1 Introduction

Scheduling with meeting job due dates have received increasing attention from managers and researchers since the Just-In-Time concept was introduced in manufacturing facilities. While meeting due dates is only a qualitative performance, it usually implies that time dependent penalties are assessed on late jobs but that no benefits are derived from completing jobs early [2]. In this case, these quantitative scheduling objectives associated with tardiness are naturally highlighted in different manufacturing environments. The total tardiness and the total weighted tardiness are two of the most common performance measures in tardiness related scheduling problems [31]. In the traditional scheduling theory, these two kinds of single machine scheduling problems have been extensively studied in the literature under the assumption that the processing time of a job is known in advance and constant throughout the entire operation process [18, 34, 9, 7, 4, 35].

However, there are many practical situations that any delay or waiting in starting to process a job may cause to increase its processing time. Examples can be found in financial management, steel production, equipment maintenance, medicine treatment and so on. Such problems are generally known as time dependent scheduling problems [11]. Among various time dependent scheduling models, there is one case in which job processing times are formulated by piecewise defined functions. In the literature, these jobs with piecewise defined processing times are mainly presented by piecewise linear deterioration and/or step-deterioration [11].

In this paper, the single machine total weighted tardiness scheduling problem with step-deteriorating jobs (SMTWTSD) are addressed. For a step-deteriorating job, if it fails to be processed prior to a pre-specified threshold, its processing time will be increased by an extra time; otherwise, it only needs the basic processing time. The corresponding single machine scheduling problem was considered firstly by Sundararaghavan and Kunnathur [32] for minimizing the sum of the weighted completion times.

The single machine scheduling problem of makespan minimization under step-deterioration effect was investigated in Mosheiov [27], and some simple heuristics were introduced and extended to the setting of multi-machine and multi-step deterioration. Jeng and Lin [15] proved that the problem proposed by Mosheiov [27] is NP-hard in the ordinary sense based on a pseudo-polynomial time dynamic programming algorithm, and introduced two dominance rules and a lower bound to develop a branch and bound algorithm for deriving optimal solutions. Cheng and Ding [3] showed the...
total completion time problem with identical job deteriorating dates is NP-complete and introduced a pseudopolynomial algorithm for the makespan problem. Moreover, Jeng and Lin [12] proposed a branch and bound algorithm incorporating a lower bound and two elimination rules for the total completion time problem. Owing to the intractability of the problem, He et al [14] developed a branch and bound algorithm and a weight combination search algorithm to derive the optimal and near-optimal solutions.

The problem was extended by Cheng et al [8] to the case with parallel machines, where a variable neighborhood search algorithm was proposed to solve the parallel machine scheduling problem. Furthermore, Layegh et al [22] studied the total weighted completion time scheduling problem on a single machine under job step deterioration assumption, and proposed a memetic algorithm with the average percentage error of 2%. Alternatively, batch scheduling with step-deteriorating effect also attracts the attention of some researchers, for example, referring to Barketau et al [3], Leung et al [24] and Mosheiov [25]. With regard to the piecewise linear deteriorating model, the single machine scheduling problem with makespan minimization was firstly addressed in [20]. Following this line of research, successive research works, such as Kubiatk and van de Velde [19], Cheng et al [6], Moslehi and Jafari [28], spurred in the literature.

However, these objective functions with due dates were rarely studied under step-deterioration model in the literature. Guo et al [13] recently found that the total tardiness problem in a single machine is NP-hard, and introduced two heuristic algorithms. As the more general case, the single machine total weighted tardiness problem (SMTWT) has been extensively studied, and several dispatching heuristics were also proposed for obtaining the near-optimal solutions [23]. To the best of our knowledge, there is no research that discusses the SMTWTS problem. Although the SMTWT problem has been proved to be strongly NP-hard [21, 23] and [29, p. 58], the complexity of the SMTWTSD problem under consideration is still open. Therefore, this paper gives the proof of strong NP-hardness of the SMTWTSD problem. Several efficient dispatching heuristics are presented and analyzed as well. These dispatching heuristics can deliver a feasible schedule within reasonable computation time for large-sized problem instances. Moreover, they can be used to generate an initial solution with certain quality required by a meta-heuristics.

The remainder of this paper is organized as follows. Section 2 provides a definition of the single machine total weighted tardiness problem with step-deteriorating jobs and formulates the problem as a mixed integer programming model. The complexity results of the problem considered in this paper and some extended problems are discussed in Section 3. In Section 4 seven improved heuristic procedures are described. Section 5 presents the computational results and analyzes the performances of the proposed heuristics. Finally, Section 6 summarizes the findings of this paper.

2 Problem description and formulation

The problem considered in this paper is to schedule n jobs of the set \( N_n := \{1, 2, \ldots, n\} \), on a single machine for minimizing the total weighted tardiness, where the jobs have stepwise processing times. Specifically, assume that all jobs are ready at time zero and the machine is available at all times. Meanwhile, no preemption is assumed. In addition, the machine can handle only one job at a time, and cannot keep idle until the last job assigned to it is processed and finished. For each job \( j \in N_n \), there is a basic processing time \( a_j \), a due date \( d_j \) and a given deteriorating threshold, also called deteriorating date \( h_j \). If the starting time \( s_j \) of job \( j \in N_n \) is less than or equal to the given threshold \( h_j \), then job \( j \) only requires a basic processing time \( a_j \). Otherwise, an extra penalty \( b_j \) is incurred. Thus, the actual processing time \( p_j \) of job \( j \) can be defined as a step-function: \( p_j = a_j \) if \( s_j \leq h_j \); \( p_j = a_j + b_j \), otherwise. Without loss of generality, the four parameters \( a_j, b_j, d_j \) and \( h_j \) are assumed to be positive integers.

Let \( \pi = (\pi_1, \ldots, \pi_n) \) be a sequence that arranges the current processing order of jobs in \( N_n \), where \( \pi_k \), \( k \in N_n \), indicates the job in position \( k \). The tardiness \( T_j \) of job \( j \) in a schedule \( \pi \) can be calculated by

\[
T_j = \max \{0, s_j + p_j - d_j\}. 
\]

The objective is to find a schedule such that the total weighted tardiness \( \sum w_j T_j \) is minimized, where the weights \( w_j, j \in N_n \) are positive constants. Using the standard three-field notation [12], this problem studied here can be denoted by 1\(|p_j = a_j\) or \( a_j + b_j, h_j\) \( \sum w_j T_j \).

Based on the above description, we formulate the problem as a 0-1 integer programming model. Firstly, the decision variable \( y_{ij}, i, j \in N_n \), is defined such that \( y_{ij} \) is 1 if job \( i \) precedes job \( j \) (not necessarily immediately) on the single-machine, and 0 otherwise. The formulation of the problem is given below.

**Objective function:**

\[
\text{minimize } Z := \sum_{j \in N_n} w_j T_j 
\]
Subject to:

\[ p_j = \begin{cases} a_j, & s_j \leq h_j, \\ a_j + b_j, & \text{otherwise}, \end{cases} \quad \forall j \in \mathbb{N}_n \]  

\[ s_i + p_i \leq s_i + M(1 - y_{ij}), \quad \forall i, j \in \mathbb{N}_n, i < j \]  

\[ s_j + p_j \leq s_i + M y_{ij}, \quad \forall i, j \in \mathbb{N}_n, i < j \]  

\[ s_j + p_j - d_j \leq T_j, \quad \forall j \in \mathbb{N}_n \]  

\[ y_{ij} \in \{0, 1\}, \quad \forall i, j \in \mathbb{N}_n, i \neq j \]  

\[ s_j, T_j \geq 0, \quad \forall j \in \mathbb{N}_n \]  

where \( M \) is a large positive constant such that \( M \to \infty \) as \( n \to \infty \). For example, \( M \) may be chosen as \( M := \max_{j \in \mathbb{N}_n} \{d_j\} + \sum_{j \in \mathbb{N}_n} (a_j + b_j) \).

In the above mathematical model, equation (2.1) represents the objective of minimizing the total weighted tardiness. Constraint (2.2) defines the processing time of each job with the consideration of step-deteriorating effect. Constraint (2.3) and (2.4) determine the starting time \( s_j \) of job \( j \) with respect to the decision variables \( y_{ij} \). Constraint (2.5) calculates the tardiness of job \( j \) with the completion time and the due date. Finally, Constraints (2.6) and (2.7) define the boundary values of variables \( y_{ij}, s_j, T_j, \) for \( i, j \in \mathbb{N}_n \).

3 Complexity results

This section discusses computational complexity of the problem under consideration. It is generally known that once a problem is proved to be strongly NP-hard, it is impossible to find a polynomial time algorithm or a pseudo polynomial time algorithm to produce its optimal solution. Then heuristic algorithms are presented to obtain near optimal solutions for such a problem. Subsequently, the studied single machine scheduling problem is proved to be strongly NP-hard.

**Theorem 1** The problem \(|p_j = a_j\) or \(a_j + b_j, h_j| \sum w_j T_j\) is strongly NP-hard.

The proof of the theorem is based on reducing 3-PARTITION to the problem \(|p_j = a_j\) or \(a_j + b_j, h_j| \sum w_j T_j\).

For a positive integer \( t \in \mathbb{N} \), let integers \( a_j \in \mathbb{N}, j \in \mathbb{N}_{3t} \) and \( b \in \mathbb{N} \) such that \( \frac{b}{4} < a_j < \frac{b}{2}, j \in \mathbb{N}_{3t} \) and \( \sum_{j \in \mathbb{N}_{3t}} a_j = tb \). The reduction is based on the following transformation. Let the number of jobs \( n = 4t - 1 \). Let the partition jobs be such that for \( j \in \mathbb{N}_{3t} \),

\[ d_j = 0, \quad h_j = tb + 2(t - 1), \quad b_j = 1, \quad w_j = a_j - \frac{b}{4} \]  

and

\[ p_j = \begin{cases} a_j, & s_j \leq h_j, \\ a_j + b_j, & \text{otherwise}. \end{cases} \]  

Introduce the notation \( \mathbb{N}_{m,n} := \{m, m + 1, \ldots, n\} \). Let the \( t - 1 \) enforcer jobs be such that for \( j \in \mathbb{N}_{3t+1,4t-1}, \)

\[ d_j = (j - 3t)(b + 1), \quad h_j = d_j - 1, \quad b_j = 1, \quad w_j = (b + b^2)(t^2 - t) \]  

and

\[ p_j = \begin{cases} 1, & s_j \leq h_j, \\ 1 + b_j, & \text{otherwise}. \end{cases} \]  

The first \( 3t \) partition jobs are due at time 0, and the last \((t - 1)\) enforcer jobs are due at \( b + 1, 2b + 2, \ldots \) and so on. The deterioration dates of all partition jobs are set at \( h_j = tb + 2(t - 1), j \in \mathbb{N}_{3t} \); while the deterioration dates of all enforcer jobs are set at \( d_j - 1, \) for \( j \in \mathbb{N}_{3t+1,4t-1} \), that is, one unit before their individual due dates, respectively. We introduce the set notation \( S_i := \{3i - 2, 3i - 1, 3i\} \), \( i \in \mathbb{N}_t \) for sets of three partition jobs. In general, assume that for \( i \in \mathbb{N}_t \),

\[ a_{3i-2} + a_{3i-1} + a_{3i} = b + \delta_i \]  

where, due to \( \frac{b}{4} < a_j < \frac{b}{2}, j \in \mathbb{N}_{3t} \), we must have

\[ \frac{b}{4} < \delta_i < \frac{b}{2} \]  

The quantity \( \delta_i, i \in \mathbb{N}_t \) describes the difference from \( b \) of the sum of basic processing times of jobs in \( S_i \), \( i \in \mathbb{N}_t \). Since \( \sum_{j \in \mathbb{N}_{3t}} a_j = tb \), we deduce that

\[ \sum_{i \in \mathbb{N}_t} \delta_i = 0. \]  

For convenience we define \( \Delta_0 = 0 \) and for \( i \in \mathbb{N}_t \),

\[ \Delta_i = \sum_{j \in \mathbb{N}_i} \delta_j, \]  

That is, \( \Delta_i \) is the accumulative difference from \( ib \) of the cumulative basic processing times of the partition jobs up to the \( i \)-th set \( S_i \) of 3 partition jobs. Note by (3.13), \( \Delta_i = 0 \). The following observation is useful.

**Lemma 1**

\[ \sum_{j \in \mathbb{N}_i} \delta_j = - \sum_{j \in \mathbb{N}_{i-1}} \Delta_j = - \sum_{j \in \mathbb{N}_i} \Delta_j. \]  

(3.15)
The proof can be done by direct calculation. The second equality is obvious because $\Delta_i = 0$. For the first equality, we have
\[
\sum_{j \in \mathbb{N}_t} j\delta_j = \delta_1 + 2\delta_2 + \ldots + t\delta_t
\]
\[
= (\delta_1 + \delta_2 + \ldots + \delta_t) + (\delta_1 + \delta_2 + \ldots + \delta_t) + \delta_t + \ldots
\]
\[
= 0 - \delta_1 - (\delta_1 + \delta_2) - \ldots - (\delta_1 + \delta_2 + \ldots + \delta_t - 1)
\]
where in the last equality we have used the equality $\sum_{j \in \mathbb{N}_t} \delta_j = 0$. Thus we continue by using the definition of $\sum_{j \in \mathbb{N}_t} \delta_j = 0$ to have
\[
\sum_{j \in \mathbb{N}_t} j\delta_j = -\Delta_1 - \Delta_2 - \ldots - \Delta_{t-2} - \Delta_{t-1}
\]
\[
= -\sum_{j \in \mathbb{N}_{t-1}} \Delta_j.
\]
The lemma is proved.

Let $\lfloor x \rfloor$ be the least integer greater than or equal to $x$, and for $k \in \mathbb{N}$, define the index set $J_k := \{3 \lfloor \frac{k}{4} \rfloor - 2, 3 \lfloor \frac{k}{4} \rfloor - 1, \ldots, k\}$.

We are ready to prove the theorem by showing that there exits a schedule with an objective value
\[
z^* = \frac{(t^2 - t)}{8}(b + b^2) + \sum_{k \in \mathbb{N}_t, j \in J_k} a_j \left( a_k - \frac{b}{4} \right)
\] (3.16)
if and only if there exists a solution for the 3-PARTITION problem.

Proof (Proof of Theorem 7)

If the 3-PARTITION problem has a solution, the corresponding 3t jobs thus can be partitioned into $t$ subsets $S_i, i \in \mathbb{N}_t$, of three jobs each, with the sum of the three processing times in each subset equal to $b$, that is, $\delta_i = 0$, for $i \in \mathbb{N}_t$, and the last $t - 1$ jobs are processed exactly during the intervals $[b, b+1], [2b+1, 2b+2], \ldots, [(t-1)b+t-2, (t-1)b+t-1]$.

In this scenario, all the $t - 1$ enforcer jobs are not tardy and all the $3t$ partition jobs are tardy. The tardiness of each partition job equals to its completion time. Moreover, no job is deteriorated.

Let $c_j, j \in \mathbb{N}_{4t}$ be the completion time of each job. Let $S_i$ be the starting time of the first job in each set $S_i, i \in \mathbb{N}_t$. When no job is deteriorated, that is all jobs are processed with basic processing time, the completion time of each job $j$ in $S_i, i \in \mathbb{N}_t$ is given by
\[
c_j = S_i + \sum_{k \in J_j} a_k.
\]
The total weighted tardiness of the 3 jobs in $S_i, i \in \mathbb{N}_t$, equals to
\[
z_i = \sum_{j \in S_i} c_j w_j
\]
\[
= S_i \sum_{j \in S_i} w_j + \sum_{j \in S_i} \sum_{k \in J_j} a_k w_j
\] (3.17)
\[
= [(i-1)b + (i-1)]b + \sum_{j \in S_i} \sum_{k \in J_j} a_k \left( a_j - \frac{b}{4} \right),
\]
since $S_i = (i-1)b + (i-1)$ and $\sum_{j \in S_i} w_j = \frac{b}{4}$, for each $i \in \mathbb{N}_t$. Thus the total weighted tardiness is $\sum_{i \in \mathbb{N}_t} z_i$ which sums to $z^*$ given by (3.16).

Conversely, if such a 3-partition is not possible, there is at least one $\delta_i \neq 0, i \in \mathbb{N}_t$. We next argue that this must imply $\Delta_t := z - z^* > 0$.

If $\Delta_i > 0, i \in \mathbb{N}_{t-1}$, then the $i$-th enforcer job will deteriorate to be processed in the extended time and entail a weighted tardiness. Introduce the notation $x_+ := \max\{x, 0\}$, and let $\mathcal{N}(i), i \in \mathbb{N}_{t-1}$ denote the number of times the value $\Delta_j > 0, j \in \mathbb{N}_t$. For convenience, we define $\mathcal{N}(0) = 0$. The value $\mathcal{N}(i)$, with $0 \leq \mathcal{N}(i) \leq i$, coincides with the cumulative extended processing time of all the enforcer jobs up to the $i$-th enforcer job. This implies that the weighted tardiness of the $i$-th enforcer job is given by
\[
((\Delta_i)_+ + \mathcal{N}(i)) \left( b + b^2 \right) \left( t^2 - t \right).
\]

On the other hand, our configuration of the deterioration dates for all the partition jobs ensures that no partition jobs can deteriorate. Therefore, by recalling equation (3.12), we have in this case that for $i \in \mathbb{N}_t$,
\[
S_i = (i-1)b + (i-1) + \Delta_{i-1} + \mathcal{N}(i-1)
\]
and
\[
\sum_{j \in S_i} w_j = \frac{b}{4} + \delta_i.
\]
In view of equation (3.17) and the weighted tardiness of the enforcer job, we deduce that the change in the objective function value caused by $\delta_i, i \in \mathbb{N}_t$, is given by
\[
\Delta z_i = ((i-1)b + (i-1) + \Delta_{i-1} + \mathcal{N}(i-1)) \delta_i
\]
\[
+ \frac{b}{4} \left( \Delta_{i-1} + \mathcal{N}(i-1) \right)
\]
\[
+ ((\Delta_i)_+ + \mathcal{N}(i)) \left( b + b^2 \right) \left( t^2 - t \right).
\] (3.18)
Therefore the total change in the objective function value is given by
\[ \Delta z = \sum_{i \in N_t} \Delta z_i \]
\[ \geq (1 + b) \sum_{i \in N_t} (i - 1) \delta_i + \sum_{i \in N_t} \Delta_i \left( \delta_i + \frac{b}{4} \right) \]
\[ + (b + b^2)(i^2 - t) \sum_{i \in N_t} N_i(i) \]
\[ = (1 + b) \sum_{i \in N_t} i \delta_i + \sum_{i \in N_t} \Delta_i \left( \delta_i + \frac{b}{4} \right) \]
\[ + (b + b^2)(i^2 - t) \sum_{i \in N_t} N_i(i). \]

In the last equality again we have used \( \Delta_i = \sum_{j \in N_t} \delta_j = 0 \). Now by equation (3.19), we continue to have
\[ \Delta z \geq (1 + b) \sum_{i \in N_t} (-\Delta_i) + \sum_{i \in N_t} \Delta_i \left( \delta_i + \frac{b}{4} \right) \]
\[ + (b + b^2)(i^2 - t) \sum_{i \in N_t} N_i(i) \]
\[ = \sum_{i \in N_t} (-\Delta_i) \left( 1 + \frac{3}{4}b - \delta_i + \frac{b}{4} \right) \]
\[ -(b + b^2)(i^2 - t) \sum_{i \in N_t} N_i(i). \]

If there is a \( \Delta_i > 0 \) for some \( i \in N_t \), then \( \sum_{i \in N_t} N_i(i) \geq 1 \). Recalling that \( \frac{b}{4} > \delta_i > -\frac{b}{4}, \) for all \( i \in \Delta_t \), we have
\[ \sum_{i \in N_t} (-\Delta_i) \left( 1 + \frac{3}{4}b - \delta_i + \frac{b}{4} \right) > (1 + b) \sum_{i \in N_t} (-\Delta_i) \]
\[ > (1 + b) \sum_{i \in N_t} (-\frac{b}{2}) \]
\[ = -\frac{1}{4}b(1 + b)(t - 1). \]

Thus in this case, \( \Delta z > 0 \) by equation (3.19). On the other hand, if all \( \Delta_i \leq 0 \), and at least one \( \delta_i \neq 0, i \in N_t \), then there must be at least one \( \Delta_i < 0 \). Thus in this case
\[ \Delta z = \sum_{i \in N_t} (-\Delta_i) \left( 1 + \frac{3}{4}b - \delta_i + \frac{b}{4} \right) \]
\[ > 0 \]
because \( 1 + \frac{3}{4}b - \delta_i > 0 \) for \( i \in N_t \). Therefore \( \Delta z = 0 \) if and only if \( \delta_i = 0, i \in N_t \). We have proved the theorem.

Remark: with small changes, the above proof also shows that the scheduling problem of total weighted tardiness (without deteriorating jobs), represented by \(| \sum w_i T_j |\), is strongly NP-hard, of which proofs might be found in [21, 23] and [24, p. 58].

After a reflection, we mention that the following three problems also are strongly NP-hard: 1) the problem of total weighted tardiness with deterioration jobs and job release times \( r_j \); 2) the problem of total tardiness with deterioration jobs and job release times \( r_j \); 3) the problem of maximum lateness of a single machine scheduling problem with job release time \( r_j \). Recall that the maximum lateness is defined to be
\[ L_{\text{max}} = \max \{ L_j : L_j = c_j - d_j, j \in N_u \}. \]

Their proofs can be obtained by slightly modifying our previous proof. In fact, the assumption of release times makes the proof a lot of easier. We summarize these results in the following corollaries.

Corollary 2 The problem \(| p_j = a_j or a_j + b_j, h_j, r_j | \sum w_i T_j |\) is strongly NP-hard.

Corollary 3 The problem \(| p_j = a_j or a_j + b_j, h_j, r_j | \sum T_j |\) is strongly NP-hard.

Corollary 4 The problem \(| p_j = a_j or a_j + b_j, h_j, r_j | L_{\text{max}} |\) is strongly NP-hard.

4 Heuristic algorithms

The problem under study is proved strongly NP-hard earlier, then some dispatching heuristics are needed to develop for solving the problem. In this section, the details of these heuristics are discussed. Dispatch heuristics gradually form the whole schedule by adding one job at a time with the best priority index among the unscheduled jobs. There are several existing heuristics designed for the problem without deteriorating jobs. Since the processing times of all jobs considered in our problem depend, respectively, on their starting times, these dispatching heuristics are modified for considering the characteristic of the problem.

Before introducing these procedures, the following notations are defined. Let \( N^* \) denote the ordered set of already-scheduled jobs and \( N^u \) the unordered set of unscheduled jobs. Hence \( N_u = N^* \cup N^u \) when the ordering of \( N^* \) is not in consideration. Let \( t^* \) denote the current time, i.e., the maximum completion time of the scheduled jobs in the set \( N^* \). We shall call \( t^* \) the current time of the sequence \( N^* \). Simultaneously, \( t^* \) is also the starting time of the next selected job. For each of the unscheduled jobs in \( N^u \), its actual processing time is calculated based on its deteriorating date and the current time \( t^* \).
For most scheduling problems with due dates, the earliest due date (EDD) rule is simple and efficient. In this paper, the rule is adopted to obtain a schedule by sorting jobs in non-decreasing order of their due dates. In the same way, the weighted shortest processing time (WSPT) schedules jobs in decreasing order of $w_j/p_j$. At each iteration, the actual processing times of unscheduled jobs in $N^u$ are needed to recalculate. This is because when step-deteriorating effect is considered, the processing time of a job is variable. Even when all jobs are necessarily tardy, the WSPT rule does not guarantee an optimal schedule. Moreover, the weighted EDD (WEDD) rule introduced by Kanet and Li [17] sequences jobs in non-decreasing order of WEDD, where

$$\text{WEDD}_j = \frac{d_j}{w_j},$$  \hspace{1cm} (4.20)

The apparent tardiness cost (ATC) heuristic introduced by Vepsalainen and Morton [34] was developed for the total weighted tardiness problem when the processing time of a job is constant and known in advance. It showed relatively good performance compared with the EDD and the WSPT. The rule with the largest ATC value is selected to be processed. The ATC for job $j$ is determined by the following equation.

$$\text{ATC}_j = \frac{w_j}{p_j} \exp(-\max\{0, d_j - p_j - t^s\}/(\kappa \rho)), \hspace{1cm} (4.21)$$

where, $\kappa$ is a “look-ahead” parameter usually between 0.5 and 4.5, and $\rho$ is the average processing time of the rest of unscheduled jobs. The processing time of an already scheduled job $j \in N^s$ may be $a_j$ or $a_j + b_j$ dependent on if it is deteriorated. Subsequently, the current time $t_s$ is calculated upon the completion of the last job in $N^s$. The parameter $\rho$ is calculated by averaging the processing times of unscheduled jobs in $N^u$ assuming their starting time is at $t^s$.

Based on the cost over time (COVERT) rule [10] and the apparent urgency (AU) rule [26], Alidaee and Ramakrishnan [11] developed a class of heuristic named COVERT-AU for the standard single machine scheduling weighted tardiness problem. The COVERT-AU heuristic combines the two well known methods, i.e. COVERT and AU. At the time $t^s$, the COVERT-AU chooses the next job with the largest priority index CA$_j$ calculated by the equation

$$\text{CA}_j = \frac{w_j}{p_j} (\kappa \rho / (\kappa \rho + \max\{0, d_j - p_j - t^s\}))). \hspace{1cm} (4.22)$$

For the convenience of description, the heuristic with equation (4.22) is denoted by CA in this paper hereafter.

The weighted modified due date (WMDD) rule was developed by Kanet and Li [17] based on modified due date (MDD). In this method, the jobs are processed in non-decreasing order of WMDD. The WMDD is calculated by the equation

$$\text{WMDD}_j = \frac{1}{w_j} \max\{p_j, (d_j - t^*)\}. \hspace{1cm} (4.23)$$

Note that when all job weights are equal, the WMDD is equal to the MDD.

The above heuristics need to recalculate the processing time of the next job for obtaining the priority index except for the EDD and the WEDD. The procedures to recalculate the processing time of the next job is significantly different from those for the problem without step-deterioration. In order to illustrate how these heuristics work, the detailed steps of the WMDD, as an example, are shown in Algorithm 1. For other heuristics, the only difference is the calculation of the priority index.

Algorithm 1 The WMDD
1: Input the initial data of a given instance;
2: Set $N^u = [\cdot], t^* = 0$ and $N^u = \{1, 2, \ldots, n\}$;
3: Set $k = 1$;
4: repeat
5: Compute the processing time of each job in the set $N^u$ based on the current time $t^*$;
6: Calculate the WMDD value of each job in $N^u$ according to equation (4.23);
7: Choose job $j$ from the set $N^u$ with the smallest value WMDD$_j$ to be scheduled in the $k$th position;
8: Update the tardiness of job $j$: $T_j = \max\{t^* + p_j - d_j, 0\}$, and $N^s = [N^u \setminus j]$;
9: Delete job $j$ from $N^u$;
10: $k = k + 1$;
11: until the set $N^u$ is empty
12: Calculate the total weighted tardiness of the obtained sequence $N^u$.

A very effective and simple combination search heuristic for minimizing total tardiness was proposed by Guo et al [13]. The heuristic is called “Simple Weighted Search Procedure” (SWSP) and works as follows: a combined value of parameters $a_j, p_j$ and $h_j$ for job $j$ is calculated as

$$m_j = \gamma_1 d_j + \gamma_2 p_j + \gamma_3 h_j, \hspace{1cm} (4.24)$$

where $\gamma_1$, $\gamma_2$ and $\gamma_3$ are three positive constants. In the SWSP, jobs are sequenced in non-decreasing order of $m$-value. To accommodate the case of the weighted tardiness, equation (4.24) is modified to compute a priority index $m'$ for a job $j$ calculated by the equation

$$m'_j = \frac{1}{w_j} (\gamma_1 d_j + \gamma_2 p_j + \gamma_3 h_j). \hspace{1cm} (4.25)$$
The modified method is called "Modified Simple Weighted Search Procedure" (MSWSP). In equation (4.25), the values of γ₁, γ₂, and γ₃ are determined by using a dynamically updating strategy. The updating strategy is similar to that proposed by Guo et al [13]. Specifically, parameter γ₁ is linearly increased by 0.1 at each iteration and its range is varied from γ₁min to γ₁max. In this study, γ₁min = 0.2 and γ₁max = 0.9 are chosen based on preliminary tests by using randomly generated instances. The parameter γ₂ adopts a similar approach with γ₁min and γ₁max replaced by γ₂min = 0.1 and γ₂max = 0.7, respectively. Once the values of parameters γ₁ and γ₂ are determined, the parameter γ₃ := max{1 − γ₁ − γ₂, 0.1}. The detailed steps of the MSWSP is shown in Algorithm 2.

Algorithm 2 The MSWSP

1: Input the initial data of a given instance;
2: Set c_o := 0, N_u := {1, . . . , n} and N_s := [ ];
3: Generate the entire set Ω of possible triples of weights.
   For each triple (ω₁, ω₂, ω₃) ∈ Ω, perform the following steps;
4: Choose job i with minimal due date to be scheduled in the first position;
5: c_{i[1]} = c_{i[0]} + a_i, N^*_s = [N^*_s, i];
6: delete job i from N_u;
7: set k = 2;
8: repeat
9:   choose job j from N_u with the smallest value m'_j to be
   scheduled in the kth position;
10: if c_{i[k−1]} > h_j then
11:   c_{i[k]} = c_{i[k−1]} + a_j + b_j;
12: else
13:   c_{i[k]} = c_{i[k−1]} + a_j;
14: end if
15: N^*_s = [N^*_s, j];
16: delete job j from N_u;
17: until the set N^*_u is empty.
18: Calculate the total weighted tardiness of the obtained schedule N^*_s;
19: Output the final solution N^*_s.

In order to further improve the quality of the near-optimal solutions, a pairwise swap movement (PS) is incorporated into these heuristics. Let N_s be the sequence output by a heuristic. A swap operation chooses a pair of jobs in positions i and j, 1 ⩽ i, j ⩽ n, from the sequence N_s, and exchanges their positions. Denote the new sequence by N'_s. Subsequently, the total weighted tardiness of the sequence N'_s is calculated. If the new sequence N'_s is better with a smaller tardiness than the incumbent one, the incumbent one is replaced by the new sequence. The swap operation is repeated for any combination of two indices i and j, where 1 ⩽ i < j ⩽ n. Thus the size of the pairwise swap movement (PS) is n(n − 1)/2. In the following, a heuristic with the PS movement is denoted by the symbol ALG_{PS}, where ALG is one of the above mentioned heuristic algorithms. For example, EDD_{PS} represents that the earliest due date rule is applied first, then the solution obtained by the EDD is further improved by the PS movement.

5 Computational experiments

In this section, the computational experiments and results are presented to analyze the performance of the above dispatching heuristics. Firstly, randomly generated test problem instances varying from small to large sizes are described. Next, preliminary experiments are carried out to determine appropriate values for the parameters used in some of the heuristics. Then, a comparative analysis of all seven dispatching heuristics is performed. Furthermore, the results of the best method are compared with optimal solutions delivered by ILOG CPLEX 12.5 for small-sized problem instances. All heuristics were coded in MATLAB 2010 and run on a personal computer with Pentium Dual-Core E5300 2.6 GHz processor and 2 GB of RAM.

5.1 Experimental design

The problem instances were generated using the method proposed by Guo et al [13] as follows. For each job j, a basic processing time a_j was generated from the uniform distribution [1, 100], a weight w_j was generated from the uniform distribution [1, 10], and a deteriorating penalty b_j is generated from the uniform distribution [1, 100×τ], where τ = 0.5. Problem hardness is likely to depend on the value ranges of deteriorating dates and due dates. For each job j, a deteriorating date h_j was drawn from the uniform distribution over three intervals H_1:= [1, A/2], H_2:= [A/2, A] and H_3:= [1, A], where A = ∑_{j∈N_n} a_j. Meanwhile, a due date d_j was generated from the uniform distribution [C_{max}(1 − R/2), C_{max}(1 − T + R/2)], where C_{max} is the value of the maximum completion time obtained by scheduling the jobs in the non-decreasing order of the ratios a_j/b_j. j ∈ N_n, T is the average tardiness factor and R is the relative range of due dates. Both T and R were set at 0.2, 0.4, 0.6, 0.8 and 1.0.

Overall, 75 different combinations were generated for different h, T, and R. For the purpose of obtaining optimal solutions, the number of jobs in each instance was taken to be one of the two sizes of 8 and 10. For the heuristics, the number of jobs can be varied from the small sizes to the large sizes, that comprises 14 sizes including 8, 10, 15, 20, 25, 30, 40, 50, 75,
In each combination of \( h \), \( T \), \( R \), and \( n \), 10 replicates were generated and solved. Thus, there are 750 instances for each problem size, totalling 10500 problem instances, which are available from http://www.researchgate.net/profile/Peng_Guo.

In general, the performances of a heuristic is measured by the average relative percentage deviation (RPD) of the heuristic solution values from optimal solutions value or best value solutions values. The average RPD value is calculated as

\[
\frac{1}{K} \sum_{k=1}^{K} \left( \frac{Z_{\text{best}}^k - Z_{\text{opt}}^k}{Z_{\text{opt}}^k} \right) \times 100,
\]

where \( K \) is the number of problem instances and \( Z_{\text{best}}^k \) and \( Z_{\text{opt}}^k \) are the objective function value of the heuristic method and the optimal solution value for instance \( k \), respectively.

The objective function value delivered by a heuristic may be equal to 0 for an instance with a low tardiness factor \( T \) and a high relative range \( R \) of due dates. It is troublesome to obtain the RPD in this case, since necessarily \( Z_{\text{best}}^k = 0 \), thus it leads to a division by 0 that is undefined. To avoid this situation, in this paper, the relative improvement versus the worst result (RIVW) used by Valente and Schaller [33] is adopted to evaluate the performance of a proposed heuristics. For a given instance, the RIVW for the heuristic solution values from optimal solutions

\[
\text{RIVW} = \left( \frac{Z_{\text{best}}^k - Z_{\text{opt}}^k}{Z_{\text{opt}}^k} \right) \times 100,
\]

where \( Z_{\text{best}}^k \) and \( Z_{\text{opt}}^k \) are the best and worst solution values delivered by all considered heuristics in comparison, respectively. When \( Z_{\text{best}}^k = Z_{\text{opt}}^k \), the RIVW value of a heuristic algorithm is set to 0. Otherwise, the RIVW value is calculated as

\[
\text{RIVW} = \left( \frac{Z_{\text{best}}^k - Z_{\text{opt}}^k}{Z_{\text{best}}^k} \right) \times 100.
\]

Based on the definition of RIVW, it can be observed that the bigger the RIVW value, the better the quality of the corresponding solution.

### 5.2 Parameter selection

In order to select an appropriate value for the parameter \( \kappa \), preliminary tests were conducted on a separate problem set, which contains instances with 20, 50, 100, and 500 jobs. For each of these job sizes \( n \), 5 replicates were produced for each combination of \( n \), \( h \), \( T \) and \( R \). Subsequently, for each problem instance, the objective function value was obtained by using all considered seven heuristics with a candidate value of \( \kappa \).

Then the results of all problem instances were analyzed to determine the best value of \( \kappa \). The candidate values of the parameter \( \kappa \) are chosen to be 0.5, 1.0, . . . , 4.5, which are usually used in the ATC and the CA for traditional single machine problem [34, 1]. The computational tests show that the solutions delivered by the ATC and the CA with \( \kappa = 0.5 \) are relatively better compared with the results obtained by other values of \( \kappa \). Thus, \( \kappa \) is set to 0.5 in the sequel.

### 5.3 Experimental results

The computational results of the proposed seven heuristics are listed in Table 5.1. Specifically, this table provides the mean RIVW values for each heuristic, as well as the number of instances with the best solution (Numbest) found by a heuristic method from 750 instances for each problem size. Each mean RIVW value for a heuristic and a particular problem size in Table 5.1 is the average of RIVW values from the 750 instances for a given problem size. From the table, the EDD and WEDD rules are clearly outperformed by the WSPT, ATC, CA, WMDD, and MSWSP heuristics. The mean RIVW values delivered by EDD and WEDD are significantly less than that given by other heuristics. This is due to the fact that the EDD rule only considers job due dates, while the WEDD relies only on weights and due dates. In addition, the two rules do not consider the effect of step-deterioration in calculating the priority index. It is worthwhile to note that the results achieved by the WEDD is better than that given by the EDD.

As far as the remaining methods, the RIVW values obtained by the WSPT and MSWSP heuristics are worse than that of other three (ATC, CA and WMDD) heuristics. But the performance of the MSWSP is better than the WSPT. This indicates that the MSWSP can produce good results for our problem, but it fails to obtain better solutions compared with the three improved methods (ATC, CA and WMDD).

There is no significant difference between the results produced by the ATC, CA and WMDD heuristics. The CA procedure provides slightly higher mean relative improvement versus the result best values. For large-sized instances, the number of the best solutions achieved by the CA is much higher than the ATC and the WMDD. Therefore, the CA procedure can be deemed as the best one among the seven dispatching heuristic algorithms.

Computational times of these dispatching heuristics for each job size are listed in Table 5.2. As the size of instances increases, the CPU time of all methods grows at different degrees. Totally, the MSWSP consumes the most time compared with other algorithms, but surprisingly its maximum CPU time is only 5.71 seconds for the intractable instance with 1000 jobs. The average CPU time of the CA which is 0.66 seconds is less than that of the MSWSP. Since the EDD and the WEDD mainly depend on the ranking index of all jobs’ due dates, their computational times are less than
Table 5.1  Computational results of the dispatching heuristic algorithms

| n   | RIVW(%) | Numbest |
|-----|---------|---------|
|     | EDD     | WSPT    | WEDD   | ATC    | CA     | WMDD   | MSWSP  |
|     | 8       | 15.56   | 39.87  | 28.03  | 48.50  | 55.01  | 49.06  | 50.40  |
|     | 10      | 15.38   | 40.76  | 26.66  | 49.31  | 57.68  | 52.36  | 50.57  |
|     | 15      | 16.97   | 40.15  | 28.23  | 55.60  | 62.25  | 57.82  | 53.01  |
|     | 20      | 17.80   | 41.35  | 27.50  | 57.73  | 65.56  | 69.42  | 54.13  |
|     | 25      | 16.92   | 41.11  | 27.60  | 58.49  | 67.46  | 62.92  | 53.26  |
|     | 30      | 17.39   | 41.47  | 27.41  | 59.93  | 68.64  | 64.49  | 52.52  |
|     | 40      | 18.44   | 41.10  | 26.38  | 61.81  | 70.31  | 66.22  | 52.84  |
|     | 50      | 18.76   | 40.59  | 27.31  | 62.62  | 71.22  | 67.17  | 52.30  |
|     | 75      | 19.00   | 41.34  | 27.41  | 63.65  | 72.03  | 68.59  | 51.70  |
|     | 100     | 20.13   | 41.07  | 27.73  | 64.49  | 72.81  | 69.09  | 51.86  |
|     | 250     | 19.54   | 41.37  | 27.46  | 65.91  | 73.74  | 69.47  | 49.92  |
|     | 500     | 19.55   | 41.47  | 27.42  | 66.47  | 74.01  | 69.42  | 49.29  |
|     | 750     | 19.46   | 41.54  | 27.37  | 66.44  | 74.08  | 69.30  | 49.12  |
|     | 1000    | 19.50   | 41.46  | 27.32  | 66.38  | 74.13  | 69.32  | 48.92  |
| Total|        |         |        |        |        |        |        |       |
| Avg. | 18.18   | 41.05   | 27.46  | 60.52  | 68.50  | 63.97  | 51.42  | 67     |

Scheduling step-deteriorating jobs to minimize the total weighted tardiness on a single machine
Table 5.2 Computational times of dispatching heuristic algorithms

| n    | EDD | WSPT | WEDD | ATC | CA | WMDD | MSWSP |
|------|-----|------|------|-----|----|------|-------|
| 8    | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | 0.01  |
| 10   | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | 0.01  |
| 15   | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | 0.02  |
| 20   | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | 0.03  |
| 25   | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | 0.04  |
| 30   | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | 0.05  |
| 40   | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | 0.06  |
| 50   | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | <0.01 | 0.08  |
| 75   | <0.01 | 0.01  | <0.01 | 0.01  | 0.02  | 0.01  | 0.14  |
| 100  | <0.01 | 0.02  | <0.01 | 0.05  | 0.02  | 0.20  |       |
| 250  | <0.01 | 0.12  | <0.01 | 0.14  | 0.30  | 0.14  | 0.65  |
| 500  | <0.01 | 0.47  | <0.01 | 0.58  | 1.19  | 0.56  | 1.81  |
| 750  | <0.01 | 1.11  | <0.01 | 1.33  | 2.72  | 1.31  | 3.48  |
| 1000 | <0.01 | 2.05  | <0.01 | 2.42  | 4.91  | 2.39  | 5.67  |
| Avg. | <0.01 | 0.27  | <0.01 | 0.32  | 0.66  | 0.32  | 0.88  |
0.01 seconds. It was observed that the CPU time of the other five algorithms follow almost the same trend.

Subsequently, the solutions delivered by the seven dispatching heuristics are improved by the PS movement. The seven heuristics with the PS movement are denoted by EDDPS, WSPTPS, WEDDPS, ATCPSPS, CAPS, WMDDPS and MSWSPPS, respectively. A comparison of these methods is given in Table 5.3. Again, this table lists the mean relative improvement versus the worst result (RIVW) for each algorithm and the number of instances with the best solution found by each of the seven algorithms with the PS movement.

From Table 5.3 it can be observed that the CAPS provides the best performance among these procedures. In fact, the CAPS not only gives the largest RIVW value, but also gives a better solution for most of the instances. For medium- and large-sized instances, the CAPS shows better performance in terms of the number of best solution (Numopt) compared with the other six methods. In particular, for the case with 1000 jobs, the CAPS gives the best solutions for 538 over the 750 instances. It is found that CAPS delivers best solutions for on average 473 out of 750 instances for all job sizes. The average RIVW values delivered by the WEDDPS, ATCPSPS, CAPS, WMDDPS and MSWSPPS are more than 40%. The RIVW value of the WSPT is only 15.86%, significantly less than that achieved by the other six methods.

Computational times of these methods are listed in Table 5.4. The average computational times of the seven methods are very close, and the gap of the average CPU times between these methods is not more than one second. As expected, the CPU times of these algorithms are increased as the number of jobs increases. But the computational times of the seven methods are not more than 80 seconds even for the 1000-job case.

In order to further analyze the results, the one-way Analysis of Variance (ANOVA) is used to check whether the observed difference in the RIVW values for the dispatching heuristics with the PS movement are statistically significant. The WSPTPS is removed from the statistical analysis since it is clearly worse than the remaining ones. The means plot and the Fisher Least Significant Difference (LSD) intervals at the 95% confidence level are shown in Figure 5.1. If the LSD intervals of two algorithms are not overlapped, the performances of the tested algorithms are statistically significantly different. Otherwise, the performances of the two algorithms do not lie significantly in the difference. As it can be seen, ATCPSPS, CAPS and WMDDPS are not statistically different because their confidence intervals are overlapped. This observation is really important since it gives the conclusion that can not be obtained from a table of average RIVW results. Moreover, the CAPS and WMDDPS are statistically significantly better than the other three methods (EDDPS, WEDDPS and MSWSPPS) by having their LSD intervals of the RIVW values higher than those of other methods. However, the WEDDPS and the MSWSPPS are not statistically different due to their overlapping confidence intervals.

To evaluate the effect of the PS movement, a comparison of the CA heuristic with the CAPS procedure is provided in Table 5.5. Table 5.5 gives the mean relative improvements versus the worst result values for the two procedures, as well as the number of times the CAPS procedure performs better than (Numbetter) or equal the CA (Numequal). Since the heuristic CA always gives worse result when comparing with the CAPS, its RIVW values in Table 5.5 equal to zero for all instances. On average, the RIVW values obtained by the CAPS are 24.71% better than the RIVW values achieved by the CA. These results show that the PS movement can significantly improve the quality of solutions delivered by the CA heuristic for most of the instances.

### Table 5.5 Comparison of CA with CAPS

| n   | CA | CAPS | Numequal | Numbetter |
|-----|----|------|----------|-----------|
| 5   | 0.00 | 22.86    | 631       | 579       |
| 10  | 0.00 | 19.56    | 171       | 597       |
| 15  | 0.00 | 24.22    | 80        | 667       |
| 20  | 0.00 | 25.73    | 59        | 691       |
| 25  | 0.00 | 26.08    | 44        | 706       |
| 30  | 0.00 | 26.09    | 47        | 703       |
| 40  | 0.00 | 27.92    | 46        | 704       |
| 50  | 0.00 | 26.74    | 55        | 695       |
| 75  | 0.00 | 27.63    | 66        | 684       |
| 100 | 0.00 | 28.09    | 72        | 678       |
| 250 | 0.00 | 25.07    | 103       | 647       |
| 500 | 0.00 | 24.40    | 110       | 640       |
| 750 | 0.00 | 23.45    | 117       | 633       |
| 1000| 0.00 | 22.86    | 119       | 631       |
| Avg.| 0.00 | 24.71    | 94        | 656       |

Next, the CAPS heuristic is compared with the optimal solutions delivered by the CPLEX 12.5 for instances with 8, 10 and 12 jobs. Here, the performance of the heuristic is measured by the mean relative improvement of the optimum objective function value versus the heuristic solution (RIVH), as well as the number of instances with the optimal solution given by the heuristic (Numopt). For a given instance, the relative improvement of the optimum objective function value versus heuristic is calculated as follows. When \( Z_{opt} = Z_{alg} \), the RIVH value is set to 0. Otherwise, the
Table 5.3 Computational results of dispatching heuristic algorithms with the PS movement

| n    | EDD<sub>PS</sub> | WSPT<sub>PS</sub> | WEDD<sub>PS</sub> | ATC<sub>PS</sub> | CA<sub>PS</sub> | WMDD<sub>PS</sub> | MSWSP<sub>PS</sub> | EDD<sub>PS</sub> | WSPT<sub>PS</sub> | WEDD<sub>PS</sub> | ATC<sub>PS</sub> | CA<sub>PS</sub> | WMDD<sub>PS</sub> | MSWSP<sub>PS</sub> |
|------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 8    | 26.95           | 12.84           | 29.26           | 31.67           | 33.12           | 31.90           | 32.54           | 243             | 441             | 288             | 506             | 535             | 514             | 422             |
| 10   | 28.76           | 14.37           | 32.27           | 35.18           | 36.57           | 35.79           | 35.02           | 162             | 339             | 179             | 460             | 483             | 468             | 335             |
| 15   | 33.13           | 15.04           | 36.71           | 40.43           | 42.39           | 40.40           | 39.80           | 148             | 233             | 140             | 392             | 435             | 389             | 218             |
| 20   | 35.00           | 15.81           | 38.67           | 43.41           | 44.91           | 43.64           | 42.11           | 139             | 202             | 110             | 370             | 416             | 370             | 181             |
| 25   | 35.88           | 14.58           | 40.10           | 44.61           | 46.77           | 44.86           | 42.06           | 131             | 166             | 98              | 346             | 426             | 358             | 160             |
| 30   | 36.92           | 14.06           | 41.30           | 46.29           | 47.81           | 46.46           | 43.76           | 134             | 181             | 113             | 333             | 423             | 351             | 158             |
| 40   | 37.79           | 16.21           | 42.93           | 48.17           | 50.03           | 48.53           | 45.07           | 135             | 174             | 104             | 318             | 420             | 365             | 134             |
| 50   | 37.99           | 16.08           | 43.40           | 48.96           | 50.79           | 49.45           | 45.68           | 130             | 154             | 98              | 305             | 441             | 320             | 133             |
| 75   | 38.47           | 16.16           | 44.23           | 50.22           | 52.24           | 50.65           | 46.34           | 137             | 164             | 100             | 272             | 460             | 304             | 139             |
| 100  | 38.97           | 15.78           | 44.96           | 51.10           | 53.00           | 51.70           | 46.68           | 136             | 153             | 101             | 264             | 471             | 291             | 133             |
| 250  | 38.98           | 16.94           | 45.44           | 51.99           | 54.21           | 52.49           | 47.02           | 141             | 159             | 101             | 234             | 527             | 254             | 140             |
| 500  | 38.87           | 17.72           | 45.43           | 52.20           | 54.56           | 52.66           | 46.95           | 141             | 161             | 103             | 244             | 520             | 263             | 140             |
| 750  | 38.58           | 18.45           | 45.36           | 51.88           | 54.51           | 52.50           | 46.82           | 144             | 166             | 100             | 252             | 523             | 254             | 144             |
| 1000 | 38.37           | 18.02           | 45.20           | 51.67           | 54.52           | 52.44           | 46.64           | 147             | 166             | 105             | 240             | 538             | 252             | 146             |
| Total| 36.05           | 15.86           | 41.09           | 46.27           | 48.24           | 46.68           | 43.36           | 148             | 204             | 124             | 324             | 473             | 340             | 184             |
Table 5.4 Computational times of dispatching heuristic algorithms with the PS movement

| n    | EDD<sub>PS</sub> | WSPT<sub>PS</sub> | WEDD<sub>PS</sub> | ATC<sub>PS</sub> | CA<sub>PS</sub> | WMDD<sub>PS</sub> | MSWSP<sub>PS</sub> |
|------|------------------|-------------------|-------------------|------------------|----------------|------------------|------------------|
| 8    | <0.01            | <0.01             | <0.01             | <0.01            | <0.01         | <0.01            | 0.01             |
| 10   | <0.01            | <0.01             | <0.01             | <0.01            | <0.01         | <0.01            | 0.01             |
| 15   | <0.01            | <0.01             | <0.01             | <0.01            | <0.01         | <0.01            | 0.02             |
| 20   | <0.01            | <0.01             | <0.01             | <0.01            | <0.01         | <0.01            | 0.03             |
| 25   | <0.01            | <0.01             | <0.01             | <0.01            | 0.01          | <0.01            | 0.04             |
| 30   | <0.01            | 0.01              | <0.01             | 0.01             | 0.01          | 0.01             | 0.05             |
| 40   | 0.01             | 0.02              | 0.02              | 0.02             | 0.02          | 0.02             | 0.08             |
| 50   | 0.02             | 0.03              | 0.02              | 0.03             | 0.04          | 0.03             | 0.11             |
| 75   | 0.07             | 0.08              | 0.07              | 0.08             | 0.10          | 0.08             | 0.21             |
| 100  | 0.14             | 0.16              | 0.14              | 0.17             | 0.19          | 0.16             | 0.34             |
| 250  | 1.47             | 1.56              | 1.47              | 1.60             | 1.75          | 1.60             | 2.12             |
| 500  | 9.71             | 9.87              | 9.79              | 10.10            | 10.66         | 10.12            | 11.56            |
| 750  | 30.3             | 30.28             | 30.35             | 30.96            | 32.06         | 30.96            | 33.69            |
| 1000 | 69.43            | 68.93             | 69.61             | 70.29            | 72.10         | 70.27            | 74.75            |
| Avg. | 7.94             | 7.92              | 7.96              | 8.09             | 8.35          | 8.09             | 8.79             |
RIVH value is calculated as

\[ \text{RIVH} = \frac{(Z_{\text{alg}} - Z_{\text{opt}})}{Z_{\text{alg}}} \times 100. \]

According to the definition of the RIVH value, the smaller the mean RIVH value, the better the quality of the solutions delivered by a heuristic is for the set of instances.

The comparison results were shown in Table 5.6 and 5.7. The two tables show that the tardiness factor \( T \) and the deteriorating interval \( H \) can significantly affect the performance of the heuristic \( \text{CA}_{\text{PS}} \). For large values of \( T \), the RIVH values are relatively small and the corresponding objective function values are on average quite close to the optimum achieved by the CPLEX. When the deteriorating interval is \( H_2 \), most jobs tend to have large deteriorating dates, and may not be deteriorated. Thus the RIVH values of the test instances with \( H_2 \) are less than that of the instances with \( H_1 \) and \( H_3 \). This observation has been demonstrated by Cheng et al.\[8\] for parallel machine scheduling problem. In this case, the heuristic \( \text{CA}_{\text{PS}} \) can give more optimal solutions compared with the instances with \( H_1 \) and \( H_3 \). Overall, the \( \text{CA}_{\text{PS}} \) is effective in solving the problem under consideration since the maximum mean RIVH value is below 20% for instances with 10 jobs.

### Table 5.6 Comparison with optimum results for instances with 8 jobs.

| \( T \) | \( R \) | \( Z_{\text{opt}} \) | \( H_1 \) | \( H_2 \) | \( H_3 \) | \( \text{Num}_{\text{opt}} \) |
|------|------|----------|------|------|------|----------|
| 0.2  | 0.2  | 40.39    | 1.07 | 16.94| 0     | 7         | 5         |
| 0.4  | 0.4  | 48.40    | 0.47 | 34.66| 3     | 8         | 4         |
| 0.6  | 0.6  | 9.46     | 7.97 | 35.52| 7     | 7         | 5         |
| 0.8  | 0.8  | 18.78    | 10.00| 0.00 | 7     | 9         | 10        |
| 1    | 1    | 9.29     | 4.56 | 6.25 | 7     | 8         | 8         |
| 0.4  | 0.2  | 13.50    | 2.32 | 14.23| 1     | 7         | 2         |
| 0.4  | 0.4  | 7.00     | 0.00 | 7.50 | 3     | 10        | 3         |
| 0.6  | 0.6  | 12.88    | 4.62 | 4.48 | 3     | 6         | 8         |
| 0.8  | 0.8  | 33.70    | 7.99 | 14.46| 4     | 6         | 5         |
| 1    | 1    | 19.30    | 9.01 | 18.17| 8     | 5         | 3         |
| 0.6  | 0.2  | 8.50     | 1.31 | 1.90 | 3     | 6         | 7         |
| 0.4  | 0.4  | 9.94     | 0.98 | 8.97 | 2     | 6         | 4         |
| 0.6  | 0.6  | 6.38     | 0.50 | 2.27 | 4     | 5         | 5         |
| 0.8  | 0.8  | 5.84     | 2.55 | 10.09| 5     | 7         | 1         |
| 1    | 1    | 13.31    | 0.03 | 4.58 | 3     | 8         | 5         |
| 0.8  | 0.2  | 3.43     | 0.36 | 3.43 | 7     | 7         | 5         |
| 0.4  | 0.4  | 9.58     | 0.14 | 1.09 | 3     | 8         | 6         |
| 0.6  | 0.6  | 4.17     | 0.12 | 1.71 | 3     | 9         | 5         |
| 0.8  | 0.8  | 3.32     | 0.87 | 0.48 | 4     | 7         | 7         |
| 1    | 1    | 3.25     | 1.06 | 1.85 | 5     | 6         | 7         |
| 1    | 0.2  | 0.77     | 0.13 | 0.78 | 8     | 7         | 6         |
| 0.4  | 0.4  | 1.16     | 0.00 | 2.35 | 6     | 9         | 3         |
| 0.6  | 0.6  | 4.02     | 0.00 | 2.36 | 4     | 9         | 6         |
| 0.8  | 0.8  | 2.79     | 0.00 | 1.48 | 4     | 9         | 6         |
| 1    | 1    | 0.75     | 0.13 | 1.28 | 6     | 8         | 5         |

**Avg.** 11.24 2.25 7.87 4.40 7.36 5.24
Table 5.7 Comparison with optimum results for instances with 10 jobs

| T  | R  | RIVH(%) | Numopt | H1  | H2  | H3  |
|----|----|---------|--------|-----|-----|-----|
|    |    |         |        | H1  | H2  | H3  |
|    |    |         |        |     |     |     |
| 0.2 | 0.2 | 55.29 | 2.52 | 38.39 | 2 | 7 | 3 |
| 0.4 | 0.00 | 30.71 | 30.71 | 36.71 | 3 | 5 | 5 |
| 0.6 | 0.10 | 5.64 | 16.71 | 58.71 | 5 | 8 | 8 |
| 0.8 | 20.00 | 16.07 | 17.36 | 78.36 | 7 | 8 | 5 |
| 1  | 14.85 | 22.31 | 31.82 | 76.82 | 7 | 6 | 5 |
| 0.4 | 0.2  | 18.28 | 0.88 | 10.54 | 1 | 7 | 4 |
| 0.4 | 20.75 | 2.59 | 27.79 | 1 | 5 | 0 |
| 0.6 | 43.98 | 7.06 | 21.03 | 0 | 5 | 3 |
| 0.8 | 31.10 | 5.64 | 16.71 | 5 | 8 | 8 |
| 0.8 | 14.85 | 7.06 | 21.03 | 0 | 5 | 3 |
| 0.4 | 10.27 | 12.77 | 4.09 | 1 | 7 | 4 |
| 0.4 | 7.45 | 5.21 | 5.36 | 5 | 6 | 5 |
| 0.6 | 3.01 | 1.55 | 5.14 | 2 | 4 | 4 |
| 0.8 | 7.70 | 3.44 | 5.92 | 2 | 5 | 3 |
| 1  | 8.51 | 2.71 | 5.88 | 1 | 3 | 3 |
| 0.8 | 0.2  | 0.63 | 0.75 | 4 | 7 | 5 |
| 0.4 | 8.01 | 1.03 | 3.61 | 1 | 8 | 3 |
| 0.6 | 3.76 | 1.03 | 2.81 | 3 | 3 | 3 |
| 0.8 | 3.52 | 0.29 | 2.05 | 5 | 7 | 2 |
| 1  | 4.66 | 1.06 | 3.66 | 3 | 6 | 3 |
| 0.4 | 0.67 | 0.18 | 1.14 | 6 | 6 | 4 |
| 0.4 | 2.66 | 0.20 | 1.61 | 2 | 7 | 4 |
| 0.6 | 2.93 | 0.13 | 0.92 | 1 | 9 | 5 |
| 0.8 | 1.44 | 0.01 | 0.92 | 4 | 9 | 5 |
| 1  | 1.37 | 0.02 | 1.51 | 4 | 9 | 3 |
| Avg. | 15.96 | 5.01 | 11.65 | 2.88 | 6.04 | 3.92 |

6 Conclusions

The total weighted tardiness as the general case has more important meaning in the practical situation. In this paper, the single machine scheduling problem with step-deteriorating jobs for minimizing the total weighted tardiness was addressed. Based on the characteristics of this problem, a mathematical programming model is presented for obtaining the optimal solution, and, the proof of the strong NP-hardness for the problem under consideration is given. Afterwards, seven heuristics are designed to obtain the near-optimal solutions for randomly generated problem instances. Computational results show that these dispatching heuristics can deliver relatively good solutions at low cost of computational time. Among these dispatching heuristics, the CA procedure as the best solution method can quickly generate a good schedule even for large instances. Moreover, the test results clearly indicate that these methods can be significantly improved by the pairwise swap movement.

In the future, the consideration of developing meta-heuristics such as a genetic algorithm or ant colony optimization approach might be an interesting issue. For medium-sized problems, it is possible that a meta-heuristic could give better solutions within reasonable computational time. Another consideration is to investigate the total weighted tardiness problem with the step-deteriorating effects under other machine settings, such as parallel machines or flow-shops.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 51405403) and the Fundamental Research Funds for the Central Universities (No. 2682014BR019).

References

1. Alidaee B, Ramakrishnan K (1996) A computational experiment of covert-au class of rules for single machine tardiness scheduling problem. Computers & industrial engineering 30(2):201–209
2. Baker KR, Trietsch D (2009) Principles of Sequencing and Scheduling. John Wiley & Sons
3. Barketau M, Cheng TE, Ng C, Kotov V, Kovalyov MY (2008) Batch scheduling of step deteriorating jobs. Journal of Scheduling 11(1):17–28
4. Bilge Ü, Kurtulan M, Kraç F (2007) A tabu search algorithm for the single machine total weighted tardiness problem. European Journal of Operational Research 176(3):1423–1435
5. Cheng T, Ding Q (2001) Single machine scheduling with step-deteriorating processing times. European Journal of Operational Research 134(3):623–630
6. Cheng T, Ding Q, Kovalyov MY, Bachman A, Janiak A (2003) Scheduling jobs with piecewise linear decreasing processing times. Naval Research Logistics (NRL) 50(6):531–554
7. Cheng TE, Ng C, Yuan J, Liu Z (2005) Single machine scheduling to minimize total weighted tardiness. European Journal of Operational Research 165(2):423–443
8. Cheng W, Guo P, Zhang Z, Zeng M, Liang J (2012) Variable neighborhood search for parallel machines scheduling problem with step deteriorating jobs. Mathematical Problems in Engineering 928312:1–20
9. Cauwels H, Potts CN, Van Wassenhove LN (1998) Local search heuristics for the single machine total weighted tardiness scheduling problem. INFORMS Journal on computing 10(3):341–350
10. Fisher ML (1976) A dual algorithm for the one-machine scheduling problem. Mathematical programming 11(1):229–251
11. Gawiejnowicz S (2008) Time-dependent scheduling. Springer
12. Graham RL, Lawler EL, Lenstra JK, Kan A (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 5:287–326
13. Guo P, Cheng W, Wang Y (2014) A general variable neighborhood search for single-machine total tardiness scheduling problem with step-deteriorating jobs. Journal of Industrial and Management Optimization 10(4):1071–1090, DOI 10.3934/jimo.2014.10.1071

14. He C, Wu C, Lee W (2009) Branch-and-bound and weight-combination search algorithms for the total completion time problem with step-deteriorating jobs. Journal of the Operational Research Society 60(12):1759–1766

15. Jeng A, Lin B (2004) Makespan minimization in single-machine scheduling with step-deterioration of processing times. Journal of the Operational Research Society 55(3):247–256

16. Jeng A, Lin B (2005) Minimizing the total completion time in single-machine scheduling with step-deteriorating jobs. Computers & operations research 32(3):521–536

17. Kanet JJ, Li X (2004) A weighted modified due date rule for sequencing to minimize weighted tardiness. Journal of Scheduling 7(4):261–276

18. Kouklas C (2010) The single-machine total tardiness scheduling problem: review and extensions. European Journal of Operational Research 202(1):1–7

19. Kubik W, van de Velde S (1998) Scheduling deteriorating jobs to minimize makespan. Naval Research Logistics (NRL) 45(5):511–523

20. Kunnathur AS, Gupta SK (1990) Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem. European Journal of Operational Research 47(1):56–64

21. Lawler EL (1977) A ‘pseudopolynomial’ algorithm for sequencing jobs to minimize total tardiness. Annals of discrete Mathematics 1:331–342

22. Layegh J, Jolai F, Amalnik MS (2009) A memetic algorithm for minimizing the total weighted completion time on a single machine under step-deterioration. Advances in Engineering Software 40(10):1074–1077

23. Lenstra J, Kan AR, Brucker P (1977) Complexity of job sequencing problems. Annals of Discrete Mathematics 1:343–362

24. Leung J, Ng C, Cheng T (2008) Minimizing sum of completion times for batch scheduling of jobs with deteriorating processing times. European Journal of Operational Research 187(3):1090–1099

25. Mor B, Mosheiov G (2012) Batch scheduling with step-deteriorating processing times to minimize flowtime. Naval Research Logistics (NRL) 59(8):587–600

26. Morton T, Rachamadugu R, Vepsalainen A (1984) Accurate mayopic heuristics for tardiness scheduling. GSIA Working Paper No, 36-83-84, Carnegie-Mellon University, PA

27. Mosheiov G (1995) Scheduling jobs with step-deterioration; minimizing makespan on a single-and multi-machine. Computers & industrial engineering 28(4):869–879

28. Moslehi G, Jafari A (2010) Minimizing the number of tardy jobs under piecewise-linear deterioration. Computers & Industrial Engineering 59(4):573–584

29. Pinedo M (2012) Scheduling: theory, algorithms, and systems. Springer

30. Potts C, van Wassenhove LN (1991) Single machine tardiness sequencing heuristics. IIE transactions 23(4):346–354

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

32. Sundararaghavan P, Kunnathur A (1994) Single machine scheduling with start time dependent processing times: some solvable cases. European Journal of Operational Research 78(3):394–403

33. Valente J, Schaller JE (2012) Dispatching heuristics for the single machine weighted quadratic tardiness scheduling problem. Computers & Operations Research 39(9):2223–2231

34. Vepsalainen AP, Morton TE (1987) Priority rules for job shops with weighted tardiness costs. Management science 33(8):1035–1047

35. Wang X, Tang L (2009) A population-based variable neighborhood search for the single machine total weighted tardiness problem. Computers & Operations Research 36(6):2105–2110
