sequence for the linear problem if all jobs are necessarily tardy [23].

The weighted modified due date (WMDD) and the apparent tardiness cost (ATC) heuristics were developed for the linear objective in [24,25], respectively. A class of dispatching rules for both the weighted and unweighted linear tardiness problem was analyzed by Alidaee and Ramakrishnan [26]. The procedure that

Table 1
Forward dispatching rules.

Heuristic	Priority index
EDD	$-d_j$
WSPT	w_j/p_j
WMDD	$\begin{cases} w_j/p_j & \text{if } s_j^F \le 0 \\ w_j/(d_j - t^F) & \text{otherwise} \end{cases}$
ATC	$\begin{cases} w_j/p_j & \text{if } s_j^F \le 0 \\ (w_j/p_j)\exp(-s_j^F/kp) & \text{otherwise} \end{cases}$
AR	$\begin{cases} w_j/p_j & \text{if } s_j^F \le 0 \\ (w_j/p_j)[kp/(kp+s_j^F)] & \text{otherwise} \end{cases}$
WPT_SLK	$(w_j/p_j)[p+2\max(t^F+p_j-d_j;0)]$
QWMDD	$\begin{cases} (w_j/p_j)[p+2\max(t^F+p_j-d_j;0)] & \text{if } s_j^F \le 0 \\ w_j/(d_j-t^F)p & \text{otherwise} \end{cases}$
QATC	$\begin{cases} (w_j/p_j)[p+2\max(t^F+p_j-d_j;0)] & \text{if } s_j^F \le 0 \\ (w_j/p_j)p\exp(-s_j^F/kp) & \text{otherwise} \end{cases}$
QAR	$\begin{cases} (w_j/p_j)[p+2\max(t^F+p_j-d_j;0)] & \text{if } s_j^F \le 0 \\ (w_j/p_j)p[kp/(kp+s_j^F)] & \text{otherwise} \end{cases}$

provided the best results for the weighted objective is denoted here by AR.

Several computational studies have shown that the WMDD, ATC and AR heuristics provide the best performance among the efficient dispatching rules available for the linear problem; see, for instance, [25,21,27,26,28,24]. These heuristics differ only in the choice of priority index when a job is early.

Indeed, when a job is tardy, and in accordance with the previously mentioned result that the WSPT rule is optimal if all jobs will necessarily be completed after their due dates, the priority index is equal to the maximum value of w_j/p_j for all these three procedures. As the job's slack becomes increasingly larger than 0, i.e. as the job becomes increasingly earlier, the priority decreases. The WMDD, ATC and AR heuristics use somewhat different expressions to achieve this reduction in the priority value.

The parameter k provides the ATC and AR heuristics with a look ahead capability. In fact, and as described in [25], the parameter k is related with the number of competing critical jobs, i.e. jobs, which are close to becoming tardy. Therefore, the multiplication of the average processing time of the currently unscheduled jobs by the parameter k allows the priority index to take into account the number of jobs, which will become tardy in the next few iterations.

The WPT_SLK heuristic is an adaptation of the WSPT rule for a quadratic setting developed in [29]. When quadratic tardiness is considered, the due dates become relevant even when all jobs are necessarily tardy, i.e. the WSPT order does not guarantee an optimal schedule. The job's due date is therefore included in the WPT_SLK priority index. The WPT_SLK identifier is used since this heuristic's priority index includes both weighted processing time (WPT) and inverse slack (SLK) components. However, and unlike in the linear case, scheduling the jobs using the WPT_SLK procedure does not guarantee an optimal schedule for the quadratic problem, even if all jobs will always be completed after their due dates.

Finally, three new forward procedures are proposed, namely the QWMDD, QATC and QAR heuristics. These dispatching rules correspond to the modifications of the WMDD, ATC and AR procedures, suitably adapted to consider the quadratic tardiness objective. The QWMDD, QATC and QAR heuristics differ from their linear counterparts only in the expression of the priority index when a job is tardy. Indeed, the quadratic procedures replace the WSPT index with that of the WPT_SLK heuristic, in accordance with the comparison of these two procedures presented before. Again, the parameter k gives the QATC and QAR heuristics a look ahead capability, just as previously described for their linear counterparts.

2.2. Backward scheduling dispatching rules

The proposed backward scheduling rules will now be presented. The priority indices of the backward heuristics are given in Table 2. The previously undescribed notation is defined as follows. Let t^B denote the current time in the backward scheduling dispatching rules, i.e. the time at which the next job to be scheduled will be completed. The slack of job j when backward scheduling is used is defined as $s_j^B = t^B - d_j$. Also, let p_{max} denote the maximum processing time of the currently unscheduled jobs. Finally, p_j^{mod} is the (possibly) modified processing time of job j and v is a parameter. Further details regarding p_j^{mod} and v will be provided in the remainder of this section.

As detailed below, the Back dispatching rule is suited to the linear problem, while the QBack_v1 to QBack_v6 heuristics are tailored for the quadratic objective. The Back rule (which is also a new heuristic procedure) is included mainly for comparison purposes, in order to see if large improvements can be obtained using heuristics that specifically take the quadratic tardiness into consideration.

Table 2
Backward dispatching rules.

Heuristic	Priority index
Back	$\begin{cases} p_j & \text{if } s_j^B \le 0 \\ -(w_j/p_j^{mod}) & \text{otherwise} \end{cases}$
QBack_v1	$\begin{cases} p_j & \text{if } s_j^B \le 0 \\ -(w_j/p_j^{mod})[p+2s_j^B] & \text{otherwise} \end{cases}$
QBack_v2	$\begin{cases} p_j & \text{if } s_j^B \le 0 \\ -(w_j/p_j^{mod})(s_j^B)^2 & \text{otherwise} \end{cases}$
QBack_v3	$\begin{cases} p_j & \text{if } s_j^B \le 0 \\ -(w_j/p_j^{mod})s_j^B & \text{otherwise} \end{cases}$
QBack_v4	$\begin{cases} p_j & \text{if } s_j^B \le 0 \\ -(w_j/p_j^{mod})[(s_j^B)^2-(\max\{t^B-p-d_j,0\})^2] & \text{otherwise} \end{cases}$
QBack_v5	$\begin{cases} p_j & \text{if } s_j^B \le 0 \\ -(w_j/p_j^{mod})[(s_j^B)^2-(\max\{t^B-p_{max}-d_j,0\})^2] & \text{otherwise} \end{cases}$
QBack_v6	$\begin{cases} p_j & \text{if } s_j^B \le 0 \\ -(w_j/p_j^{mod})[(s_j^B)^2-v(\max\{t^B-p_{max}-d_j,0\})^2] & \text{otherwise} \end{cases}$

The backward dispatching rules schedule an early job whenever such a job is available. Indeed, this job will have an objective function value of zero, and will decrease the tardiness of all jobs that are currently tardy. This is achieved by setting the priority of early jobs to a positive value (more specifically, the priority is set equal to p_j), while the priority value of a tardy job is negative. Thus, the backward scheduling heuristics differ only in the choice of the priority index for jobs that are currently tardy. We remark that these comments are also applicable to the linear problem, for which the Back rule is suited.

When all unscheduled jobs are tardy, the backward scheduling rules select a job j using the weight and current tardiness, as well as the (possibly) modified processing time. Indeed, the weight and current tardiness of job j determine not only the current objective function value of that job, but also the potential for decreasing that objective function value if job j is scheduled at a later iteration. On the other hand, the processing time of job j is a measure of the reduction that can be achieved in the other jobs' tardiness if job j is chosen at the current iteration.

In the Back heuristic, the priority index only includes the weight and the processing time (possibly modified, as will shortly be detailed), in accordance with the comments that were made regarding the WSPT rule in the context of the forward procedures. That is, when all unscheduled jobs are tardy, the Back rule selects the job according to the WSPT rule, which, as previously mentioned, is optimal when all jobs are necessarily tardy [23]. Therefore, the Back dispatching procedure is suited to the linear problem.

The quadratic heuristics, and given the remarks previously made concerning the WPT_SLK heuristic, additionally include the due date in the priority index. The QBack_v1 rule actually incorporates the WPT_SLK index (again, with a possibly modified processing time), and the other quadratic backward heuristics use somewhat different expressions. Since these heuristics incorporate the WPT_SLK index, or variations of this index, they are suited to the quadratic problem.

In the backward rules, and when a job j is currently tardy, its processing time has been replaced in the priority index by p_j^{mod}. Three different versions were considered for each of the backwards heuristics, each corresponding to using a different strategy to set the value of p_j^{mod}. In the first version, p_j^{mod} is simply set equal to p_j, i.e. the original processing time of the job. The other two versions calculate, at each iteration, and for jobs whose tardiness is positive, the current value of those jobs' tardiness. Let T^{min} represent the minimum of these values and T_j^{min} be the minimum of those values for jobs other than j. In the second and third versions, p_j^{mod} is set equal to min $\{p_j, T^{min}\}$ and min $\{p_j, T_j^{min}\}$, respectively.

The second and third versions were developed due to the following. Higher processing times lead to higher priority values for tardy jobs. Indeed, a higher p_j decreases the tardiness of other jobs by a larger amount, and may actually lead to one of those jobs becoming early. However, under certain circumstances, a larger processing time may actually increase the priority by more than an adequate value. For instance, assume that both jobs j and k are currently tardy and $p_j > p_k$. Also assume that scheduling either j or k at the current iteration will lead to another job being early at the next iteration. In this case, scheduling either j or k will allow the selection of an early job at the next step. Therefore, the fact that $p_j > p_k$ does not actually translate into an actual advantage for scheduling j instead of k. The second and third versions correspondingly reduce the processing time of a job when it is larger than the time required for some job to become early.

The rationale for the priority index of heuristics QBack_v4 and QBack_v5 is somewhat similar. The priority expression for these two heuristics includes an additional term when compared with that of QBack_v2 (and also QBack_v1 and QBack_v3). This term attempts to capture the reduction in the squared tardiness of job j if another job is scheduled at the current iteration. Indeed, if there is a large reduction in the objective function value of j if some other job is selected for processing, then this favors not scheduling job j at the current iteration.

Finally, the QBack_v6 heuristic attempts to combine the priority indices of the QBack_v2 and QBack_v5 procedures. In fact, preliminary tests showed that QBack_v5 outperformed the other backward rules, with the exception of QBack_v2 for instances with certain specific characteristics. Though QBack_v5 was clearly superior to QBack_v2 for most instance types, QBack_v2 provided better results when several jobs were about to become early, i.e. when several jobs had small positive slacks. The QBack_v6 heuristic, through the use of the v parameter, allows for the priority index to change between those of the QBack_v2 and QBack_v5 procedures, according to the characteristics of the currently unscheduled jobs.

More specifically, at each iteration the value of v is determined as follows. Let s^B denote the average slack of the currently unscheduled jobs, i.e. the average of the s_j^B values. When $p \geq s^B$, v is set at 0. Indeed, in this situation the slacks are small. Setting v at 0 therefore assures that the QBack_v6 index becomes equal to that of the QBack_v2 rule, which performs well under these circumstances. When $p < s^B$ and $s^B/t^B > w$, where w is a user-defined value, the slacks are large. Consequently, v is set at 1, and the QBack_v6 priority index is then equal to that of the QBack_v5 rule. Finally, when $p < s^B$ and $s^B/t^B \leq w$, v is set equal to $(s^B - p)/s^B$. Indeed, in this situation the slacks are neither small nor large. Therefore, the priority index of the QBack_v6 rule is set somewhere in between those of the QBack_v2 and QBack_v5 heuristics, according to the relative size of the slack.

3. Computational results

In this section, the computational experiments and results are presented. First, the set of test problems used to obtain the computational results is described. Next, the preliminary tests that were performed in order to determine adequate values for the parameters required by some of the heuristics are presented. A comparison of the dispatching rules is then performed. Finally, the results of the best heuristics are compared with optimal solutions for small problem sizes.

3.1. 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, 1000, 1500 and 2000 jobs. The approach used to generate the problems was the same as the one that was used to create the linear weighted tardiness problem instances available in the OR-Library (http://people.brunel.ac.uk/~mastjjb/jeb/orlib/wtinfo.html).

More specifically, the problems were randomly generated as follows. For each job j, an integer processing time p_j was generated from a uniform distribution [1, 100], while an integer weight w_j was obtained from a uniform distribution [1, 10].

For each job 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 and R is the range of due dates. Both the tardiness factor and the range of due dates parameters were set at 0.2, 0.4, 0.6, 0.8 and 1.0.

For each combination of problem size n, T and R, 50 instances were randomly generated. Therefore, a total of 1250 instances were generated for each problem size. The procedures were coded in C++ and executed on a personal computer with a Core 2 Quad Q6600 2.40 GHz processor.

3.2. Parameter adjustment tests

Extensive preliminary tests were conducted to determine adequate values for the parameters k and w, as well as the best choice for p_j^{mod}. For each instance, the objective function value was calculated for each considered value of the appropriate parameter or each choice for p_j^{mod}. These results were then analyzed to select k, w and p_j^{mod}. The experiments were performed on a separate problem set, which included instances with 25, 50, 100, 250, 500, 1000 and 2000 jobs, and contained 5 instances for each combination of n, T and R.

Two approaches were considered to determine a value for k, required by the ATC, AR, QATC and QAR rules. The first is to set k at a constant value. We considered the values {0.5, 1.0, …, 4.5}, since k is usually set within this range for the linear problem; see, for instance, [25,30]]. The second approach allows for k to change at each iteration. As previously mentioned, k should increase with the number of critical competing jobs, which is implemented as follows.

At each iteration, a critical slack value s_{crit}^F is first calculated. This value is set at $s_{crit}^F = s_{prop}^F P_{unsch}^F$, where $0 \leq s_{prop}^F < 1$ is a parameter and P_{unsch}^F is the sum of the processing times of the unscheduled jobs. A job j is considered critical if $0 < s_j^F \leq s_{crit}^F$, since it is not yet tardy ($s_j^F > 0$), but is about to become so ($s_j^F \leq s_{crit}^F$); k is then set equal to the number of critical jobs, or 0.5 if no job is critical. The values {0.0, 0.1, …, 0.9} were considered for s_{prop}^F.

The best results of both approaches were compared to choose the most adequate setting. A fixed value of $k=0.5$ was selected for the ATC and AR heuristics. The dynamic approach provided better results for the QATC and QAR procedures, and s_{prop}^F was set at 0.3 and 0.1 for these two rules, respectively.

The QBack_v6 procedure requires a value for w. The values {0.1, 0.2, …, 0.9} were considered for this parameter, and w was set at 0.5. Regarding p_j^{mod}, which is used in all backward rules, the second version was selected for the QBack_v2, QBack_v4 and QBack_v5 procedures, while the third alternative provided the

Table 3
Comparison of quadratic and linear dispatching rules.

n	WPT_SLK versus WSPT					QWMDD versus WMDD				QATC versus ATC				QAR versus AR				QBack_v1 versus Back_v1			
	%ivw_q	%ivw_l	btr	eql	wrs	%ivw_q	%ivw_l	btr	eql	wrs	%ivw_q	%ivw_l	btr	eql	wrs	%ivw_q	%ivw_l	btr	eql	wrs	%ivw_q	%ivw_l	btr	eql	wrs
10	24.63	0.12	1051	161	38	16.55	0.70	922	187	141	7.12	5.97	714	177	359	10.77	2.65	831	231	188	14.71	0.00	868	376	6
15	31.72	0.03	1175	61	14	21.19	0.55	1035	115	100	7.34	6.33	748	155	347	13.17	2.67	920	163	167	19.27	0.03	985	263	2
20	36.77	0.02	1220	25	5	22.26	0.59	1037	115	98	7.79	4.86	769	176	305	14.01	3.96	935	135	180	21.01	0.01	1018	230	2
25	40.62	0.01	1241	6	3	23.33	0.51	1064	113	73	9.04	5.89	810	166	274	13.88	4.11	947	128	175	23.68	0.00	1032	218	0
30	42.77	0.01	1245	3	2	23.58	0.78	1025	120	105	9.63	4.99	846	174	230	13.75	5.20	928	107	215	25.12	0.00	1042	208	0
40	46.05	0.00	1250	0	0	23.12	0.65	1004	136	110	10.55	5.15	867	185	198	14.08	6.48	933	107	210	25.73	0.00	1034	216	0
50	47.00	0.00	1249	0	1	21.87	0.73	982	140	128	10.86	5.66	880	178	192	12.90	6.89	945	100	205	26.86	0.00	1043	207	0
75	49.41	0.00	1250	0	0	20.39	0.90	939	160	151	12.17	5.56	916	176	158	13.40	6.57	969	111	170	27.85	0.00	1037	213	0
100	50.82	0.00	1250	0	0	18.89	0.72	919	181	150	13.15	5.40	937	183	130	13.67	5.91	966	136	148	27.95	0.00	1032	218	0
250	53.06	0.00	1250	0	0	17.85	0.29	944	200	106	14.85	7.06	961	173	116	14.42	6.29	979	151	120	28.64	0.00	1022	228	0
500	53.72	0.00	1250	0	0	17.62	0.08	981	211	58	16.09	6.39	971	171	108	15.75	5.58	1004	158	88	28.30	0.00	1017	233	0
750	53.92	0.00	1250	0	0	18.24	0.07	1025	208	17	16.36	6.55	965	176	109	16.45	5.61	1001	156	93	28.90	0.00	1022	228	0
1000	54.01	0.00	1250	0	0	18.94	0.04	1037	207	6	16.64	6.77	963	166	121	16.68	5.72	1001	154	95	29.02	0.00	1028	222	0
1500	54.14	0.00	1250	0	0	19.12	0.03	1039	209	2	17.03	7.10	965	168	117	16.85	5.72	1000	157	93	29.01	0.00	1023	227	0
2000	54.24	0.00	1250	0	0	19.76	0.01	1043	206	1	17.15	6.62	960	179	111	17.28	5.97	997	152	101	29.06	0.00	1025	225	0
Avg.	46.19	0.01	1229	17	4	20.18	0.44	1000	167	83	12.38	6.02	885	174	192	14.47	5.29	957	143	150	25.67	0.00	1015	234	1

best results for Back, QBack_v1 and QBack_v3. Since the QBack_v6 rule is a combination of QBack_v2 and QBack_v5, for both of which the second version provided the best results, this same version was used in QBack_v6.

3.3. Comparison of the heuristic procedures

A comparison of some of the dispatching rules that specifically consider the quadratic objective with their linear tardiness counterparts is provided in Table 3. For each pair of quadratic heuristic and its linear counterpart, this table gives the mean relative improvement versus the worst result (%ivw) of the quadratic (%ivw_q) and the linear (%ivw_l) procedures, as well as the number of times each quadratic rule provides a solution that is better (btr), equal (eql) or worse (wrs) than the one provided by the corresponding linear procedure. The avg. line in this and subsequent tables provides the mean of the appropriate performance measure over all the instances for all problem sizes.

The particular nature of the squared weighted tardiness problem motivated the use of the relative improvement versus the worst result performance measure, instead of the more usual relative improvement a procedure provides over another heuristic. Indeed, and particularly for instances with a low tardiness factor T and a high range of due dates R, the objective function value given by the heuristic procedures can be equal to 0, meaning that all jobs are completed on time. This is troublesome when the relative improvement is used, since division by 0 is undefined, and motivated the use of a different performance measure.

More specifically, and for a given instance, the relative improvement versus the worst result of heuristic H_i, when compared with heuristics $H_1,H_2,\ldots,H_z$ is calculated as follows. Let ofv_{best} and ofv_{worst} be the best and worst objective function values obtained by all the z heuristic procedures, respectively. When $ofv_{best}=ofv_{worst}$, the relative improvement versus the worst result of heuristic H_i is set at 0. Otherwise, the relative improvement versus the worst result is calculated as $(ofv_{worst}-ofv_{H_i})/ofv_{worst}\times 100$, where ofv_{H_i} is the objective function value of heuristic H_i. This performance indicator thereby measures the relative improvement a given heuristic provides over the worst result obtained among all procedures being compared, and circumvents the division by 0 issue.

Since the purpose of Table 3 is to analyze the performance of each quadratic procedure versus the corresponding linear heuristic, the %ivw_q and %ivw_l values given in this table were calculated separately for each pair of quadratic heuristic and its linear counterpart. More specifically, and using the WPT_SLK and WSPT procedures as an example, these values were calculated as follows. The %ivw_q is the relative improvement WPT_SLK provides over the worst result among the WPT_SLK and WSPT heuristics. Similarly, %ivw_l is the relative improvement given by WSPT versus the worst result between WPT_SLK and WSPT. The same reasoning applies to the other pairs of quadratic heuristic and its linear counterpart.

The results presented in Table 3 show that the heuristics that have been suitably adapted to the quadratic objective outperform their linear tardiness counterparts, which is to be expected. This table shows that the difference in performance is quite significant. Indeed, the quadratic dispatching rules not only provide a much larger relative improvement versus the worst result, but they also obtain better (or equal) results for most, or in some cases actually all, of the test instances.

Therefore, it is most certainly recommended to use heuristics that have been designed in order to take into account the quadratic tardiness objective, instead of simply relying on procedures originally developed for the linear problem. Thus, in the

Table 4
Comparison of forward dispatching rules.

n	%ivw					QAR versus EDD			QAR versus WPT_SLK			QAR versus QWMDD			QAR versus QATC		
	EDD	WPT_SLK	QWMDD	QATC	QAR	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs
10	22.18	27.32	47.81	50.71	47.74	941	50	259	738	475	37	320	504	426	215	732	303
15	23.36	28.74	53.37	54.73	52.31	955	43	252	855	368	27	320	374	556	273	633	344
20	24.02	29.83	55.88	57.13	55.25	957	48	245	911	314	25	267	338	645	354	539	357
25	24.49	30.63	58.19	58.64	57.19	958	49	243	944	280	26	278	300	672	378	482	390
30	24.48	32.16	59.73	60.12	58.97	958	48	244	940	274	36	285	299	666	419	464	367
40	24.48	32.82	60.92	61.23	60.23	962	65	223	976	247	27	250	299	701	440	448	362
50	24.65	33.94	62.53	62.25	61.73	966	68	216	960	262	28	250	306	694	479	423	348
75	24.82	34.93	63.43	63.35	62.98	963	92	195	972	250	28	259	349	642	475	425	350
100	24.82	35.87	64.32	64.16	64.03	963	123	164	955	264	31	254	387	609	494	450	306
250	24.48	37.17	65.03	65.01	65.11	955	148	147	907	332	11	277	508	465	481	511	258
500	24.24	37.85	65.22	65.37	65.52	965	152	133	850	392	8	319	554	377	433	553	264
750	24.33	37.89	65.24	65.42	65.60	964	150	136	836	407	7	333	563	354	445	564	241
1000	24.23	38.02	65.39	65.58	65.75	960	150	140	825	422	3	334	582	334	416	582	252
1500	24.18	38.21	65.52	65.74	65.92	960	151	139	802	447	1	334	602	314	404	602	244
2000	24.17	38.18	65.43	65.68	65.86	952	150	148	799	450	1	348	601	301	392	600	258
Avg.	24.20	34.24	61.20	61.67	60.95	959	99	192	885	345	20	295	438	517	406	534	310

remainder of this section, the linear rules analyzed in Table 3 will no longer be considered.

Table 4 provides a comparison of the forward dispatching rules. More specifically, this table provides the mean relative improvement versus the worst result for each procedure, as well as the number of times the QAR heuristic performed better, equal or worse than the other forward procedures. The relative improvement versus the worst result values presented in Table 4, as well as those given in subsequent tables, were now calculated taking into account all the heuristics considered in the respective table. As an example, the %ivw for the EDD rule is the relative improvement this heuristic provides over the worst result among the EDD, WPT_SLK, QWMDD, QATC and QAR procedures. The same applies to the following tables.

These results show that the WPT_SLK and (even more so) the EDD rules are clearly outperformed by the QWMDD, QATC and QAR heuristics. In particular, the relative improvement versus the worst result values visibly illustrate the quite large performance gap between these procedures. This is to be expected, since the EDD rule only considers the jobs' due dates, while the WPT_SLK heuristic is only suited to situations where most or all jobs will be tardy.

The QWMDD, QATC and QAR heuristics produce somewhat close results. When medium and large size instances are considered, the QAR procedure provides marginally higher relative improvement values. Also, for the largest instances, this heuristic achieves a better result for a slightly larger number of instances. Therefore, the QAR heuristic will then be the only forward dispatching procedure considered in the remainder of this section.

A comparison of the backward dispatching rules is given in Table 5. Again, this table provides the mean relative improvement versus the worst result for each procedure, as well as the number of times the QBack_v6 heuristic generated a solution that is better, equal or worse than those of the other backward procedures.

The results presented in Table 5 show that the QBack_v6 heuristic provides the best performance among the backward scheduling procedures. Indeed, this procedure not only gives the largest relative improvement values, but also generates a better (or equal) solution for most of the test instances. The approach used in the QBack_v6 heuristic therefore managed to successfully combine the priority indices of the QBack_v2 and QBack_v5 procedures, since both these procedures are outperformed by the QBack_v6 rule.

For some instance types, the superior performance provided by the QBack_v6 heuristic is actually much larger than the aggregate results in Table 5 show. In order to avoid an excessive number of tables, however, we have decided to illustrate the effect of the tardiness factor and range of due dates parameters only in the following analysis of the best forward and backward procedures, as well as in the comparison with optimal solutions.

A comparison of the QBack_v6 heuristic with the QAR procedure is provided in Table 6. Indeed, the QBack_v6 and the QAR rules provided the best results among the backward and forward scheduling procedures, respectively. Table 6 gives the mean relative improvement versus the worst result values for these heuristics, as well as the number of times the QBack_v6 procedure performs better, equal or worse than the QAR rule.

These results show that the QBack_v6 heuristic significantly outperforms the QAR procedure. Indeed, the relative improvement versus the worst result is considerably higher for the QBack_v6 dispatching rule. Also, this procedure generates a better (or equal) solution for most of the test instances.

The heuristic procedures were all quite efficient; indeed, the average computation times of all procedures were under 0.5 s for the largest instances. Therefore, we have opted not to include a table with the runtimes. However, we remark that the QBack_v6 procedure is more computationally demanding than the QAR heuristic. For instances with 1500 and 2000 jobs, the QAR procedure required an average computation time, in seconds, equal to 0.039 and 0.069, respectively. The average runtimes of QBack_v6 rule, for these same instances, were correspondingly 0.242 and 0.431. Though the QBack_v6 heuristic therefore requires more computation time than the QAR procedure, it is nevertheless extremely efficient and can handle quite large instances.

The effect of the T and R parameters on the relative performance of these dispatching rules is illustrated in Table 7, for instances with 1000 jobs. This table shows that the tardiness factor T has a significant effect on the relative performance of the heuristics. When T is high, that is when most jobs will be tardy, the heuristics provided close results. For these instances, the relative improvement values are then negligible for both heuristics. However, the QBack_v6 still provides better results for most of the instances.

The difference in performance is however much wider for instances with a medium and low value of T, i.e. instances where the number of tardy jobs is not as high. Under these conditions, the QBack_v6 procedure provides greatly improved results over the QAR rule. In fact, the difference in the relative improvement

Table 5
Comparison of backward dispatching rules.

n	%ivw						v6 versus v1			v6 versus v2			v6 versus v3			v6 versus v4			v6 versus v5		
	v1	v2	v3	v4	v5	v6	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs
10	0.81	4.04	2.78	4.16	4.69	4.80	608	553	89	408	759	83	441	712	97	203	963	84	11	1233	6
15	1.12	3.85	3.34	4.37	4.95	5.16	760	392	98	650	527	73	585	525	140	312	780	158	31	1192	27
20	1.34	4.00	3.55	4.77	5.45	5.68	863	300	87	729	456	65	710	392	148	452	605	193	46	1170	34
25	1.60	4.24	3.68	5.23	5.80	6.05	895	273	82	779	391	80	756	347	147	499	535	216	58	1132	60
30	1.84	4.14	3.79	5.35	5.97	6.27	924	271	55	810	363	77	804	327	119	557	477	216	73	1116	61
40	2.36	3.09	3.98	4.92	5.42	5.81	944	266	40	864	326	60	866	300	84	667	394	189	115	1043	92
50	2.49	3.33	4.19	5.27	5.81	6.22	972	244	34	879	303	68	901	282	67	708	337	205	127	1008	115
75	3.06	2.77	4.42	5.31	5.77	6.25	983	241	26	899	290	61	926	259	65	788	289	173	173	919	158
100	3.56	2.61	4.88	5.74	6.09	6.56	994	238	18	923	279	48	950	258	42	835	289	126	216	878	156
250	4.38	1.71	5.24	5.62	5.91	6.45	991	250	9	920	286	44	960	266	24	905	283	62	310	840	100
500	4.83	1.58	5.73	6.05	6.35	6.85	981	247	22	945	278	27	951	262	37	906	280	64	327	843	80
750	4.97	1.34	5.76	6.07	6.16	6.69	960	249	41	941	276	33	937	261	52	905	274	71	328	844	78
1000	5.18	1.76	6.21	6.63	6.81	7.32	955	246	49	951	273	26	931	260	59	908	279	63	341	837	72
1500	5.25	1.25	5.87	6.29	6.34	6.83	934	252	64	946	280	24	918	263	69	889	284	77	331	840	79
2000	5.27	1.33	6.09	6.28	6.34	6.81	935	251	64	956	273	21	915	262	73	899	276	75	332	840	78
Avg.	3.20	2.74	4.64	5.47	5.86	6.25	913	285	52	840	357	53	837	332	81	696	423	131	188	982	80

Table 6
Comparison of QBack_v6 with QAR.

n	%ivw		QBack_v6 versus QAR		
	QBack_v6	QAR	btr	eql	wrs
10	17.19	0.16	915	172	163
15	18.31	0.13	985	91	174
20	19.29	0.12	1009	64	177
25	19.55	0.15	1040	56	154
30	19.84	0.05	1059	51	140
40	19.21	0.10	1045	66	139
50	18.94	0.11	1034	68	148
75	17.08	0.04	1026	92	132
100	14.51	0.05	998	123	129
250	12.76	0.02	969	148	133
500	11.68	0.00	979	152	119
750	11.78	0.01	985	150	115
1000	11.83	0.00	989	150	111
1500	11.76	0.00	996	152	102
2000	12.18	0.00	1009	150	91
Avg.	15.73	0.06	1003	112	135

Table 7
Comparison of QBack_v6 with QAR for instances with 1000 jobs.

T	R	%ivw		QBack_v6 versus QAR		
		QBack_v6	QAR	btr	eql	wrs
0.2	0.2	6.24	0.00	50	0	0
	0.4	82.05	0.00	50	0	0
	0.6	0.00	0.00	0	50	0
	0.8	0.00	0.00	0	50	0
	1.0	0.00	0.00	0	50	0
0.4	0.2	6.79	0.00	50	0	0
	0.4	7.44	0.00	50	0	0
	0.6	6.51	0.00	50	0	0
	0.8	72.29	0.00	50	0	0
	1.0	100.00	0.00	50	0	0
0.6	0.2	2.82	0.00	50	0	0
	0.4	5.90	0.00	50	0	0
	0.6	4.08	0.00	50	0	0
	0.8	0.47	0.00	49	0	1
	1.0	0.01	0.00	36	0	14
0.8	0.2	0.95	0.00	50	0	0
	0.4	0.09	0.00	50	0	0
	0.6	0.00	0.00	35	0	15
	0.8	0.00	0.00	35	0	15
	1.0	0.00	0.00	35	0	15
1.0	0.2	0.00	0.00	40	0	10
	0.4	0.00	0.00	37	0	13
	0.6	0.00	0.00	39	0	11
	0.8	0.00	0.00	42	0	8
	1.0	0.00	0.00	41	0	9

over the worst results values for the two heuristics is quite large for these instances. Also, the QBack_v6 heuristic provides better results for nearly all of the test instances.

The QBack_v6 rule is then the recommended procedure among all the considered dispatching rules, since it provided the best results and can quickly generate a schedule even for large size problems. The backward scheduling heuristics proved to be greatly superior to the forward scheduling procedures. This is most likely due to the quadratic nature of the objective function, which penalizes tardiness much more heavily than in the linear tardiness setting. In forward scheduling procedures, most or all jobs could be quite early in the first iterations, making it harder to select an appropriate job. Backward scheduling heuristics, however, consider the most important decisions right at the start, since they choose between the tardy jobs first, and this is particularly relevant in a quadratic objective function setting.

3.4. Comparison with optimal results

The QBack_v6 and QAR heuristics will now be compared with optimal solutions, for instances with up to 40 jobs. The optimum results were obtained using the branch-and-bound procedure developed by Schaller and Valente [6]. As detailed in [6], the branch-and-bound algorithm can only solve small to medium size instances within reasonable computation times. Though the runtimes in [6] were obtained using a different computer, it is clear that the heuristics presented in this paper are far more efficient and can solve much larger instances (and harmonizing the runtimes using the benchmarks at www.spec.org or the Dongarra coefficients [31] leads to the same conclusion).

Table 8 gives the mean relative improvement the optimum objective function value provides over the heuristic solution (%ivh), as well as the number of times an optimal solution was generated by the heuristic (n_opt). For a given instance, the relative improvement of the optimum objective function value versus heuristic H_i is calculated as follows. Let ofv_{opt} denote the optimum objective function value. When $ofv_{H_i} = 0$, the relative improvement versus the heuristic solution is set at 0. Otherwise,

Table 8
Comparison with optimum results.

n	QBack_v6		QAR	
	%ivh	n_opt	%ivh	n_opt
10	0.51	836	17.51	242
15	0.59	670	18.71	154
20	0.65	531	19.77	109
25	0.78	485	20.08	77
30	0.70	441	20.38	71
40	0.66	372	19.70	72
Avg.	0.65	556	19.36	121

Table 9
Comparison with optimum results for instances with 25 jobs.

T	R	QBack_v6		QAR	
		%ivh	n_opt	%ivh	n_opt
0.2	0.2	2.46	36	21.07	1
	0.4	0.68	47	73.69	0
	0.6	0.00	50	67.96	15
	0.8	0.00	50	71.62	14
	1.0	0.00	50	65.08	17
0.4	0.2	0.73	17	6.34	0
	0.4	2.02	15	12.28	0
	0.6	2.28	25	21.72	0
	0.8	2.96	34	53.92	1
	1.0	1.49	40	74.10	2
0.6	0.2	0.39	5	4.17	0
	0.4	0.74	12	5.82	0
	0.6	1.65	8	7.28	0
	0.8	1.52	12	7.41	0
	1.0	1.31	7	4.65	0
0.8	0.2	0.20	3	1.45	0
	0.4	0.20	10	1.18	0
	0.6	0.27	7	0.50	0
	0.8	0.21	9	0.85	4
	1.0	0.17	3	0.47	3
1.0	0.2	0.03	4	0.06	2
	0.4	0.05	6	0.09	2
	0.6	0.03	14	0.08	2
	0.8	0.03	5	0.06	7
	1.0	0.07	16	0.09	7

the relative improvement provided by the optimum is calculated as $(ofv_{H_i} - ofv_{opt})/ofv_{H_i} \times 100$.

The relative improvement provided by the optimum when compared with the heuristic solution was chosen over the more usual relative deviation from the optimum due to the same reason that motivated the use of the relative improvement versus the worst result performance measure. Indeed, the relative deviation from the optimum is given by $(ofv_{H_i} - ofv_{opt})/\ ofv_{opt} \times 100$. As previously mentioned, the optimum objective function values can be equal to 0 for instances with low T and high R. Hence, for such instances the relative deviation from the optimum would be undefined, thus motivating the use of the relative improvement versus the heuristic result.

The results in Table 8 again illustrate that the QBack_v6 procedure is superior to the QAR heuristic. Also, these results show that the QBack_v6 rule performs extremely well. Indeed, this rule obtains an optimal solution for many of the test instances, and its objective function values are on average quite close to the optimum (the mean relative improvement provided by the optimum is below 0.8%).

The effect of the T and R parameters is given in Table 9, for instances with 25 jobs. Again, this table shows that the tardiness factor T has a significant effect on the performance of the heuristics. For large values of T, the relative improvement provided by the optimum is quite small. When T assumes a low or medium value, on the other hand, this relative improvement increases. The difference in performance between the QBack_v6 and QAR procedures is much more visible for these latter instance types. Indeed, for these instances, the QBack_v6 heuristic quite significantly outperforms the QWMDD rule. These results are in accordance with those previously presented in Table 7.

These results again show that the QBack_v6 procedure performs quite well. Indeed, it provides results that are usually rather close to the optimum, and it performs most adequately under all instance types. Thus, this heuristic is not only most efficient, but also quite effective.

4. Conclusion

This paper considered the single machine scheduling problem with weighted quadratic tardiness costs. Several efficient dispatching rules were proposed and analyzed. On the one hand, we considered existing rules for the linear problem, as well as heuristics suitably adapted to the quadratic objective function. On the other hand, both forward and backward scheduling procedures were analyzed.

The computational results show that the heuristics that specifically take into account the quadratic objective significantly outperform their linear counterparts. Also, the backward scheduling approach is superior for the considered problem, most likely due to the quadratic nature of the objective function, which penalizes tardiness quite heavily. The difference in performance is significantly more noticeable for the harder instances with a medium or low tardiness factor.

The best of the backward scheduling heuristics is both quite efficient and effective. Indeed, this procedure can quickly generate a schedule even for large instances. Also, it provides quite good results, since its results are usually rather close to the optimum, and it performs most adequately under all instance types.

A possibility for future research would be to investigate the performance of metaheuristics on medium size problems. For these problems, it is possible that these types of procedures could provide better objective function values within reasonable computation times. Another opportunity lies in considering the weighted quadratic tardiness objective function under other machine settings, such as parallel machines or flowshops. Also, it could be interesting to apply the backward scheduling approach to other problems that include tardiness as a major (or sole) component of their objective function.

Acknowledgement

The authors would like to thank an anonymous referee for several helpful comments that helped improve this paper.

References

[1] Wagner BJ, Davis DJ, Kher HV. The production of several items in a single facility with linearly changing demand rates. Decision Sciences 2002;33(3): 317–46.
[2] Taguchi G. Introduction to quality engineering: designing quality into products and processes. Tokyo, Japan: Asian Productivity Organization; 1986.
[3] Hoitomt DJ, Luh PB, Max E, Pattipati KR. Scheduling jobs with simple precedence constraints on parallel machines. IEEE Control Systems Magazine 1990;10(2):34–40.

[4] Sun XQ, Noble JS, Klein CM. Single-machine scheduling with sequence dependent setup to minimize total weighted squared tardiness. IIE Transactions 1999;31(2):113–24.
[5] Thomalla CS. Job shop scheduling with alternative process plans. International Journal of Production Economics 2001;74(1–3):125–34.
[6] Schaller J, Valente JMS. Minimizing the weighted sum of squared tardiness on a single machine. Computers and Operations Research 2012;39(5): 919–28.
[7] Lawler EL. A "Pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. Annals of Discrete Mathematics 1977(1):311–42.
[8] Lenstra JK, Rinnooy Kan AHG, Brucker P. Complexity of machine scheduling problems. Annals of Discrete Mathematics 1977(1):343–62.
[9] Gupta SK, Sen T. Minimizing a quadratic function of job lateness on a single-machine. Engineering Costs and Production Economics 1983;7(3): 187–94.
[10] 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(3):237–43.
[11] 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(2):169–75.
[12] Schaller J. Minimizing the sum of squares lateness on a single machine. European Journal of Operational Research 2002;143(1):64–79.
[13] Soroush HM. A note on "minimizing a weighted quadratic function of job lateness in the single machine system". International Journal of Production Economics 2009;121(1):296–7.
[14] Soroush HM. Single-machine scheduling with inserted idle time to minimise a weighted quadratic function of job lateness. European Journal of Industrial Engineering 2010;4(2):131–66.
[15] Schaller J. Single machine scheduling with early and quadratic tardy penalties. Computers and Industrial Engineering 2004;46(3):511–32.
[16] Valente JMS. Beam search heuristics for the single machine scheduling problem with linear earliness and quadratic tardiness costs. Asia Pacific Journal of Operational Research 2009;26(3):319–39.
[17] Valente JMS. An exact approach for the single machine scheduling problem with linear early and quadratic tardy penalties. Asia Pacific Journal of Operational Research 2008;25(2):169–86.
[18] Valente JMS. Heuristics for the single machine scheduling problem with early and quadratic tardy penalties. European Journal of Industrial Engineering 2007;1(4):431–48.
[19] Valente JMS, Schaller JE. Improved heuristics for the single machine scheduling problem with linear early and quadratic tardy penalties. European Journal of Industrial Engineering 2010;4(1):99–129.
[20] Abdul-Razaq TS, Potts CN, van Wassenhove LNA. Survey of algorithms for the single-machine total weighted tardiness scheduling problem. Discrete Applied Mathematics 1990;26(2–3):235–53.
[21] Potts CN, van Wassenhove LN. Single-machine tardiness sequencing heuristics. IIE Transactions 1991;23(4):346–54.
[22] Sen T, Sulek JM, Dileepan P. Static scheduling research to minimize weighted and unweighted tardiness: a state-of-the-art survey. International Journal of Production Economics 2003;83(1):1–12.
[23] Smith WE. Various optimizers for single-stage production. Naval Research Logistics Quarterly 1956;3:59–66.
[24] Kanet JJ, Li XM. A weighted modified due date rule for sequencing to minimize weighted tardiness. Journal of Scheduling 2004;7(4):261–76.
[25] Vepsalainen APJ, Morton TE. Priority rules for job shops with weighted tardiness costs. Management Science 1987;33(8):1035–47.
[26] Alidaee B, Ramakrishnan KR. A computational experiment of COVERT-AU class of rules for single machine tardiness scheduling problem. Computers and Industrial Engineering 1996;30(2):201–9.
[27] Chambers RJ, Carraway RL, Lowe TJ, Morin TL. Dominance and decomposition heuristics for single-machine scheduling. Operations Research 1991;39(4): 639–47.
[28] Volgenant A, Teerhuis E. Improved heuristics for the n-job single-machine weighted tardiness problem. Computers and Operations Research 1999;26(1): 35–44.
[29] Valente JMS, Alves RAFS. Heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties. Computers and Operations Research 2008;35(11):3696–713.
[30] Holsenback JE, Russell RM, Markland RE, Philipoom PR. An improved heuristic for the single-machine, weighted-tardiness problem. Omega International Journal of Management Science 1999;27(4):485–95.
[31] Dongarra JJ. Performance of various computers using standard linear equations software (Linpack benchmark report). Computer science technical report. University of Tennessee; 2010.