Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 679702, 10 pages
http://dx.doi.org/10.1155/2014/679702

Research Article

Approximation Algorithms and an FPTAS for the Single Machine Problem with Biased Tardiness Penalty

G. Moslehi and K. Kianfar

Department of Industrial and Systems Engineering, Isfahan University of Technology, Isfahan 84156 83111, Iran

Correspondence should be addressed to G. Moslehi; moslehi@cc.iut.ac.ir

Received 27 January 2014; Accepted 13 March 2014; Published 27 April 2014

Academic Editor: Bernard J. Geurts

This paper addresses a new performance measure for scheduling problems, entitled "biased tardiness penalty." We study the approximability of minimum biased tardiness on a single machine, provided that all the due dates are equal. Two heuristic algorithms are developed for this problem, and it is shown that one of them has a worst-case ratio bound of 2. Then, we propose a dynamic programming algorithm and use it to design an FPTAS. The FPTAS is generated by cleaning up some states in the dynamic programming algorithm, and it requires $O(n^3/\varepsilon)$ time.

1. Introduction

In this paper, we study a single machine scheduling problem of minimizing total biased tardiness about a common due date. Every job i ($1 \leq i \leq n$) has a processing time p_i, a weight tardiness factor w_i, and a base tardiness factor u_i. The machine is available at time zero and can process at most one job at a time. The jobs have a common due date d. The biased tardiness penalty of job i is defined as

$$Z_i = \begin{cases} 0, & \text{if } C_i \leq d \\ u_i + w_i (C_i - d), & \text{if } C_i > d, \end{cases} \quad (1)$$

where C_i is the completion time of job i. Figure 1 shows the biased tardiness penalty of job i based on its completion time in a sequence. The resulting problem is denoted by $1 \mid d_i = d \mid$ BTP, where BTP means "*biased tardiness penalty.*"

Biased tardiness penalty is a kind of performance measures that, according to our observations, has not been studied in the literature in spite of its wide use in practical situations. One of the most common applications of biased tardiness is in designing delivery contracts. In many of delivery contracts, once an order is delivered later than its due date, a fixed penalty must be paid, and when the delivery becomes tardier, the related penalty will increase as well. Many of the practical conditions support this assumption; for example, consider a company where just one day delay in receiving raw materials will break down its production line. In this case, the initial damage caused by late delivery is huge while if the delay increases, the additional damage is relatively small. Another application for biased tardiness penalty is in transportation systems where we must pay extra money for solo-transporting a piece of goods if it is not ready to be carried with other orders.

Problem $1 \parallel \sum w_i T_i$ is NP-hard in the strong sense if the tardiness weights are not all equal [1, 2] and is optimally solvable in pseudo-polynomial time for a fixed number of distinct due dates [3]. Cheng et al. [4] have shown that the schedule that minimizes $\max_j w_j T_j$ gives an $(n-1)$-approximation for this problem. Kolliopoulos and Steiner [3] design pseudo-polynomial algorithms for the case that there is only a fixed number of different due dates. They also develop an FPTAS if, in addition, the tardiness weights are bounded by a polynomial function of n. Karakostas et al. [5] consider the same problem, design a pseudo-polynomial algorithm, and apply a rounding scheme to obtain the desired approximation scheme.

In a special case of problem $1 \parallel \sum w_i T_i$ where the due date is common for all jobs, the resulting problem is

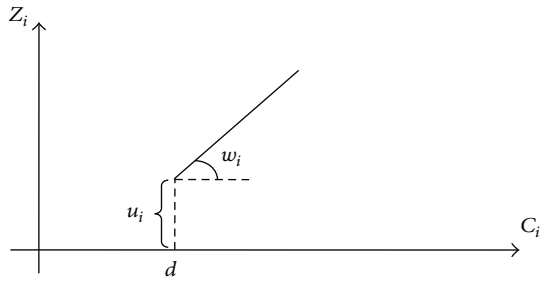

Figure 1: Biased tardiness penalty for job i.

proved to be NP-hard in the ordinary sense by Yuan [6], and Lawler and Moore [7] provide a pseudo-polynomial dynamic programming algorithm in $O(n^2d)$ time. Fathi and Nuttle [8] develop a 2-approximation algorithm that requires $O(n^2)$ time. Kellerer and Strusevich [9] propose an FPTAS of $O(n^6 \log W/\varepsilon^3)$ time complexity, where W is the sum of tardiness weights; later, Kacem [10] studies the same problem and develops another approach to obtain a more effective FPTAS in $O(n^2/\varepsilon)$ time.

If the tardiness weights are equal, problem $1 \parallel \sum T_i$ is NP-hard in the ordinary sense as proved by Du and Leung [11], and it is solvable by a pseudo-polynomial dynamic programming algorithm proposed by Lawler [1]. For this problem, Lawler [12] proposes a dynamic programming algorithm and converts it into an FPTAS of $O(n^7/\varepsilon)$ time complexity. Koulamas [13] provides a faster FPTAS running in $O(n^5 \log n + n^5/\varepsilon)$ time by applying an alternative rounding scheme in conjunction with implementing Kovalyov's [14] bound improvement procedure. Della Croce et al. [15] consider some popular constructive and decomposition heuristics and conclude that none of them guarantees a constant worst-case ratio bound. Kovalyov and Werner [16] study the approximability of this problem on parallel machines with a common due date.

To examine the complexity of problem $1 \mid d_i = d \mid$ BTP, we compare it with the problem of minimizing weighted tardiness on a single machine and common due date. If we set $u_i = 0$ for all jobs i, the considered problem transforms to problem $1 \mid d_i = d \mid \sum w_i T_i$ that is shown in [1, 2] to be NP-hard in the ordinary sense.

The remainder of this paper is organized as follows. In Sections 2 and 3, we describe two heuristic algorithms for problem $1 \mid d_i = d \mid$ BTP and prove their worst-case ratio bounds. Section 4 describes a dynamic programming algorithm that, in Section 5, we convert to an FPTAS using the technique of *structuring the execution of an algorithm*. Concluding remarks are given in Section 6.

2. SPT Algorithm

In this heuristic algorithm, jobs are sequenced according to a nondecreasing order of processing times, and, hence, it can be implemented in $O(n \log n)$ time.

Theorem 1. *Let $w_{\min}$ and $w_{\max}$ be, respectively, the smallest and largest weight tardiness factors, and let $u_{\min}$ and $u_{\max}$ be the smallest and largest base tardiness factors. Then, one has*

$$\frac{Z^{SPT}}{Z^*} \le \max\left\{\frac{u_{\max}}{u_{\min}}, \frac{w_{\max}}{w_{\min}}\right\}, \tag{2}$$

where Z^{SPT} is the penalty created by SPT algorithm and Z^* shows the optimal penalty for problem $1 \mid d_i = d \mid BTP$.

Proof. Consider two ordered sets $\{A_w\}$ and $\{A_u\}$ that include a nondecreasing order of the weight tardiness factors and base tardiness factors, respectively. Suppose that we create n dummy jobs by pairing processing times in SPT ordering and tardiness factors according to their reverse order in sets $\{A_w\}$ and $\{A_u\}$. It can be easily verified that the associated total penalty, called LB^*, is a lower bound on the total penalty of any sequence for real jobs.

Similarly, create another set of n dummy jobs by pairing processing times in SPT ordering but tardiness factors consistent with the order of sets $\{A_w\}$ and $\{A_u\}$. It can be easily tested that if we sequence these dummy jobs according to SPT ordering, the related total penalty, called $\mathrm{UB}^{\mathrm{SPT}}$, is an upper bound on Z^{SPT} for the real jobs. Let n_T be the number of tardy jobs under SPT ordering. Also, let $A_w^{[r]}$ and $A_u^{[r]}$ denote the rth job in sets $\{A_w\}$ and $\{A_u\}$, respectively. Also, let $C_{[r]}^{\mathrm{SPT}}$ denote the completion time of rth job in SPT ordering. Thus, we have

$$\begin{aligned}
\mathrm{UB}^{\mathrm{SPT}} &= A_w^{[n-n_T+1]}\left(C_{[n-n_T+1]}^{\mathrm{SPT}} - d\right) + \cdots + A_w^{[n]}\left(C_{[n]}^{\mathrm{SPT}} - d\right) \\
&\quad + A_u^{[n]} + \cdots + A_u^{[n-n_T+1]}, \\
\mathrm{LB}^* &= A_w^{[n_T]}\left(C_{[n-n_T+1]}^{\mathrm{SPT}} - d\right) + \cdots + A_w^{[1]}\left(C_{[n]}^{\mathrm{SPT}} - d\right) \\
&\quad + A_u^{[1]} + \cdots + A_u^{[n_T]}.
\end{aligned} \tag{3}$$

From (3) and the fact that the values are nondecreasingly ordered in sets $\{A_w\}$ and $\{A_u\}$, we get

$$\begin{aligned}
\mathrm{UB}^{\mathrm{SPT}} &\le w_{\max}\left[\left(C_{[n-n_T+1]}^{\mathrm{SPT}} - d\right) + \cdots + \left(C_{[n]}^{\mathrm{SPT}} - d\right)\right] \\
&\quad + u_{\max} n_T, \\
\mathrm{LB}^* &\ge w_{\min}\left[\left(C_{[n-n_T+1]}^{\mathrm{SPT}} - d\right) + \cdots + \left(C_{[n]}^{\mathrm{SPT}} - d\right)\right] \\
&\quad + u_{\min} n_T.
\end{aligned} \tag{4}$$

And if we signify the term $(C_{[n-n_T+1]}^{\mathrm{SPT}} - d) + \cdots + (C_{[n]}^{\mathrm{SPT}} - d)$ by T^{SPT}, then

$$\begin{aligned}
\frac{\mathrm{UB}^{\mathrm{SPT}}}{\mathrm{LB}^*} &\le \frac{w_{\max} T^{\mathrm{SPT}} + u_{\max} n_T}{w_{\min} T^{\mathrm{SPT}} + u_{\min} n_T} \\
&\le \max\left\{\frac{w_{\max}}{w_{\min}}, \frac{u_{\max}}{u_{\min}}\right\},
\end{aligned} \tag{5}$$

which completes the proof. □

Table 1: Jobs' parameters in Example 2.

Job i	p_i	w_i	u_i
1	100	100	20
2	10	20	4

The following example illustrates that the worst-case ratio bound obtained by SPT ordering is tight for problem1 | $d_i = d$ | BTP.

Example 2. Suppose that we have two jobs with parameters given in Table 1 and a common due date $d = 100$.

SPT ordering generates the sequence (2-1) with total penalty equal to 1020, while the optimal sequence for this example is (1-2) with the total penalty of 204. So,

$$\frac{Z^{\mathrm{SPT}}}{Z^*} = \frac{1020}{204} = 5,$$
$$\max\left\{\frac{u_{\max}}{u_{\min}}, \frac{w_{\max}}{w_{\min}}\right\} = \max\left\{\frac{20}{4}, \frac{100}{20}\right\} = 5. \tag{6}$$

3. Algorithm MPR (Minimum Penalty Rate)

In this section, we present another heuristic algorithm for problem 1 | $d_i = d$ | BTP and show that the worst-case ratio bound of this algorithm is 2. Let $G = (g_{[1]}, g_{[2]}, \ldots, g_{[n]})$ denote the sequence generated by MPR algorithm, where $g_{[i]}$ represents the ith job within this sequence. This algorithm requires at most n iterations and during its rth iteration the $(n - r + 1)$th element of G, $g_{[n-r+1]}$, will be determined. Also, if there are some unscheduled jobs at each iteration filling the whole remaining period through due date (remaining tardiness period), we will choose the one making the minimum penalty and save the related sequence, $\widehat{G}$, beside the main sequence G. Finally, the algorithm sorts tardy jobs in the main and secondary sequences according to WSPT (weighted shortest processing time) ordering and returns the sequence with smaller penalty.

Let $Z_{g[r,n]}$ denote sum penalties related to the jobs from rth position through the last job in sequence G. So, the algorithm is as follows.

Algorithm MPR.

(1) Let $U = \{1, 2, \ldots, n\}$ be the set of unscheduled jobs and let r be the counter index of positions in the sequence. Set $C_{[n]} = P_{\mathrm{sum}} = \sum_{i=1}^{n} p_i$ and $r = n$.

(2) Define $\widehat{U} = \{i \in U \mid p_i \geq C_{[n]} - d\}$. If $\widehat{U}$ is empty, then $\widehat{Z}_{\mathrm{best}} = \infty$; else, consider the job with the minimum penalty in $\widehat{U}$ as job $\hat{i}$ that is calculated by $\widehat{Z}_{\mathrm{best}} = u_{\hat{i}} + w_{\hat{i}}(C_{[n]} - d) = \min_{i\in\widehat{U}}\{u_i + w_i(C_{[n]} - d)\}$ and also $\hat{g}_{[n]} = \hat{i}$.

Table 2: Jobs' parameters in Example 3.

Job i	p_i	w_i	u_i
1	7	8	18
2	6	4	16
3	4	2	24
4	6	5	17

(3) For all jobs $i \in U$, calculate θ_i values by (7) and select a job k such that $\theta_k = \min_{i\in U}\{\theta_i\}$. If there is a tie, select the job with the smaller processing time;

$$\theta_i = \frac{u_i + w_i\left(C_{[r]} - d\right)}{\min\left(p_i, C_{[r]} - d\right)}. \tag{7}$$

(4) Set $U = U/\{k\}$, $g_{[r]} = k$, and $C_{[r-1]} = C_{[r]} - p_k$.

(5) If $\widehat{U} = \{i \in U \mid p_i \geq C_{[r-1]} - d\}$ is not empty, then calculate the value $\widehat{Z} = u_{\hat{i}} + w_{\hat{i}}(C_{[r-1]} - d) = \min_{i\in\widehat{U}}\{u_i + w_i(C_{[r-1]} - d)\}$. If $\widehat{Z} + Z_{g[r,n]} < \widehat{Z}_{\mathrm{best}}$, then set $\widehat{Z}_{\mathrm{best}} = \widehat{Z} + Z_{g[r,n]}$, $\hat{g}_{[r-1]} = \hat{i}$, and $\hat{g}_{[i]} = g_{[i]}$ $\forall i = r, \ldots, n$.

(6) If $C_{[r-1]} > d$, then $r = r - 1$ and go back to step 3.

(7) Schedule the remaining jobs at the beginning of sequences G and $\widehat{G}$.

(8) Sort the tardy jobs in sequences G and $\widehat{G}$ according to the nonincreasing order of w_i/p_i ratios.

(9) If $Z^G \leq Z^{\widehat{G}}$, then return the sequence $G = (g_{[1]}, g_{[2]}, \ldots, g_{[n]})$ with $Z_{g[1,n]}$ penalty, and, else, return the sequence $\widehat{G} = (\hat{g}_{[1]}, \hat{g}_{[2]}, \ldots, \hat{g}_{[n]})$ with penalty equal to $Z_{\hat{g}[1,n]}$.

It can be easily seen that MPR algorithm runs in $O(n^2)$ time. The following example illustrates the implementation of MPR algorithm on a simple problem.

Example 3. Suppose a problem with four jobs and a common due date $d = 10$. Table 2 shows the jobs' parameters.

At first, $C_{[4]} = 23$, $\widehat{Z}_{\mathrm{best}} = \infty$, and $\widehat{U}$ is empty. At the first iteration of running the algorithm ($r = 4$), we have $\theta_1 = 17.43$, $\theta_2 = 11.33$, $\theta_3 = 12.5$, and $\theta_4 = 13.67$, where job 2 has the minimum value, and, hence, it is sequenced at the last position of G. Also, $C_{[3]} = 17$, $\widehat{U} = \{1\}$, and $\widehat{Z} = 74$, and considering $Z_{g[4,4]} = 68$, we get $\widehat{Z}_{\mathrm{best}} = 142$. So, $\hat{g}_{[4]} = 2$ and $\hat{g}_{[3]} = 1$.

At the second iteration ($r = 3$), we have $\theta_1 = 10.57$, $\theta_3 = 9.5$, and $\theta_4 = 8.67$, where job 4 has the minimum value, and, hence, $g_{[3]} = 4$. The algorithm calculates $C_{[2]} = 11$, $\widehat{U} = \{1, 3\}$, and $\widehat{Z} = 26$, and considering $\widehat{Z} + Z_{g[3,4]} = 146 > \widehat{Z}_{\mathrm{best}}$, the value of $\widehat{Z}_{\mathrm{best}}$ will remain unchanged. At the last iteration ($r = 2$), we have $\theta_1 = 26$ and $\theta_3 = 26$, and the algorithm sequences job 3 at the second position of sequence G. Also, $C_{[1]} = 7$ and $\widehat{U}$ is empty.

After arbitrary scheduling the remaining jobs at the beginning of sequences G and $\widehat{G}$, we get $G = (1, 3, 4, 2)$

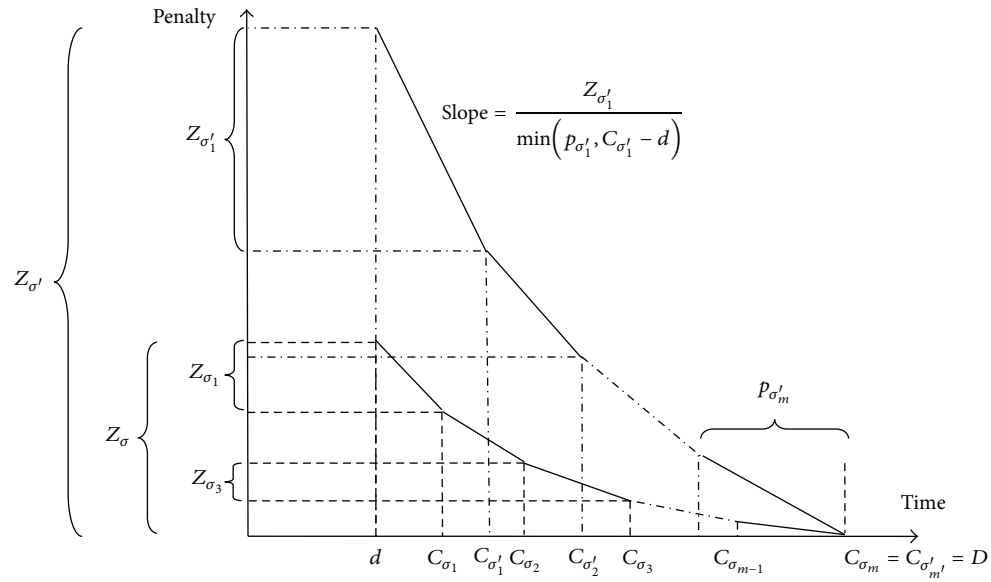

Figure 2: Penalties related to sequences σ and σ'.

and $G' = (3, 4, 1, 2)$. These sequences are modified to $G = (1, 4, 2, 3)$ and $G' = (3, 4, 1, 2)$ after implementing step 8. Also, $Z^G = 134$ and $Z^{\widehat{G}} = 142$, which forces the algorithm to select sequence G as the final output.

Here, we present two theorems about problem $1 \mid d_i = d \mid$ BTP which are used for proving the worst-case ratio bound of MPR algorithm.

Theorem 4. *Consider a problem* $1 \mid d_i = d \mid$ *BTP. Define two sequences* $\sigma = (\sigma_1, \ldots, \sigma_m)$ *and* $\sigma' = (\sigma'_1, \ldots, \sigma'_{m'})$ *on a common time interval, where the relation* $Z_i / \min(p_i, C_i - d) \leq Z_j / \min(p_j, C_j - d)$ *holds for all jobs* i *in* σ *and jobs* j *in* σ'. *Then,* $Z^\sigma \leq Z^{\sigma'}$.

Proof. Consider two sequences σ and σ' with tardiness penalties shown in Figure 2. Suppose that $\theta^\sigma(t)$ and $\theta^{\sigma'}(t)$ denote the slope of the functions related to sequences σ and σ', respectively; then, according to the theorem's assumption, we have $\theta^\sigma(t) \leq \theta^{\sigma'}(t)$ for all $t \in [d, D]$. It is obvious that for all $t \in [d, D]$ (especially for point d) the function related to sequence σ falls under the function related to sequence σ' and, hence, $\sum_{i=1}^{m} Z_{\sigma_i} \leq \sum_{j=1}^{m'} Z_{\sigma'_j}$. A similar conclusion can be made for the case where $C_{\sigma_m} < C_{\sigma'_{m'}}$ holds. □

Theorem 5. *In any optimal sequence for problem* $1 \mid d_i = d \mid$ *BTP, the tardy jobs with start time greater than or equal to* d *must be sequenced in WSPT ordering. This means that*

$$\forall i, j \in Z^* : d \leq C_i - p_i < C_j - p_j \Longrightarrow \frac{p_i}{w_i} \leq \frac{p_j}{w_j}. \quad (8)$$

Proof. The proof is easily done by swapping each pair of the adjacent tardy jobs. □

Table 3: Jobs' parameters in Example 7.

Job i	p_i	w_i	u_i
1	n	$n + 1/n$	n
2 to $n + 1$	1	$1 + 1/n$	1

Theorem 6. *Algorithm MPR gives a 2-approximation for problem* $1 \mid d_i = d \mid$ *BTP.*

Proof. See the appendix. □

The following example illustrates that the worst-case ratio bound obtained by MPR is tight for problem $1 \mid d_i = d \mid$ BTP.

Example 7. Suppose that we have $n + 1$ jobs with parameters given in Table 3 and a common due date $d = n$.

Algorithm MPR gives the sequence $(2, 3, \ldots, n, 1)$ with $n^2 + n + 1$ total penalty, while the optimal penalty is $(1/2)n^2 + 2n + 1/2$ related to the sequence $(1, 2, \ldots, n)$. Thus,

$$\frac{Z^G}{Z^*} = \frac{n^2 + n + 1}{(1/2)\, n^2 + 2n + 1/2} \Longrightarrow \lim_{n \to \infty} \frac{Z^G}{Z^*} = 2. \quad (9)$$

4. Dynamic Programming (DP) Algorithm

Without loss of generality, we consider that jobs are indexed according to the WLPT ordering. For problem $1 \mid d_i = d \mid$ BTP, an optimal schedule belongs to the class of schedules in which the early jobs are processed starting at time zero and are followed by a straddling job, called job α, that starts no later than time d and is completed after time d; in turn, the straddling job is followed by the block of tardy jobs. The early jobs can be processed in any order, while, according to Theorem 5, tardy jobs that start at or after the due date must be processed according to WSPT numbering. Let us introduce the following notations.

(1) For each $\alpha = \{1, 2, \ldots, n\}$
  (1.1) Set $\nu_0^{(\alpha)} = \{[0, 0]\}$
  (1.2) For each $k = \{1, 2, \ldots, \alpha - 1, \alpha + 1, \ldots, n\}$
    (i) Consider every state $[t, f]$ in $\nu_{k-1}^{(\alpha)}$
      (a) If $t + p_k \leq d$ then add $[t + p_k, f]$ to $\nu_k^{(\alpha)}$
      (b) If $P_{\text{sum}} - \sum_{i=1}^{k} p_i + t > d$ then add $\left[t, f + u_k + w_k\left(P_{\text{sum}} - \sum_{i=1}^{k-1} p_i + t - d\right)\right]$ to $\nu_k^{(\alpha)}$
    (ii) Delete the state space $\nu_{k-1}^{(\alpha)}$
  (1.3) Set $Z_\alpha^* = \min_{[t,f] \in \nu_n^{(\alpha)}} \{f + u_\alpha + w_\alpha (t + p_\alpha - d)\}$
(2) Calculate the optimal value from $Z^* = \min_{\alpha | Z_\alpha^* \neq \emptyset} Z_\alpha^*$

Algorithm 1: DP algorithm.

$[t, f]$ is a state in the state space, where t denotes the total processing time of early jobs and f is the total penalty of state.

$\nu_k^{(\alpha)}$ is a set of states for the first k jobs, except for job α.

Z_α^* is the minimum penalty value for problem $1 \mid d_i = d \mid \text{BTP}$ with a fixed straddling job α.

P_{sum} is the total processing time of all jobs.

This DP algorithm schedules early jobs starting from time zero and the tardy jobs so that they become complete exactly at time P_{sum}. According to this, the algorithm can be described as in Algorithm 1.

Let UB^* be an upper bound on the optimal penalty, and since $f \leq \text{UB}^*$ and $t \leq d$, we can restrict the number of states $\nu_k^{(\alpha)}$ by $d \times \text{UB}^*$. The complexity of substep 1.2 of the DP algorithm is proportional to $\sum_{k=1}^{n} |\nu_k^{(\alpha)}|$ that leads to $O(n \cdot d \cdot \text{UB}^*)$ time. However, this complexity can be reduced to $O(n \cdot d)$ by selecting a state $[t, f]$ with the smallest value of f at each iteration k and for every t. Similarly, we can get the complexity of substep 1.3 as $O(d)$, and so, the complexity of step 1 is $O(n^2 \cdot d)$. Step 2 requires $O(n)$ time, and the final complexity of DP algorithm will be calculated as $O(n^2 \cdot d)$. The following example illustrates the details of DP algorithm.

Example 8. Consider an instance of problem $1 \mid d_i = d \mid \text{BTP}$ with 3 jobs. The parameters of the jobs are given in Table 4, and the common due date is given as $d = 6$ in this example.

Table 5 shows the states generated in each states space regarding the selected straddling jobs as well as subsequences coupled with these states. The optimal value is $Z^* = \min\{57, 56, 58\} = 56$ related to sequence $(3, 2, 1)$ which is obtained by inserting straddling job $\alpha = 2$ into the subsequence $(3, \ , 1)$.

Table 4: Jobs' parameters in Example 8.

Job i	p_i	w_i	u_i
1	5	2	12
2	7	3	14
3	3	4	9

5. FPTAS Algorithm

One of the standard approaches to generate an FPTAS is the technique of *structuring the execution of an algorithm*. Here, the main idea is to take the exact but slow DP algorithm described in Section 4 and to interact with it while it is working. If the algorithm generates a lot of auxiliary states during its execution, then we may remove some of these states and clean up the algorithm's memory. This method was introduced by Ibarra and Kim [17] for solving the knapsack problem, and in the recent years numerous scheduling problems have applied such an approach (see [18–22]). First, let us introduce the following notations.

ε is the error bound of FPTAS algorithm.

$\nu_k^{(\alpha)\#}$ is a set of states generated by FPTAS for the first k jobs, except for job α.

$Z_\alpha^\#$ is the minimum penalty generated by FPTAS for problem $1 \mid d_i = d \mid \text{BTP}$ with a fixed straddling job α.

$Z^\#$ is the minimum penalty generated by FPTAS for problem $1 \mid d_i = d \mid \text{BTP}$.

Consider the penalty of algorithm MPR, called Z_H, as an upper bound for the problem. To reduce the number of states in each iteration, we split the feasible interval $[0, Z_H]$ related to the second coordinate of state $[t, f]$ into $L+1$ equal subintervals $I_m = [(m-1)\Delta, m\Delta)$ $1 \leq m \leq L+1$ of length Δ. For each of the resulting subintervals I_m, we keep at most one state with the smallest value t. Given an arbitrary $\varepsilon > 0$, define

$$\text{LB} = \frac{Z_H}{2}, \qquad L = \left\lceil \frac{2n}{\varepsilon} \right\rceil, \qquad \Delta = \frac{Z_H}{L}. \tag{10}$$

The FPTAS algorithm works on the reduced state space $\nu_k^{(\alpha)\#}$ instead of $\nu_k^{(\alpha)}$ and can be described as in Algorithm 2.

5.1. Worst-Case Analysis of the FPTAS Algorithm. The worst-case analysis is based on comparing the execution of DP and FPTAS algorithms. First, a lemma is provided that will be used to prove the worst-case ratio bound of FPTAS.

(1) For each $\alpha = \{1, 2, \ldots, n\}$
  (1.1) Set $\nu_0^{(\alpha)\#} = \{[0, 0]\}$
  (1.2) For each $k = \{1, 2, \ldots, \alpha - 1, \alpha + 1, \ldots, n\}$
    (i) Consider every state $[t, f]$ in $\nu_{k-1}^{(\alpha)\#}$
      (a) If $t + p_k \le d$ then add $[t + p_k, f]$ to $\nu_k^{(\alpha)\#}$
      (b) If $P_{\text{sum}} - \sum_{i=1}^{k} p_i + t > d$ then add $\left[t, f + u_k + w_k\left(P_{\text{sum}} - \sum_{i=1}^{k-1} p_i + t - d\right)\right]$ to $\nu_k^{(\alpha)\#}$
    (ii) Delete the state space $\nu_{k-1}^{(\alpha)\#}$
    (iii) Let $[t, f]_m$ be a state in $\nu_k^{(\alpha)\#}$ such that $f \in I_m$ with the smallest t (break ties by choosing the state of the smallest f). Set $\nu_k^{(\alpha)\#} = \{[t, f]_m \mid 1 \le m \le L + 1\}$
  (1.3) Set $Z_\alpha^\# = \min_{[t,f] \in \nu_n^{(\alpha)\#}} \{f + u_\alpha + w_\alpha (t + p_\alpha - d)\}$
(2) Calculate the final solution of FPTAS from $Z^\# = \min_{\alpha | Z_\alpha^\# \neq \emptyset} Z_\alpha^\#$

Algorithm 2: FPTAS algorithm.

Table 5: Procedure of DP algorithm in Example 8.

	$\nu_0^{(\alpha)}$		$\nu_1^{(\alpha)}$		$\nu_2^{(\alpha)}$		Z_α^*
	State	Sequence	State	Sequence	State	Sequence	
$\alpha = 1$	[0, 0]	(, ,)	[0, 41]	(, , 2)	[3, 41]	(3, , 2)	41 + 16 = 57
$\alpha = 2$	[0, 0]	(, ,)	[5, 0] [0, 30]	(1, ,) (, , 1)	[5, 45] [3, 30] [0, 55]	(1, , 3) (3, , 1) (, 3, 1)	$\min \begin{Bmatrix} 45 + 32 \\ 30 + 26 \\ 55 + 17 \end{Bmatrix} = 56$
$\alpha = 3$	[0, 0]	(, ,)	[5, 0] [0, 30]	(1, ,) (, , 1)	[5, 41]	(1, , 2)	41 + 17 = 58

Lemma 9. *Let $[t, f]$ be an arbitrary state in $\nu_k^{(\alpha)}$. The FPTAS algorithm generates at least one state $[t^\#, f^\#]$ in $\nu_k^{(\alpha)\#}$ such that $t^\# \le t$ and $f^\# \le f + k\Delta$.*

Proof. The proof is done by induction on k. For $k = 0$, obviously we have $\nu_0^{(\alpha)\#} = \nu_0^{(\alpha)}$. Suppose that the lemma is valid up to $k - 1$ and we want to show its validity for iteration k. Let $[t, f]$ be a state in $\nu_k^{(\alpha)}$ generated by the DP algorithm from a feasible state $[t', f']$ at iteration $k - 1$. Here, two cases can be distinguished. In the first case $[t, f] = [t' + p_k, f']$ and in the second case $[t, f] = [t', f' + u_k + w_k \max\{0, P_{\text{sum}} - \sum_{i=1}^{k-1} p_i + t' - d\}]$ holds. We prove the statement for iteration k in these two cases.

Case 1 ($[t, f] = [t' + p_k, f']$). Since $[t', f'] \in \nu_{k-1}^{(\alpha)}$, there exists a state $[t'^\#, f'^\#] \in \nu_{k-1}^{(\alpha)\#}$ such that $t'^\# \le t'$ and $f'^\# \le f' + (k - 1)\Delta$. Therefore, the FPTAS algorithm generates the state $[t'^\# + p_k, f'^\#]$ that may be eliminated when cleaning up the state subset. Let $[\lambda, \mu]$ be the remaining state in $\nu_k^{(\alpha)\#}$ that is in the same interval as $[t'^\# + p_k, f'^\#]$. Thus, we drive that

$$\begin{aligned} \lambda &\le t'^\# + p_k \le t' + p_k = t, \\ \mu &\le f'^\# + \Delta \le f' + (k - 1)\Delta + \Delta = f + k\Delta. \end{aligned} \tag{11}$$

Consequently, the lemma holds for iteration k in this case.

Case 2 ($[t, f] = [t', f' + u_k + w_k \max\{0, P_{\text{sum}} - \sum_{i=1}^{k-1} p_i + t' - d\}]$). Since $[t', f'] \in \nu_{k-1}^{(\alpha)}$, there exists a state $[t'^\#, f'^\#] \in \nu_{k-1}^{(\alpha)\#}$ such that $t'^\# \le t'$ and $f'^\# \le f' + (k - 1)$. Therefore, the FPTAS algorithm generates the state $[t'^\#, f'^\# + u_k + w_k \max\{0, P_{\text{sum}} - \sum_{i=1}^{k-1} p_i + t'^\# - d\}]$ at iteration k that may be eliminated during the cleaning up procedure. Let $[\lambda', \mu']$ be the remaining state in $\nu_k^{(\alpha)\#}$ that is in the same interval as $[t'^\#, f'^\# + u_k + w_k \max\{0, P_{\text{sum}} - \sum_{i=1}^{k-1} p_i + t'^\# - d\}]$. So, we have

$$\begin{aligned} \lambda' &\le t'^\# \le t' = t, \\ \mu' &\le f'^\# + u_k + w_k \max\left\{0, P_{\text{sum}} - \sum_{i=1}^{k-1} p_i + t'^\# - d\right\} + \Delta \\ &\le f' + (k - 1)\Delta + u_k \\ &\quad + w_k \max\left\{0, P_{\text{sum}} - \sum_{i=1}^{k-1} p_i + t'^\# - d\right\} + \Delta \\ &\le f' + (k - 1)\Delta + u_k \\ &\quad + w_k \max\left\{0, P_{\text{sum}} - \sum_{i=1}^{k-1} p_i + t' - d\right\} + \Delta \\ &= f + k\Delta. \end{aligned} \tag{12}$$

Thus, the lemma is proved for iteration k in this case, too. □

Theorem 10. *Given an arbitrary $\varepsilon > 0$, the FPTAS algorithm outputs a sequence with $Z^{\#}$ penalty such that $Z^{\#} \leq (1+\varepsilon)Z^{*}$.*

Proof. There exists a straddling job, called α^{*}, in the optimal sequence for problem $1 \mid d_i = d \mid \text{BTP}$. Since the FPTAS algorithm checks all jobs as straddling, then obviously job α^{*} will be selected in one of its iterations.

By definition, the optimal sequence can be related to a state $[t^{*}_{\alpha^{*}}, f^{*}_{\alpha^{*}}]$ in $\nu_n^{(\alpha^{*})}$. According to Lemma 9, the FPTAS algorithm generates a state $[t^{\#}_{\alpha *}, f^{\#}_{\alpha *}]$ in $\nu_n^{(\alpha^{*})\#}$ such that $t^{\#}_{\alpha^{*}} \leq t^{*}_{\alpha^{*}}$ and

$$\begin{aligned} f^{\#}_{\alpha^{*}} &\leq f^{*}_{\alpha^{*}} + n\Delta \\ &= f^{*}_{\alpha^{*}} + n\frac{Z_H}{L} = f^{*}_{\alpha^{*}} + n\frac{Z_H}{\lceil 2n/\varepsilon \rceil} \\ &\leq f^{*}_{\alpha^{*}} + n\frac{Z_H}{2n/\varepsilon} = f^{*}_{\alpha^{*}} + \varepsilon \cdot \text{LB}. \end{aligned} \tag{13}$$

It is clear that $Z^{*} \geq \text{LB}$. Let $T^{*}_{\alpha^{*}}$ and $T^{\#}_{\alpha^{*}}$ denote the tardiness of job α^{*} in the optimal and FPTAS solutions, respectively. From $f^{*}_{\alpha^{*}} + T^{*}_{\alpha^{*}} = Z^{*}$, we have

$$\begin{aligned} t^{\#}_{\alpha^{*}} \leq t^{*}_{\alpha^{*}} &\Longrightarrow T^{\#}_{\alpha^{*}} \leq T^{*}_{\alpha^{*}} \\ &\Longrightarrow f^{\#}_{\alpha^{*}} + T^{\#}_{\alpha^{*}} \leq f^{*}_{\alpha^{*}} + \varepsilon \cdot \text{LB} + T^{*}_{\alpha^{*}} \\ &\Longrightarrow Z^{\#} \leq (1+\varepsilon)\, Z^{*}. \end{aligned} \tag{14}$$

This will complete the proof. □

5.2. Complexity of the FPTAS Algorithm. MPR algorithm runs in $O(n^2)$ time as the initial phase of FPTAS. The state space $\nu_k^{(\alpha)\#}$ $(k = \{1, 2, \ldots, n\})$ is generated at each iteration of substep 1.2 and in $O(n^2)$ time. Since $|\nu_k^{(\alpha)\#}| \leq L$, we have

$$\sum_{k=1}^{n} \left|\nu_k^{(\alpha\#)}\right| \leq nL = n\left\lceil \frac{2n}{\varepsilon} \right\rceil \leq n\left(\frac{2n}{\varepsilon} + 1\right). \tag{15}$$

According to this, substep 1.2 requires $O(n^2/\varepsilon)$ time. Noting that step 1 iterates n times for every selection of α values, the complexity of this step is $O(n^3/\varepsilon)$. Finally, step 2 requires $O(n)$ time, and the final complexity of the FPTAS algorithm is computed as $O(n^3/\varepsilon)$.

6. Conclusion

In this paper, we presented a new performance measure for scheduling problems, called *biased tardiness penalty*. According to this performance measure, two kinds of penalties are assigned to each tardy job: one fixed penalty and the other that linearly increases by the increase in tardiness value. Two approximation algorithms were designed with the polynomial running times. The first approximation algorithm, SPT, gives a worst-case ratio bound linking to size of instances, while the second approximation algorithm, MPR, has a constant worst-case ratio bound of 2. Next, we developed a dynamic programming algorithm and converted it to an FPTAS using the method of *structuring the execution of an algorithm*. The resulting FPTAS runs in $O(n^3/\varepsilon)$ time.

Appendix

Proof of Theorem 6

We consider two main cases for the sequence of tardy jobs and prove the worst-case ratio bound in both cases. Recall that Z^{G} and Z^{*} denote the penalties from MPR and optimal sequences, respectively. Let p_σ indicate the sum processing times of jobs in a sequence σ and let C_i^{σ} indicate completion time of job i in a sequence σ. Also, $Z^{\sigma}_{(t)}$ shows the penalty of sequence σ if it ends at time t.

Case 1. The first tardy job in MPR sequence before the sorting phase (step 8) is also tardy in optimal sequence.

Algorithm MPR schedules jobs from the end of sequence to the beginning, while some of the selected tardy jobs are also tardy in the optimal sequence and some others are not. According to this, we can show the sequence of tardy jobs before the sorting phase (step 8) as in Figure 3. In this figure, sets H' contain the jobs that only are tardy in the heuristic sequence and sets B' contain the jobs that are tardy in both heuristic and optimal sequences. Without loss of generality, suppose that each set B'_i contains a single job because sets H' can be empty. Let sequence G' begin with a job in B'_k from tardy jobs in optimal sequence. Put other tardy jobs in optimal sequence into set B'_{k+1}. Here, two subcases are identified.

Subcase 1.1 (B'_{k+1} is empty). In this case, sets B'_1 to B'_k contain all tardy jobs in the optimal sequence. Figure 4 shows the sequence of tardy jobs after execution of the sorting phase (step 8). From Theorem 5, the jobs included in sets B have the same order in both sequences G and the optimal sequence. So, we have

$$\begin{aligned} Z^{G} \leq Z^{*} &- \left[w_{B_1} p_{H_1} + w_{B_2} \sum_{i=1}^{2} p_{H_i} + \cdots + w_{B_k} \sum_{i=1}^{k} p_{H_i} \right] \\ &+ \sum_{j=1}^{k} \left[\sum_{g_{[i]} \in H_j} \left(u_{g_{[i]}} + w_{g_{[i]}} \left(\sum_{r=j}^{k} p_{B_r} + \sum_{r=j+1}^{k} p_{H_r} \right. \right. \right. \\ &\left. \left. \left. + \sum_{r=g_{[i]}+1}^{g_{[n_j]}} p_r \right) \right) \right], \end{aligned} \tag{A.1}$$

where the second term shows the decrease in penalty values of B_1 to B_k in sequence G compared with the related penalties in optimal sequence. The second term indicates penalty related to sets H_1 to H_k in the heuristic sequence.

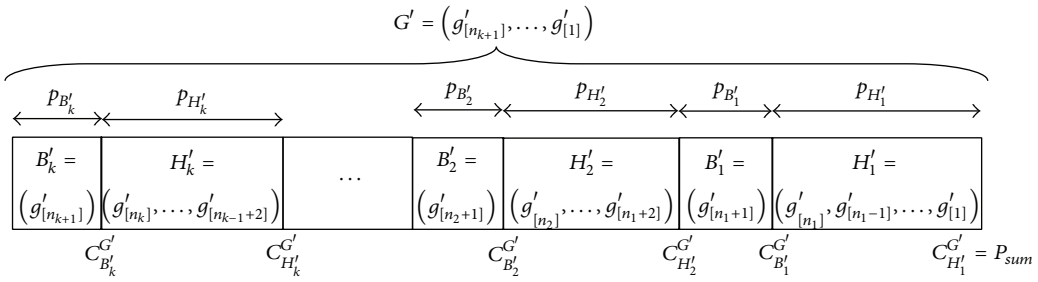

Figure 3: Sequence of tardy jobs in Case 1 before sorting phase.

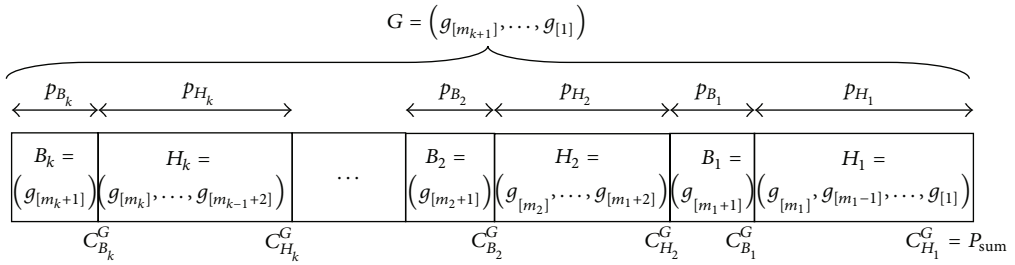

Figure 4: Sequence of tardy jobs in Case 1 after sorting phase.

Regarding the fact that all the jobs in sequence G which are included in some sets H_j come after the job in B_j, we get

$$\begin{aligned}
\frac{w_i}{p_i} \le \frac{w_{B_j}}{p_{B_j}} &\Longrightarrow w_i \le \frac{w_{B_j}}{p_{B_j}} \cdot p_i \\
&\Longrightarrow \sum_{i \in H_j} w_i \le \frac{w_{B_j}}{p_{B_j}} \cdot \sum_{i \in H_j} p_i \\
&\Longrightarrow p_{B_j} \cdot \sum_{i \in H_j} w_i \le w_{B_j} \cdot \sum_{i \in H_j} p_i \\
&\qquad \forall i \in H_j.
\end{aligned} \tag{A.2}$$

From (A.1) and (A.2),

$$\begin{aligned}
Z^G &\le Z^* + \sum_{j=1}^{k} \left[\sum_{g_{[i]} \in H_j} \left(u_{g_{[i]}} + \sum_{g_{[i]} \in H_j} w_{g_{[i]}} \left(\sum_{r=j+1}^{k} p_{H_r} + \sum_{r=g_{[i]}+1}^{g_{[m_j]}} p_r \right) \right) \right],
\end{aligned} \tag{A.3}$$

$$Z^G \le Z^* + Z^{[H_k, \ldots, H_1]}_{\left(d + \sum_{i=1}^{k} p_{H_i}\right)}. \tag{A.4}$$

Before the sorting phase (step 8), jobs are sequenced according to the nondecreasing order of θ_i's, and MPR algorithm in step 8 considers all the jobs filling the whole tardiness period in each iteration; thus,

$$\left.\begin{aligned}
\sum_{i=1}^{k} p_{H'_i} + \sum_{i=1}^{k-1} p_{B'_i} < P_{\text{sum}} - d \\
\sum_{i=1}^{k} p_{B'_i} \ge P_{\text{sum}} - d
\end{aligned}\right\} \Longrightarrow \sum_{i=1}^{k} p_{H'_i} < p_{B'_k}. \tag{A.5}$$

Now, from Theorem 4 and (A.5) and the fact that the job in B'_k is selected at the last iteration of algorithm, we conclude that

$$Z^{[H'_k, \ldots, H'_1]}_{\left(d + \sum_{i=1}^{k} p_{H'_i}\right)} \le Z^{[B'_k]}_{\left(d + \sum_{i=1}^{k} p_{H'_i}\right)}. \tag{A.6}$$

From Theorem 5, we get

$$Z^{[H_k, \ldots, H_1]}_{\left(d + \sum_{i=1}^{k} p_{H_i}\right)} \le Z^{[H'_k, \ldots, H'_1]}_{\left(d + \sum_{i=1}^{k} p_{H'_i}\right)}. \tag{A.7}$$

Also, by (A.4), (A.6), and (A.7), it follows that

$$\begin{aligned}
Z^G &\le Z^* + Z^{[H_k, \ldots, H_1]}_{\left(d + \sum_{i=1}^{k} p_{H_i}\right)} \\
&\le Z^* + Z^{[H'_k, \ldots, H'_1]}_{\left(d + \sum_{i=1}^{k} p_{H'_i}\right)} \\
&\le Z^* + Z^{[B'_k]}_{\left(d + \sum_{i=1}^{k} p_{H'_i}\right)}.
\end{aligned} \tag{A.8}$$

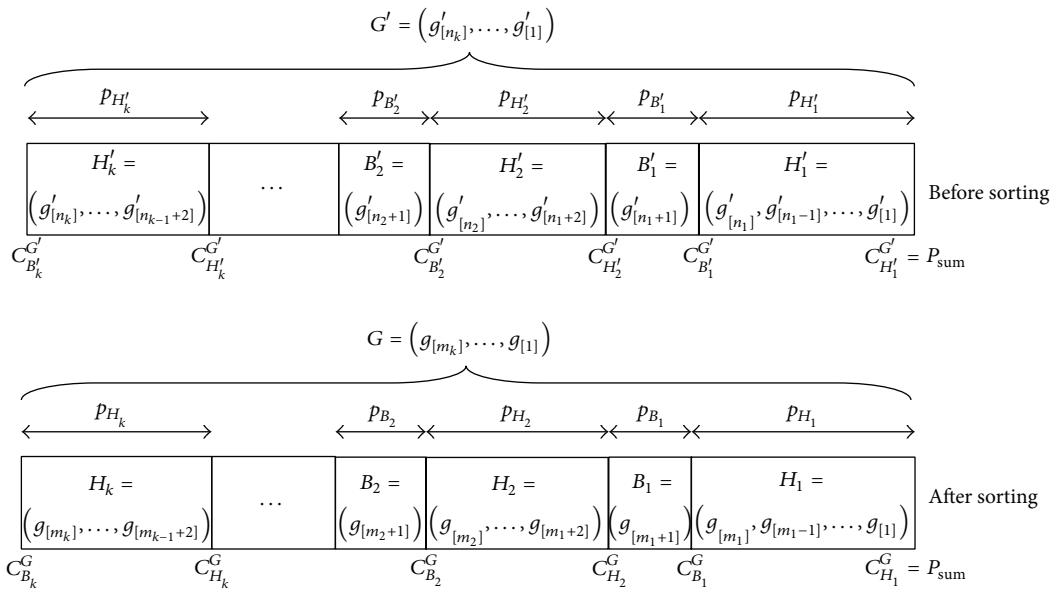

FIGURE 5: Sequence of tardy jobs in Case 2.

Finally, noting that $C^*_{B'_k} \geq P_{\text{sum}} - \sum_{i=1}^{k-1} p_{B'_i}$ and $d + \sum_{i=1}^{k} p_{H'_i} + \sum_{i=1}^{k-1} p_{B'_i} \leq P_{\text{sum}}$ we conclude that $d + \sum_{i=1}^{k} p_{H'_i} \leq C^*_{B'_k}$ and (A.8) leads to the proof of $Z^G \leq 2Z^*$ in this case.

Subcase 1.2 (B'_{k+1} is not empty). Similar to Subcase 1.1, we can show that $Z^G \leq Z^* + Z^{[H_k,\ldots,H_1]}_{(d+\sum_{i=1}^{k} p_{H_i})}$. Substitute the job in B'_k by two dummy jobs, a tardy job having the same tardiness factors as B'_k and a processing time $C^{G'}_{B'_k} - d$ and an early job having tardiness factors equal to zero and a processing time $p_{B'_k} - (C^{G'}_{B'_k} - d)$. This substitution will not affect the generality of the proof because the penalty of sequence G' remains unchanged under this substitution, while the optimal penalty cannot increase. So,

$$\left.\begin{aligned} \sum_{i=1}^{k} p_{H'_i} + \sum_{i=1}^{k} p_{B'_i} &= P_{\text{sum}} - d \\ \sum_{i=1}^{k+1} p_{B'_i} &\geq P_{\text{sum}} - d \end{aligned}\right\} \Longrightarrow \sum_{i=1}^{k} p_{H'_i} < p_{B'_{k+1}}. \quad \text{(A.9)}$$

From Theorem 4 and (A.9) and the fact that the heuristic algorithm has not selected any job in B'_{k+1}, it follows that

$$Z^{[H'_k,\ldots,H'_1]}_{\left(d+\sum_{i=1}^{k} p_{H'_i}\right)} \leq Z^{[B'_{k+1}]}_{\left(d+\sum_{i=1}^{k} p_{H'_i}\right)}. \quad \text{(A.10)}$$

According to $Z^{[H_k,\ldots,H_1]}_{(d+\sum_{i=1}^{k} p_{H_i})} \leq Z^{[H'_k,\ldots,H'_1]}_{(d+\sum_{i=1}^{k} p_{H'_i})}$, we conclude that $Z^G \leq Z^* + Z^{[B'_{k+1}]}_{(d+\sum_{i=1}^{k} p_{H'_i})}$. According to $C^*_{B'_{k+1}} \geq P_{\text{sum}} - \sum_{i=1}^{k} p_{B'_i}$ and $d + \sum_{i=1}^{k} p_{H'_i} + \sum_{i=1}^{k} p_{B'_i} = P_{\text{sum}}$, we get $d + \sum_{i=1}^{k} p_{H'_i} \leq C^*_{B'_{k+1}}$ which results in the proof of the theorem in this subcase.

Case 2. The first tardy job before sorting phase (step 8) is not tardy in optimal sequence.

Figure 5 shows the sequence of tardy jobs before and after sorting in step 8. Assume that B'_k contains tardy jobs in optimal sequence that are not tardy in sequence G'. B'_k cannot be empty because in that condition jobs in B'_1 to B'_{k-1} must fill the whole tardiness period from d to P_{sum}, and considering $\sum_{i=1}^{k-1} p_{B'_i} \geq P_{\text{sum}} - d$, there is no need that the heuristic algorithm selects tardy jobs in H'_k. Without loss of generality, the first job in H'_k can be substituted by two dummy jobs, a tardy job having the same tardiness factors as the first job in H'_k and processing time $C^{G'}_{H'_k} - d$ and an early job having tardiness factors equal to zero and processing time $\sum_{i \in H'_k} p_i - (C^{G'}_{H'_k} - d)$. This substitution gets H'_k to exactly fill the tardiness period while it has no effect on the heuristic penalty and will not increase the optimal penalty.

From Figure 5 and (A.2), we can show the relation between the optimal and heuristic sequences as follows:

$$\begin{aligned} Z^G &\leq Z^* \\ &\quad + \sum_{j=1}^{k} \left[\sum_{g_{[i]} \in H_j} \left(u_{g_{[i]}} + \sum_{g_{[i]} \in H_j} w_{g_{[i]}} \left(\sum_{r=j+1}^{k} p_{H_r} + \sum_{r=g_{[i]}+1}^{g_{[m_j]}} p_r \right) \right) \right], \\ Z^G &\leq Z^* + Z^{[H_k,\ldots,H_1]}_{\left(d+\sum_{i=1}^{k} p_{H_i}\right)}. \end{aligned} \quad \text{(A.11)}$$

Finally, according to $\sum_{i=1}^{k} p_{H'_i} < p_{B'_k}$ and $d + \sum_{i=1}^{k} p_{H'_i} \leq C^*_{B'_k}$ we conclude the proof of the theorem in this case.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] E. L. Lawler, "A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness," *Annals of Discrete Mathematics*, vol. 1, pp. 331–342, 1977.

[2] J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker, "Complexity of machine scheduling problems," *Annals of Discrete Mathematics*, vol. 1, pp. 343–362, 1977.

[3] S. G. Kolliopoulos and G. Steiner, "Approximation algorithms for minimizing the total weighted tardiness on a single machine," *Theoretical Computer Science*, vol. 355, no. 3, pp. 261–273, 2006.

[4] T. C. E. Cheng, C. T. Ng, J. J. Yuan, and Z. H. Liu, "Single machine scheduling to minimize total weighted tardiness," *European Journal of Operational Research*, vol. 165, no. 2, pp. 423–443, 2005.

[5] G. Karakostas, S. G. Kolliopoulos, and J. Wang, "An FPTAS for the inimum total weighted tardiness problem with a fixed number of distinct due dates," in *Proceedings of the 15th International Computing and Combinatorics Conference*, Niagara Falls, 2009.

[6] J. J. Yuan, "The NP-hardness of the single machine common due date weighted tardiness problem," *Systems Science and Mathematical Sciences*, vol. 5, no. 4, pp. 328–333, 1992.

[7] E. L. Lawler and J. M. Moore, "Functional equation and its application to resource allocation and sequencing problems," *Mgmt Science*, vol. 16, no. 1, pp. 77–84, 1969.

[8] Y. Fathi and H. W. L. Nuttle, "Heuristics for the common due date weighted tardiness problem," *IIE Transactions (Institute of Industrial Engineers)*, vol. 22, no. 3, pp. 215–225, 1990.

[9] H. Kellerer and V. A. Strusevich, "A fully polynomial approximation scheme for the single machine weighted total tardiness problem with a common due date," *Theoretical Computer Science*, vol. 369, no. 1–3, pp. 230–238, 2006.

[10] I. Kacem, "Fully polynomial time approximation scheme for the total weighted tardiness minimization with a common due date," *Discrete Applied Mathematics. The Journal of Combinatorial Algorithms, Informatics and Computational Sciences*, vol. 158, no. 9, pp. 1035–1040, 2010.

[11] J. Du and J. Y.-T. Leung, "Minimizing total tardiness on one machine is NP-hard," *Mathematics of Operations Research*, vol. 15, no. 3, pp. 483–495, 1990.

[12] E. L. Lawler, "A fully polynomial approximation scheme for the total tardiness problem," *Operations Research Letters*, vol. 1, no. 6, pp. 207–208, 1982.

[13] C. Koulamas, "A faster fully polynomial approximation scheme for the single-machine total tardiness problem," *European Journal of Operational Research*, vol. 193, no. 2, pp. 637–638, 2009.

[14] M. Y. Kovalyov, "Improving the complexities of approximation algorithms for optimization problems," *Operations Research Letters*, vol. 17, no. 2, pp. 85–87, 1995.

[15] F. Della Croce, A. Grosso, and V. Th. Paschos, "Lower bounds on the approximation ratios of leading heuristics for the single-machine total tardiness problem," *Journal of Scheduling*, vol. 7, no. 1, pp. 85–91, 2004.

[16] M. Y. Kovalyov and F. Werner, "Approximation schemes for scheduling jobs with common due date on parallel machines to minimize total tardiness," *Journal of Heuristics*, vol. 8, no. 4, pp. 415–428, 2002.

[17] O. H. Ibarra and C. E. Kim, "Fast approximation algorithms for the knapsack and sum of subset problems," *Journal of the Association for Computing Machinery*, vol. 22, no. 4, pp. 463–468, 1975.

[18] D. Shabtay, Y. Bensoussan, and M. Kaspi, "A bicriteria approach to maximize the weighted number of just-in-time jobs and to minimize the total resource consumption cost in a two-machine flow-shop scheduling system," *International Journal of Production Economics*, vol. 136, no. 1, pp. 67–74, 2012.

[19] D. Shabtay and Y. Bensoussan, "Maximizing the weighted number of just-in-time jobs in several two-machine scheduling systems," *Journal of Scheduling*, vol. 15, no. 1, pp. 39–47, 2012.

[20] M. Ji and T. C. E. Cheng, "Batch scheduling of simple linear deteriorating jobs on a single machine to minimize makespan," *European Journal of Operational Research*, vol. 202, no. 1, pp. 90–98, 2010.

[21] G. Steiner and R. Zhang, "Approximation algorithms for minimizing the total weighted number of late jobs with late deliveries in two-level supply chains," *Journal of Scheduling*, vol. 12, no. 6, pp. 565–574, 2009.

[22] I. Kacem and A. R. Mahjoub, "Fully polynomial time approximation scheme for the weighted flow-time minimization on a single machine with a fixed non-availability interval," *Computers and Industrial Engineering*, vol. 56, no. 4, pp. 1708–1712, 2009.