International Journal of Industrial Engineering & Production Research (2017)

September 2017, Volume 28, Number 3
pp. 221- 240

DOI: 10.22068/ijiepr.28.3.221 http://IJIEPR.iust.ac.ir/

Minimizing a General Penalty Function on a Single Machine Via Developing Approximation Algorithms and FPTASs

Kamran Kianfar [*] **& Gasem Moslehi**

Kamran Kianfar, *Faculty of Engineering, University of Isfahan*
Gasem Moslehi, *Department of Industrial and Systems Engineering, Isfahan University of Technology*

KEYWORDS

Single machine scheduling,
Tardy/Lost penalty,
Dynamic programming,
Approximation algorithm,
FPTAS.

ABSTRACT

This paper addresses the Tardy/Lost penalty minimization on a single machine. According to this penalty criterion, if the tardiness of a job exceeds a predefined value, the job will be lost and penalized by a fixed value. Besides its application in real world problems, Tardy/Lost measure is a general form of popular objective functions such as weighted tardiness, late work and tardiness with rejection; hence, the results of this study are applicable to them. Initially, we present two approximation algorithms. Then, two special cases of the main problem are considered. In the first case, all jobs have the same tardiness weights where a fully polynomial time approximation scheme (FPTAS) is developed using the technique of "structuring the execution of an algorithm". The second special case occurs when none of the jobs can be early. For this case, a 2-approximation and a dynamic programming algorithms are developed, were the latter is converted to an FPTAS.

1. Introduction

In this paper, we study a single machine scheduling problem to minimize Tardy/Lost penalties of common due dates. Every job i $(i = 1,2,\dots,n)$ has a processing time, p_i, a tardiness weight, w_i, a loss penalty factor, s_i, and two due dates d_i^1 and d_i^2. In the case that a job is completed before the first due date, d_1, no penalty is assigned; if the completion time is between d_1 and d_2, the job will be penalized by a linear tardiness penalty; otherwise, the job will be penalized by a fixed loss penalty, s_i. It is assumed that lost jobs must still be processed on machines, and loss penalty for job i is greater than the maximum allowed tardiness penalty.

Fig. 1 shows the Tardy/Lost penalty function for job *i* based on its completion time.

All data are supposed to be positive integers, and jobs preemption is not allowed. The jobs have two common due dates, called $\boldsymbol{d_1}$ and $\boldsymbol{d_2}$. The machine is continually available from time zero and it can process at most one job at a time. The resulting problem is denoted by $\boldsymbol{1|d_i^1 = d_1, d_i^2 = d_2|TL}$, where *TL* indicates the Tardy/Lost penalty function. For the sake of brevity, we name this problem as $\boldsymbol{P1}$ in the rest of this paper.

* **Corresponding author:** ***Kamran Kianfar***

Email: k.kianfar@eng.ui.ac.ir

Received 18 January 2017; revised 21 May 2017; accepted 16 July 2017

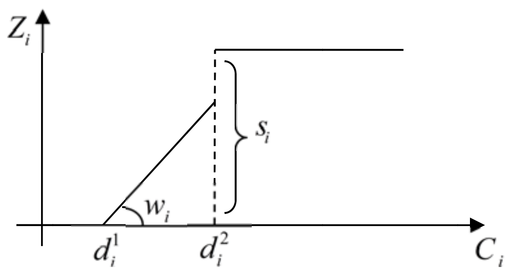

Fig. 1. Tardy/Lost penalty function with common due dates

Yuan [1] showed the weighted tardiness minimization problem with a common due date NP-hard. Since weighted tardiness is a special case of Tardy/Lost penalty function, the problem considered in this paper is also assumed NP-hard.

Objective functions of real-life scheduling problems are often much more complex than the well-known performance measures such as weighted tardiness. Depending on the type of contractual penalties and expected goodwill of future revenue losses incurred, many types of non-linear tardiness penalty functions may arise. Tardy/Lost penalty function combines the features of two well-known performance measures: weighted tardiness and weighted number of tardy jobs.

From a practical point of view, *TL* penalty function is applicable in delivery contracts, most of which are arranged based on two due dates. If an order is early, then no penalty occurs; the order will be penalized if its delivery time exceeds the first due date. The penalty will increase in proportion to the delivery time until the second due date is reached. If the order be delivered later than the second due date, it becomes lost and takes a maximum fixed penalty. The Tardy/Lost performance measure can be considered as a special case for scheduling problems with order acceptance assumption. Here, the objective is minimizing weighted tardiness on a single machine and a common due date where the rejection cost for job i can be defined as loss penalty s_i. Shabtay [2], in the survey paper, studied some scheduling problems with order acceptance assumption.

The main contributions of this paper can be summarized as: (1) Introducing a new and general penalty function for scheduling problems, which can be applied in many practical situations. (2) A number of the scheduling performance measures, such as weighted tardiness, weighted number of tardy jobs, weighted late work, tardiness with rejection ability, and weighted completion time, are special cases for the *TL* function and all the developed algorithms can be applied to those cases. Even knapsack problem is a special case for the introduced problem. (3) According to Woeginger [3], an FPTAS is the strongest possible polynomial time approximation result that we can derive for an NP-hard problem; we have developed some FPTASs with very low time complexities for a general problem and its special cases. (4) We have developed three approximation algorithms, and all of them are proved tight.

In sections 4 and 5, two special cases for the *TL* penalty function are discussed. The first special case assigns a common tardiness weight to all jobs. According to this assumption, all jobs have the same tardiness weights, yet with different loss penalties. In the second special case, we set the first due date of jobs equal to zero; so, the problem changes to that of minimizing weighted completion time of jobs supposing that each job completed after due date is lost with a fixed amount of penalty.

Table 1 provides some assumptions under which the *TL* penalty function and its two special cases are converted to some traditional performance measures in scheduling literature.

Tab. 1. Some measures derived from Tardy/Lost penalty function and its special cases

Assumptio	main penalty function	special case 1	special case 2

$d_i^2 = \infty$	weighted tardiness	tardiness	weighted completion time
$d_i^2 = d_i^1$	weighted number of tardy jobs	weighted number of tardy jobs	
$s_i = w_i(d_i^2 - d_i^1)$	weighted tardiness with rejection modified weighted late work [4]	tardiness with rejection	weighted completion time with rejection
$s_i = w_i(d_i^2 - d_i^1)$	weighted late work	late work	

According to Table 1, scheduling problems with rejection are special cases for the Tardy/Lost performance measure. If we set $s_i = w_i(d_i^2 - d_i^1)$ for each job i, then the TL penalty function is equivalent to minimizing total weighted tardiness with rejection, because we can compare tardiness and rejection costs for a job and select the smaller one in rejection problems. Rejection assumption has been investigated widely in the literature. Engels et al. [5] investigated the problem of minimizing weighted completion times and rejection penalties and developed some approximation algorithms. Shabtay et al. [6] proposed a bi-criteria approach to scheduling a single machine with rejection and positional penalties. The quality of a solution is measured by scheduling criterion $F1$ which is dependent on the completion times of accepted jobs and the total rejection cost $F2$. They considered four optimization problems of (i) minimizing $F1$+$F2$, (ii) minimizing $F1$ subject to$F2 \leq R$, (iii) minimizing $F2$ subject to $F1 \leq R$, and (iv) identifying the set of Pareto-optimal schedules for $(F1, F2)$. If criterion $F1$ represents total tardiness or completion time of jobs, then problem (i) is a special case for our proposed Tardy/Lost penalty function where $s_i = w_i(d_i^2 - d_i^1)$. The survey papers by Slotnick [7] and Shabtay et al. [2] study a number of scheduling problems with rejection.

The problem $1||\sum \bar{T}_w$ is NP-hard in a strong sense if the tardiness weights are not all equal [8] and is optimally solvable in pseudo-polynomial time for a fixed number of distinct due dates [9]. Cheng et al. [10] showed that the schedule that minimizes $\max_j w_j T_j$ gives an $(n$-1$)$-approximation for this problem. Kolliopoulos and Steiner [9] designed pseudo-polynomial algorithms for the case that there is only a fixed number of different due dates. They also developed an FPTAS if, in addition, the tardiness weights are bounded by a polynomial function of n. Karakostas et al. [11] considered the same problem and designed a pseudo-polynomial algorithm and applied a rounding scheme to obtain an approximation scheme.

In a special case of problem $1||\sum \bar{T}_w$where due date is common for all jobs, Lawler and Moore [12] provided a pseudo-polynomial dynamic programming algorithm in $O(n^2d)$ time and Fathi and Nuttle [13] developed a 2-appromation algorithm that requires $O(n^2)$ time. Kellerer and Strusevich [14] proposed an FPTAS of $O(n^6 \log W/\varepsilon^3)$ time complexity where W is the sum of tardiness weights; later, Kianfar and Moslehi [15] studied the same problem and developed another approach to obtain a more effective FPTAS in $O(n^2/\varepsilon)$ time.

If the tardiness weights are equal, the problem $1||\bar{T}$ is NP-hard in the ordinary sense as proved by Du and Leong [16] and it is solvable by a pseudo-polynomial dynamic programming algorithm proposed by Lawler [17]. For this problem, Lawler [18] proposed a dynamic programming algorithm and converted it into an FPTAS of $O(n^7/\varepsilon)$ complexity. Koulamas [19] provided 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 [20] bound improvement procedure. Della Croce et al. [21] considered some popular constructive and decomposition heuristics, and concluded that none of them guarantees a constant worst-case ratio bound. Kovalyov and Werner [22] studied the approximability of this problem on parallel machines with a common due date.

A number of researchers have addressed the problem of late work minimization on a single machine. Potts and Van Wassenhove [23] proposed a polynomial time algorithm based on the similarity between tardiness and late work parameters. In another study [24], they developed a branch-and-bound algorithm for the problem which formed a family of approximation algorithms based on truncated enumeration. References [25-29] may be consulted for other studies devoted to the late work criterion. The results concerning late work are partially reviewed in Chen et al. [30] and Leung [31], but Sterna [32] addresses the first complete review of the topic.

Kethley and Alidaee [4] modified the definition of the late work criterion by introducing two due dates for each job, called due date and deadline. They called the proposed performance criterion as "*modified Weighted Late Work*" which is a special case for the Tardy/Lost penalty function we consider in this study (see Table 1). Kianfar and Moslehi [33] considered the Tardy/Lost penalty function on a single machine scheduling problem and developed two dynamic programming algorithms as well as a branch-and-bound. In another study [34], they proposed a 2-approximation algorithm and a dynamic programming for a special case of the Tardy/Lost penalty function. Biased tardiness penalty is another special case for the Tardy/Lost measure, studied by Moslehi and Kianfar [35].

The performance measure we study in this paper is a kind of regular measure that is non-decreasing in completion times of jobs. With such performance measures, jobs are penalized only for being tardy, and all inventory costs due to early completions are ignored. Pinedo [36] indicated that, in practice, the penalty function associated with a scheduling problem may follow a function in which early jobs are assigned no penalty and those that are finished after their due dates are assigned a penalty that increases at a given rate. Within the penalty function, the job reaches a point where the penalty assignment changes and increases at a much slower pace. The function identified by Pinedo [36] is general; however, two more specific functions that react similarly are the late work criterion [23, 24] and the deferral cost function [37].

Deferral cost functions have been studied by Kahlbacher [38] who considered general penalty functions monotonous with respect to absolute lateness. He examined several specific cases of the penalty function for situations in which machine idle times are allowed or not. He also constructed an FPTAS of the order $O(n^3/\varepsilon)$, where ε denotes precision of the approximation solution. Federgruen and Mosheiov [39] considered a class of single machine scheduling problems with a common due date and general earliness and tardiness penalties. In this study, some polynomial greedy algorithms were proposed and, for convex cost structures, they also examined the worst-case ratio bound if the due date was non-restrictive. Baptiste and Sadykov [40] considered the objective of minimizing a piecewise linear function. They introduced a new Mixed Integer Programming (MIP) model based on time interval decomposition.

Zhou and Cai [41] examined two types of regular performance measures, the total cost and maximum cost, with general cost functions. They studied a stochastic scheduling model on a single machine where processing times are random variables and the machine is subject to stochastic breakdowns. In the paper by Shabtay[42], two continuous and non-decreasing objective functions are considered that include penalties due to earliness, tardiness, number of tardy jobs, and due date assignments. The research by Ventura and Radhakrishnan [43] was focused on scheduling jobs with varying processing times and distinct due dates on a single machine.

Approximation algorithms and FPTASs are common methods to analyze optimization problems. Some examples about implementing FPTAS algorithms in real-world cases can be found in references [44-48].

The rest of this paper is organized as follows. In Sections 2 and 3, we propose two approximation algorithms and examine their worst-case ratio bounds. Section 4 deals with the first special case of the main problem $P1$, called problem $P2$, where a dynamic programming algorithm is converted to an FPTAS. In Section 5, a 2-approximation algorithm, as well as an FPTAS, is proposed for the second special case of problem $P1$, called $P3$. Concluding remarks will be presented in Section 6.

2. LTW/LLP[1] Approximation Algorithm

In this section, an approximation algorithm with a bounded worst-case error is proposed for problem $P1$. Then, using a numerical example, we show that the obtained error bound is tight for this problem. We refer to this algorithm as *LTW/LLP* because it selects tardy jobs based on Largest Tardiness Weight and selects lost jobs based on Largest Loss Penalty.

The algorithm requires at most n iterations while, in each iteration k, a job is scheduled in the kth position from the end of sequence, i.e., position (n-k+1). Since TL penalty function is a type of ordinary performance measures, no idle time is considered for a machine. So, we assume that jobs are scheduled continually from time zero to $P_{sum}=\sum_{i=1}^{n}p_i$. Now, let's define some notations used in the remaining of this paper. Also, suppose that each notation, including an asterisk (*), is related to the optimal sequence.

$\sigma^{LTW/LLP}$: The sequence generated by *LTW/LLP* algorithm

Z^{σ}: The penalty value generated by sequence σ

Z_i^{σ}: The penalty of job i in sequence σ

C_i^{σ}: The completion time of job i in sequence σ

$C_{[i]}^{\sigma}$: The completion time of a job in the *i*th position of sequence σ

p_{σ}: The sum processing times of jobs in sequence σ

It may be easily seen that *LTW/LLP* uses a simple sorting of n elements and, hence, runs in $O(n\log n)$ time. The steps of the *LTW/LLP* algorithm are as follows:

Step 1. Let U_w (U_s) be a set of unscheduled jobs sorted according to non-decreasing order of tardiness weights (loss penalties). Set $C_{[n]}^{LTW/LLP}=P_{sum}$ and $r=n$.

Step 2. Schedule the first job in set U_s, job k, into the *r*th position of sequence. Remove job k from U_s and U_w. Set $C_{[r-1]}^{LTW/LLP}=C_{[r]}^{LTW/LLP}-p_k$ and $r=r-1$.

Step 3. If $C_{[r]}^{LTW/LLP}>d_2$, then go back to Step 2; else if $C_{[r]}^{LTW/LLP}>0$, go to Step 4 else, go to Step 7.

Step 4. Schedule the first job in U_w, job k', into the *(n-r+1)*th position of sequence. Remove job k' from U_w. Set $C_{[r-1]}^{LTW/LLP}=C_{[r]}^{LTW/LLP}-p_{k'}$ and $r=r-1$.

Step 5. If $C_{[r]}^{LTW/LLP}>0$, then go back to Step 4; else go to Step 6.

Step 6. Schedule the remaining unscheduled jobs into the beginning of sequence using any arbitrary order

Step 7. Return the *TL* penalty related to the final sequence.

Example 1. Consider a problem $1|d_i^1=d_1,d_i^2=d_2|TL$ with three jobs according to the data from Table 2. Suppose that $d_1=10$ and $d_2=20$.

Tab. 2. Parameters of jobs in Example 1

job i	p_i	w_i	s_i
1	6	4	46
2	11	7	52
3	9	3	42

Initially, $C_{[3]}^{Ltw/LLP}=26$ and $r=3$. The algorithm *LTW/LLP* schedules job 3 with loss penalty 42 to the end of the schedule. We get $C_{[2]}^{LTW/LLP}=17<d_2$ and the algorithm goes to step 4 where job 1 with $w_1=4$ is selected to be scheduled at the second position of the sequence. The remaining job 2 is then positioned to the

beginning of the schedule. The obtained sequence is 2-1-3 with total penalty of 72.

Theorem 1. Let n' be the number of tardy or lost jobs generated by the *LTW/LLP* algorithm for an instance of the problem $P1$. Then, $Z^{LTW/LLP}/Z^* \le n'$.

Proof. Suppose that job k is the most costly job in $\sigma^{LTW/LLP}$, (i.e., $Z_k^{LTW/LLP} \ge Z_i^{LTW/LLP} \forall i = 1,\dots,n$). Here, we distinguish between two different cases:

Case 1. Job k completes at least as late in σ^* as in $\sigma^{LTW/LLP}$, (i.e., $C_k^* \ge C_k^{LTW/LLP}$)

In this case,

$$w_k max\{0, C_k^* - d_1\} \ge w_k max\left\{0, C_k^{LTW/LLP} - d_1\right\} \tag{1}$$

From Eq. (1), it follows that $Z_k^* \ge Z_k^{LTW/LLP}$ and, considering $Z^* \ge Z_k^*$, we get $Z^* \ge Z_k^{LTW/LLP}$.

Case 2. Job k completes earlier in σ^* than it does in $\sigma^{LTW/LLP}$, (i.e., $C_k^* < C_k^{LTW/LLP}$)

In this case, there must be a job m that precedes k in $\sigma^{LTW/LLP}$, but completes at least as late in σ^* as k does in $\sigma^{LTW/LLP}$. We may divide this case into two sub-cases.

Sub-case 2.1. $C_k^{LTW/LLP} \le d_2$

Regarding $C_m^{LTW/LLP} < C_k^{LTW/LLP}$, we get $C_m^{LTW/LLP} < d_2$. Since job m precedes job k in $\sigma^{LTW/LLP}$, it must be true that $w_m \ge w_k$. It follows that

$$\left.\begin{matrix} C_m^* \ge C_k^{LTW/LLP} \\ w_m \ge w_k \end{matrix}\right\} \Rightarrow \begin{cases} w_m max\{0, C_m^* - d_1\} \ge w_k max\left\{0, C_k^{LTW/LLP} - d_1\right\} \\ s_m \ge w_k max\left\{0, C_k^{LTW/LLP} - d_1\right\} \end{cases}$$
$$\Rightarrow \; Z_m^* \ge Z_k^{LTW/LLP} \tag{2}$$

Finally, from $Z^* \ge Z_m^*$, it will be concluded that $Z^* \ge Z_k^{LTW/LLP}$.

Sub-case 2.2. $C_k^{LTW/LLP} > d_2$

Regarding $C_m^{LTW/LLP} < C_k^{LTW/LLP}$ we get $s_m \ge s_k$ and thus,

$$C_m^* \ge C_k^{LTW/LLP} > d_2 \quad \Rightarrow \quad Z_m^* = s_m \ge s_k = Z_k^{LTW/LLP} \tag{3}$$

So, from Eq. (3) and $Z^* \ge Z_m^*$, we derive $Z^* \ge Z_k^{LTW/LLP}$.

There are n' tardy or lost jobs in $\sigma^{LTW/LLP}$ and, from the fact that job k is the most costly one in this sequence, we get

$$Z_k^{LTW/LLP} \le Z^* \quad \Rightarrow \quad Z^{LTW/LLP} \le n' Z_k^{LTW/LLP} \le n' Z^* \tag{4}$$

that completes the proof.

The following example illustrates that the worst-case ratio bound obtained by *LTW/LLP* algorithm is tight for problem $P1$.

Example 2. Consider a problem $P1$ with 2 jobs described in Table 3. Let K be a very big positive integer and set $d_1 = 0$.

Tab. 3. Parameters of jobs in Example 2

Job i	p_i	w_i	s_i
1	d_2	1	K
2	d_2	$\frac{K}{d_2}$	$K+1$

LTW/LLP algorithm gives the sequence $(2,1)$ with $2K$ total penalty while the optimal penalty for this problem is equal to $d_2 + K + 1$ related to the sequence $(1,2)$.Thus,

$$\lim_{K\to\infty} \frac{Z^{LTW/LLP}}{Z^*} = \lim_{K\to\infty} \frac{2K}{K+d_2+1} = 2 \tag{5}$$

Since 2 jobs are tardy or lost in $\sigma^{LTW/LLP}$, the worst-case ratio bound of this algorithm is tight.

3. SPT² Approximation Algorithm

In this algorithm, jobs are sequenced according to non-decreasing order of processing times; therefore, it will be implemented in $O(n\ logn)$ time.

Theorem 2. Let w_{min} and w_{max} be respectively the smallest and largest tardiness weights and s_{min} and s_{max} be the smallest and largest loss penalties. Then, for each instance of problem $P1$, we have $\frac{Z^{SPT}}{Z^*} \leq max\left\{\frac{w_{max}}{w_{min}}, \frac{s_{max}}{s_{min}}\right\}$

Proof. Create n dummy jobs all having tardiness weight $\boldsymbol{w_{min}}$ and loss penalty $\boldsymbol{s_{min}}$. It can be easily tested that if we schedule these dummy jobs according to *SPT* ordering, the related *TL* penalty, called $\boldsymbol{LB}^*$, is a lower bound on the total *TL* penalty of any sequence for the real jobs. Similarly, create a set of n dummy jobs all with tardiness weight $\boldsymbol{w_{max}}$ and loss penalty $\boldsymbol{s_{max}}$. We can verify that the associated total penalty, called $\boldsymbol{UB^{SPT}}$, is an upper bound on $\boldsymbol{Z^{SPT}}$ for the real jobs. Let $\boldsymbol{n_E}$, $\boldsymbol{n_T}$, and $\boldsymbol{n_L}$ be the number of early, tardy, and lost jobs in *SPT* ordering, respectively. Then,

$$UB^{SPT} = w_{max}\left[\left(C^{SPT}_{[n_E+1]} - d_1\right) + \ldots + \left(C^{SPT}_{[n_T]} - d_1\right)\right] + s_{max}n_L \tag{6}$$

$$LB^* = w_{min}\left[\left(C^{SPT}_{[n_E+1]} - d_1\right) + \ldots + \left(C^{SPT}_{[n_T]} - d_1\right)\right] + s_{min}n_L \tag{7}$$

And, if we signify the term $\left(C^{SPT}_{[n_E+1]} - d_1\right) + \ldots + \left(C^{SPT}_{[n_T]} - d_1\right)$ by T^{SPT}, then

$$\frac{UB^{SPT}}{LB^*} \leq \frac{w_{max}T^{SPT} + s_{max}n_L}{w_{min}T^{SPT} + s_{min}n_L} \leq max\left\{\frac{w_{max}}{w_{min}}, \frac{s_{max}}{s_{min}}\right\} \tag{8}$$

that completes the proof.

The following example illustrates that the worst-case ratio bound obtained by *SPT* approximation algorithm is tight for problem $P1$.

Example 3. Consider problem $P1$ with 2 jobs described in Table 4. Set $d_1 = K$ and $d_2 = K + 1$, where K is a very big integer.

Tab. 4. Parameters of jobs in Example 3

job i	p_i	w_i	s_i
1	*K*	1	*K*
2	*K*+1	1	2*K*

SPT algorithm generates the sequence $(1,2)$ with total penalty equal to$2K$, while the optimal sequence for this example is $(2,1)$ with total penalty of *K*+1. So,

$$\lim_{K\to\infty} \frac{Z^{SPT}}{Z^*} = \frac{2K}{K+1} = 2 \tag{9}$$

$$max\left\{\frac{w_{max}}{w_{min}}, \frac{s_{max}}{s_{min}}\right\} = max\left\{\frac{1}{1}, \frac{2K}{K}\right\} = 2 \tag{10}$$

The above relations show that the worst-case ratio bound generated by the *SPT* algorithm is tight for this instance and, hence, for problem $P1$ in general form.

4. First Special Case

In this section, we study a special case of the problem defined in section 1 in which all jobs have a common tardiness weight, yet with different loss penalties (problem P2). All other assumptions are the same as the ones in $1|d_i^1 = d_1, d_i^2 = d_2|TL$. This problem will be denoted by $1|d_i^1 = d_1, d_i^2 = d_2, w_i = w|TL$, or simply problem $P2$, and its objective function is described as follows:

$$Z_i = \begin{cases} 0 & if \quad C_i \leq d_1 \\ w(C_i - d_1) & if \quad d_1 < C_i \leq d_2 \\ s_i & if \quad C_i > d_2 \end{cases} \tag{11}$$

Therefore, the problem in this special case is about scheduling n jobs on a single machine with two common due dates with the objective of minimizing total penalty function defined by Eq. (11). First, we discuss the time complexity of problem $P2$. Then, we propose an FPTAS for generating approximate solutions.

4-1. Time complexity analysis of the problem

Here, the time complexity of problem $P2$ is examined, and it will be proved that this problem is at least NP-hard in an ordinary sense. Consider a case in which all tardiness weights are equal to zero and, therefore, the objective function converts to the following form.

$$Z_i = \begin{cases} 0 & if \quad C_i \le d_2 \\ s_i & if \quad C_i > d_2 \end{cases} \tag{12}$$

We want to show that minimizing the above objective function on a single machine has a direct relation with the knapsack problem. The knapsack problem includes a set of n objects I, each of which having a volume equal to v_i and a profit equal to q_i. The goal is to put a sub-set of these objects into a knapsack of volume V, such that the selected objects have the maximum total profit.

It can be easily verified that two problems $P2$ and knapsack are transformable to each other by setting $v_i = p_i$, $q_i = s_i$ and $V = d_2$. This transformation is done by $Z^{knap} + Z^{TL} = \sum_{i=1}^{n} s_i$ where Z^{knap} and Z^{TL} denote the objective functions of the knapsack and $1|d_i^1 = d_1, d_i^2 = d_2, w_i = 0|TL$ problems, and $\sum_{i=1}^{n} s_i$ is the sum of loss penalties. Since $\sum_{i=1}^{n} s_i$ is a constant value, we conclude that minimizing the objective function in problem $P2$ is equivalent to maximizing the sum profits in the knapsack problem.

It is obvious that problem $P2$ is at least as complex as the knapsack problem; so, regarding the fact that the knapsack problem is NP-hard in an ordinary sense, we will conclude that the problem considered in this section is at least a type of NP-hard problems in an ordinary sense.

4-2. Dynamic programming algorithm

Now, we propose a dynamic programming algorithm and convert it to an FPTAS by restricting the number of states generated in each iteration. The main idea of this dynamic programming algorithm is taken from the study by Lawler and Moore [12] where they considered the problem of minimizing tardiness with relative and absolute deadlines that can be denoted as $1|d_i = d, w_i = 1|TL$.

Suppose that jobs are indexed according to non-decreasing order of processing times (*SPT*). An optimal solution is composed of two groups of jobs; the first group includes early and tardy jobs with completion times no more than d_2, while the second group includes lost jobs with completion times greater than d_2. In an optimal sequence, jobs in the first group must be scheduled according to their order of indices (*SPT*) and lost jobs in the second group may have any arbitrary order.

In the following dynamic programming algorithm, jobs in the first group, early/tardy jobs, are continually scheduled from time zero, while lost jobs are scheduled from end of the sequence to the beginning according to their order of indices so that they complete exactly at time $P_{sum} = \sum_{i=1}^{n} p_i$.

In this algorithm, each state in state space $\mathrm{v_k}$ shows a partial sequence for the first k jobs and is indicated by the vector $[t, f]$. Variable t shows the sum of processing times for the jobs in the first group and f is the total penalty of the corresponding partial sequence. This algorithm can be described as follows:

Algorithm DP1

```
1.   Set $v_0 = \{[0,0]\}$
2.   For each $k = \{1,2,\dots,n\}$, consider all states $[t,f]$ in $v_{k-1}$
     o  If $t + p_k \le d_2$, then add state $[t + p_k, f + w\,.max\{0, t + p_k - d_1\}]$ to states space $v_k$
     o  Add $[t, f + s_k]$ to states space $v_k$
2.1. For all the states $[t,f] \in v_k$ with equal value for $t$, keep at most one state having the minimum value of $f$.
2.2. Remove state space $v_{k-1}$
3.   Return the optimal solution $Z^* = \min_{[t,f] \in v_n} \{f\}$
```

To calculate the time complexity of this algorithm, regarding that all input parameters are integers, we can restrict the number of states in each v_k by d_2. This is because variable t can at most take d_2 different values, and in each iteration and for each t, we keep at most one state with the smallest f value. The running time of Step 2 is proportional to $\sum_{k=1}^{n} |v_k|$ and is $O(n.d_2)$. Finally, regarding that Step 3 needs $O(d_2)$ time, the total complexity of this algorithm will be obtained as $O(n\,.d_2)$.

4-3. FPTAS algorithm

This FPTAS is based on two phases. In the first phase, algorithm *LTW/LLP* is used to determine an upper bound for problem $P2$; in the second phase, the execution of algorithm *DP1* is modified in order to reduce the number of states and running time. One common way for transforming a dynamic programming algorithm to FPTAS is the technique of *structuring the execution of an algorithm*. The main idea of this technique is to remove a special part of the states generated by the algorithm in such a way that the modified algorithm becomes faster yielding an approximate solution instead of the optimal one. This method was first introduced by Ibarra and Kim [49] for solving the knapsack problem and has in recent years been extended to numerous scheduling problems (see [50-54]).

Let Z_H denote the objective value returned by the algorithm *LTW/LLP* for an instance of problem $P2$ and ε be the maximum acceptable error for FPTAS. The FPTAS algorithm works on the reduced state space $v_k^{\#}$ instead of v_k. 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 sub-intervals $I_m = [(m-1)\Delta, m\Delta]$, $1 \le m \le L$ of length Δ. For each of the resulting sub-intervals I_m, we keep at most one state with the smallest value of *t*. Given an arbitrary $\varepsilon > 0$, define

$$LB = \frac{Z_H}{n} \quad , \quad L = \left\lceil \frac{n^2}{\varepsilon} \right\rceil \quad ,$$
$$\Delta = \frac{Z_H}{L} \tag{13}$$

Let LB denote a lower bound on the optimal value for problem $P2$. Since, $L = O(n^2/\varepsilon)$, the complexity of the FPTAS algorithm is $O(n^3/\varepsilon)$. This algorithm can be described as follows.

Algorithm FPTAS1

```
1.   Set $v_o^{\#} = \{[0,0]\}$
2.   For each $k = \{1,2,\dots,n\}$, consider all states $[t,f]$ in $v_{k-1}^{\#}$
     o   If $t + p_k \le d_2$, then add state $[t + p_k, f + w.max\{0, t + p_k - d_1\}]$ to states space $v_k^{\#}$
     o   Add $[t, f + s_k]$ to the states space $v_k^{\#}$
2.1.     For all the states $[t,f] \in v_k^{\#}$ with equal value for $t$, keep at most one state having the minimum value of $f$.
2.2.     Let $[t,f]_m$ be a state in $v_k^{\#}$ such that $f \in I_m$ and $t$ has the minimum value. Set $v_k^{\#} = \{[t,f]_m \mid 1 \le m \le L\}$.
2.3.     Remove states space $v_{k-1}^{\#}$
3.   Return the final solution $Z^{\#} = \min_{[t,f]\in v_n^{\#}} \{f\}$.
```

Example 4. This numerical example is designed to describe the procedure of algorithm *FPTAS*1. Consider the data from Table 2 and suppose that all the jobs have a common tardiness weight $w = 4$ and let $\epsilon = 0.5$. Algorithm *LTW/LLP* generates sequence 2-1-3 with total penalty 74. We have $LB = \frac{74}{3} = 24.7$, $L = \left\lceil \frac{9}{0.5} \right\rceil = 18$, and $\Delta = \frac{74}{18} = 4.11$. Jobs 1, 3, and 2 are selected based on the SPT ordering and are scheduled, respectively, during the algorithm's iterations. Initially, $v_0^{\#} = [0,0]$. At the first step, state space $v_1^{\#} = \{[6,0], [0,46]\}$ is created. Four states are generated by the second iteration where the algorithm removes the state [9,46] and the resulting state space is $v_2^{\#} = \{[15,20], [6, 42], [0,88]\}$. Job 2 is scheduled in the third iteration, where five states are generated and, after removing states [17,70] and [11,92], the remaining state space is $v_3^{\#} = \{[15,72], [6,94], [0,140]\}$. Therefore, the final solution is $Z^{\#} = 72$, which obviously is better than the solution from algorithm *LTW/LLP*.

4-4. Worst-case analysis of algorithm FPTAS1

The worst-case analysis is based on a comparison of execution of algorithms *DP1* and *FPTAS1*. We may remark that the main action of *FPTAS1* consists in reducing the cardinality of the state spaces by splitting the interval of f into sub-intervals and then replacing all vectors belonging to the same sub-interval by a single approximate state with the smallest *t*. First, a lemma is provided to prove the worst-case ratio bound of *FPTAS1*.

Lemma 1. Let $[t,f] \in v_k$ be an arbitrary state in *DP1*. The FPTAS algorithm generates at least one state $[t^{\#}, f^{\#}]$ in $v_k^{\#}$, 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 $v_0^{\#} = v_0$. 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 v_k generated by *DP1* from a feasible state $[t', f']$ at iteration $k-1$. Here, two cases can be distinguished, and we prove the statement for iteration *k* in these two cases.

Case 1. $[t,f] = [t' + p_k, f' + w.max\{0, t' + p_k - d_1\}]$

Since $[\boldsymbol{t}', \boldsymbol{f}'] \in \boldsymbol{v_{k-1}}$, there is a state $[\boldsymbol{t}'^{\#}, \boldsymbol{f}'^{\#}] \in \boldsymbol{v_{k-1}^{\#}}$, such that $\boldsymbol{t}'^{\#} \le \boldsymbol{t}'$ and $\boldsymbol{f}'^{\#} \le \boldsymbol{f}' + (\boldsymbol{k} - \boldsymbol{1})\boldsymbol{\Delta}$. Therefore, *FPTAS1* generates the state $[\boldsymbol{t}'^{\#} + \boldsymbol{p_k}, \boldsymbol{f}'^{\#} + \boldsymbol{w}.\boldsymbol{max}\{\boldsymbol{0}, \boldsymbol{t}'^{\#} + \boldsymbol{p_k} - \boldsymbol{d_1}\}]$ in the k^{th} iteration that may be eliminated when cleaning up the state sub-set. Let $[\boldsymbol{\lambda}, \boldsymbol{\mu}]$ be the remaining state in $\boldsymbol{v_k^{\#}}$ that is at the same interval as $[t'^{\#}+p_k, f'^{\#}+w\,.max\{0, t'^{\#}+p_k-d_1\}]$. Thus, we drive that

$$\lambda \le t'^{\#}+p_k \le t'+p_k \\ = t \tag{14}$$

and

$$\mu \le f'^{\#}+w.max\{0,t'^{\#}+p_k-d_1\}+\Delta \\ \le f'+(k-1)\Delta+w.max\{0,t'+p_k-d_1\}+\Delta \\ = f+k\Delta \tag{15}$$

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

Case 2. $[t,f]=[t',f'+s_k]$

Since the state $[t',f']$ is in v_{k-1}, there exists a state $[t'^{\#},f'^{\#}]\in v_{k-1}^{\#}$ such that $t'^{\#}\le t'$ and $f'^{\#}\le f'+(k-1)\Delta$. Therefore, *FPTAS1* generates the state $[t'^{\#},f'^{\#}+s_k]$ in iteration k that may be eliminated during the cleaning up procedure. Let $[\lambda',\mu']$ be the remaining state in $v_k^{\#}$ that is at the same interval as $[t'^{\#},f'^{\#}+s_k]$. So,

$$\lambda' \le t'^{\#} \le t' \\ = t \tag{16}$$

and

$$\mu' \le f'^{\#}+s_k+\Delta \\ \le f'+(k-1)\Delta+s_k+\Delta \\ = f+k\Delta \tag{17}$$

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

Theorem 3. Given an arbitrary $\varepsilon>0$, algorithm *FPTAS1* outputs a sequence with $Z^{\#}$ penalty for problem $P2$, such that $Z^{\#}-Z^{*}\le \varepsilon\ Z^{*}$.

Proof. By definition, the optimal sequence can be related to a state $[t^*,f^*]$ in v_n. According to Lemma 1, the algorithm *FPTAS1* generates a state $[t^{\#},f^{\#}]$ in $v_n^{\#}$ such that $t^{\#}\le t^*$ and

$$f^{\#}\le f^*+n\Delta=f^*+n\frac{Z_H}{L}=f^*+n\frac{n.LB}{\lceil n^2/\varepsilon\rceil} \\ \le f^*+\varepsilon\,.LB \\ \le (1+\varepsilon)f^* \tag{18}$$

It is clear that there always exists a feasible state in *FPTAS1* related to any feasible state generated by *DP1*. It is because $t^{\#}\le t$ holds in all states of the FPTAS algorithm and none of the states will be lost by constraint $t+p_k\le d_2$ in Step 2. This will complete the proof.

5. Second Special Case

In this section, we study another special case of the main problem $1|d_i^1=d_1,d_i^2=d_2|TL$ in which the first common due date is equal to zero. This problem will be denoted by $1|d_i^1=0,d_i^2=d_2|TL$, or simply problem $P3$, and its objective function is described as follows:

$$Z_i = \begin{cases} w_i.C_i & if \quad C_i\le d_2 \\ s_i & if \quad C_i>d_2 \end{cases} \tag{19}$$

Formally, the problem in this special case is about scheduling *n* jobs on a single machine with one common due date and objective of minimizing total penalty function defined by Eq. (19). Similar to the previous special case, the time complexity of problem $P3$ may be examined, and it will be proved that this problem is at least NP-hard in an ordinary sense. Consider a case in which all tardiness weights, w_i, are equal to zero. The resulting objective function is the same as Eq. (12), and then, it can be transformed to the knapsack problem (see section 4.1). Since problem $P3$ is at least as complex as the knapsack problem, we will conclude that the problem considered in this section is at least NP-hard in an ordinary sense.

5-1. MLCR[2] approximation algorithm

Here, we present an approximation algorithm for problem $P3$ and show that the worst-case ratio bound of this algorithm is equal to 2. We refer to this algorithm as *MLCR* and name the sequence it generates as *G*. We also adopt notation $g_{[i]}$ to represent a job in the ith position in sequence $G=(g_{[1]},g_{[2]},\dots,g_{[n]})$. This algorithm requires at most *n* iterations while in each iteration *r*, the $(n-r+1)^{th}$ element of *G*, $g_{[n-r+1]}$, is determined among the unscheduled jobs that generates the minimum value for $s_i/min(C_{[n-r+1]}^{G}-d_2,p_i)$. Furthermore, if there exists an unscheduled job filling the remaining period through $d_2$during an iteration, the relevant sequence (sequence $\widehat{G}$) will be saved in addition to the main sequence *G*. The algorithm iterates until the sequence of scheduled jobs

2 - Minimum Loss Cost Ratio

passes d_2 and then, puts the unscheduled jobs into the beginnings of sequences G and $\hat{G}$ based on the weighted shortest processing time (*WSPT*) ordering. The algorithm returns the best of two sequences G and $\hat{G}$ as the final result. Let $Z^G_{[r,n]}$ denote sum penalties related to the jobs from *r*th position through the last job in sequence *G*. The steps of the algorithm are as follows:

Step 1. Let $U = \{1,2,..,n\}$ be a set of unscheduled jobs and *r* be the counter index of positions in the sequence. Set $C_{[n]} = P_{sum}$ and $r = n$.

Step 2. Define $\hat{U} = \{i \in U \mid p_i \geq C_{[n]} - d_2\}$. If $\hat{U}$ is empty, then $\hat{Z}_{best} = \infty$; else, select a job $\hat{\imath}$ with minimum loss penalty (s_i) from $\hat{U}$. Let $\hat{g}_{[n]} = \hat{\imath}$ and $\hat{Z}_{best} = s_{\hat{\imath}}$.

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

$$\theta_i = \frac{s_i}{min(p_i, C_{[r]} - d)} \tag{20}$$

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

Step 5. If $\hat{U} = \{i \in U \mid p_i \geq C_{[r-1]} - d_2\}$ is not empty, select job $\hat{\imath}$ with the minimum loss penalty from Ûand set $\hat{Z} = s_{\hat{\imath}}$. If $\hat{Z} + Z^G_{[r,n]} < \hat{Z}_{best}$, then $\hat{Z}_{best} = \hat{Z} + Z^G_{[r,n]}$ and $\hat{g}_{[i]} = g_{[i]}$ $\forall i = r,\dots,n$ and $\hat{g}_{[r-1]} = \hat{\imath}$.

Step 6. If $C_{[r-1]} > d_2$, then $r = r - 1$ and go back to Step 3; else, go to Step 7.

Step 7. Put unscheduled jobs at the beginning of each of sequences G and $\hat{G}$ based on the WSPT ordering.

Step 8. If $Z^G \leq Z^{\hat{G}}$, return sequence $G = (g_{[1]}, g_{[2]},\dots,g_{[n]})$; else, return sequence $\hat{G} = (\hat{g}_{[1]}, \hat{g}_{[2]},\dots,\hat{g}_{[n]})$.

This algorithm includes a simple sorting of n elements, and hence, its time complexity is $O(n\,logn)$. Next, we will provide a numerical example to illustrate how the algorithm *MLCR* works.

Example 5. Consider problem $P3$ with 4 jobs described in Table 5. Set $d_1 = 0$ and $d_2 = 20$.

Tab. 5. Parameters of jobs in Example 5

job i	p_i	w_i	s_i
1	5	2	80
2	14	3	150
3	8	5	250
4	10	6	140

First, $C_{[4]} = 37$, $\hat{U}$ is empty, and $\hat{Z}_{best} = \infty$. In the first iteration ($r = 4$), we have $\theta_1 = 16, \theta_2 = 37.5, \theta_3 = 31.25, \theta_4 = 14$ where job 4 has the minimum value and, hence, $g_{[4]} = 4$. Furthermore, $C_{[3]} = 27$, $\hat{U} = \{2,3\}$, job 2 is selected to fill the lost period, and $\hat{Z} = 150$. From $Z_{g[4,4]} = 140$, we get $\hat{Z}_{best} = 290$, $\hat{g}_{[4]} = 4$, and $\hat{g}_{[3]} = 2$. In second iteration ($r = 3$), we get $\theta_1 = 16, \theta_2 = 21.43, \theta_3 = 35.71$ where job 1 has the minimum value; so, $g_{[3]} = 1$. Simply, we get $C_{[2]} = 22$, $\hat{U} = \{2,3\}$, $\hat{\imath} = 2$, $\hat{Z} = 150$ and, considering that $\hat{Z} + Z^G_{[3,4]} = 150 + 220 > \hat{Z}_{best} = 290$, the value of $\hat{Z}_{best}$ will remain unchanged and equal to 290. In the next iteration ($r = 2$), job 2 is sequenced into the second position of *G* while $C_{[1]} = 8$. By assigning the remaining unscheduled jobs to the beginning of the sequences, we will get $G = (3,2,1,4)$ and $\hat{G} = (3,1,2,4)$. Finally, $Z^G = 510$ and also $Z^{\hat{G}} = 356$ will be obtained and sequence $\hat{G}$ is returned as the approximate solution.

In the following, we present two theorems about problem $P3$, which are used for proving the worst-case ratio bound of algorithm *MLCR*.

Theorem 4. In any optimal sequence for problem $P3$, the tardy jobs with start time less than or equal to d_2 must be sequenced by WSPT ordering.

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

Theorem 5. Consider problem $P3$. Define two sequences $\sigma = (\sigma_1,\dots,\sigma_m)$ and $\sigma' = (\sigma'_1,\dots,\sigma'_{m'})$ of lost jobs on the common time interval $[d_2, P_{sum}]$, where relation $\frac{Z^\sigma_i}{min(p_i, C^\sigma_i - d_2)} \leq \frac{Z^{\sigma'}_j}{min(p_j, C^{\sigma'}_j - d_2)}$ holds for all $i \in \sigma$ and $\forall j \in \sigma'$; If $p_\sigma \leq p_{\sigma'}$ holds, then $Z^\sigma \leq Z^{\sigma'}$.

Proof. Consider two sequences $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma'}$ with loss penalties shown in Fig. 2. Suppose that $\boldsymbol{\theta^\sigma(t)}$ and $\boldsymbol{\theta^{\sigma'}(t)}$ denote the slopes of the functions related to sequences $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma'}$ at time *t*, respectively; then, according to theorem's assumption, we have $\boldsymbol{\theta^{\sigma}(t) \leq \theta^{\sigma'}(t)}$ for all $\boldsymbol{t \in [d_2, P_{sum}]}$. Obviously, for all $\boldsymbol{t \in [d_2, P_{sum}]}$, the function related to sequence $\boldsymbol{\sigma}$ falls under the function related to sequence $\boldsymbol{\sigma'}$; regarding $\boldsymbol{p_{\sigma} \leq p_{\sigma'}}$ we get $\boldsymbol{Z^{\sigma} \leq Z^{\sigma'}}$.

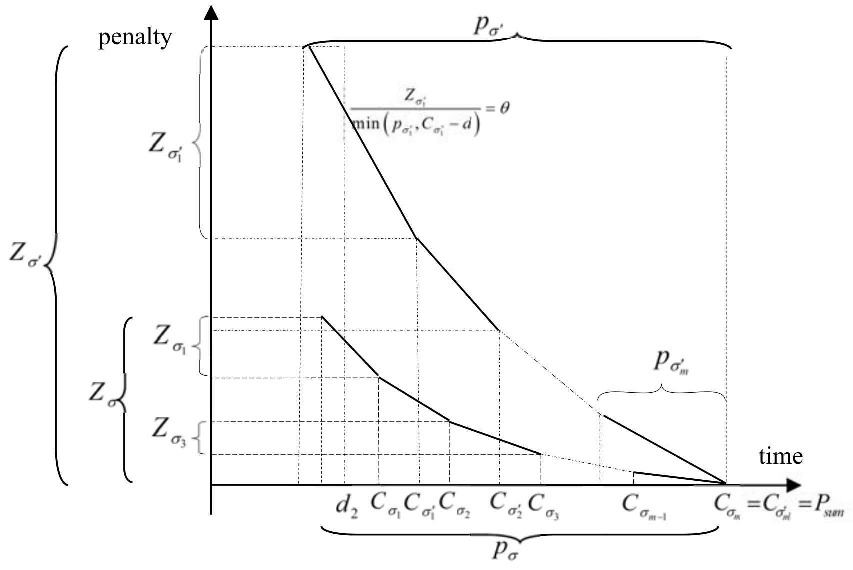

Fig. 2. Penalties related to sequences σ and σ'

Theorem 6. Algorithm *MLCR* is a 2-approximation for problem $P3$.

Proof. Recall that G is the sequence generated by algorithm *MLCR* and each notation including an asterisk (*) is related to the optimal sequence. Algorithm *MLCR* schedules jobs iteratively from the end of the sequence to the beginning. Through this procedure, jobs are classified into two groups: tardy and lost. On the other hand, each job may be tardy, or lost in optimal sequence. Thus, we get four groups of jobs: $\{\boldsymbol{H}, \boldsymbol{B}, \boldsymbol{R}, \boldsymbol{Q}\}$, as shown in Fig. 3. Sets $\boldsymbol{H}$ include a number of jobs which are lost in G and tardy in the optimal sequence. Sets $\boldsymbol{R}$ include some other jobs which are tardy in G and lost in the optimal sequence. Sets $\boldsymbol{B}$ and $\boldsymbol{Q}$ indicate the lost jobs in both sequences and the tardy jobs in both sequences, respectively. For example, jobs in sets $\{\boldsymbol{H_1^G, H_2^G, \ldots, H_{k1}^G}\}$ and $\{\boldsymbol{H_1^*, H_2^*, \ldots, H_{m2}^*}\}$ are the same jobs, but differently grouped by sequence G and optimal sequence.

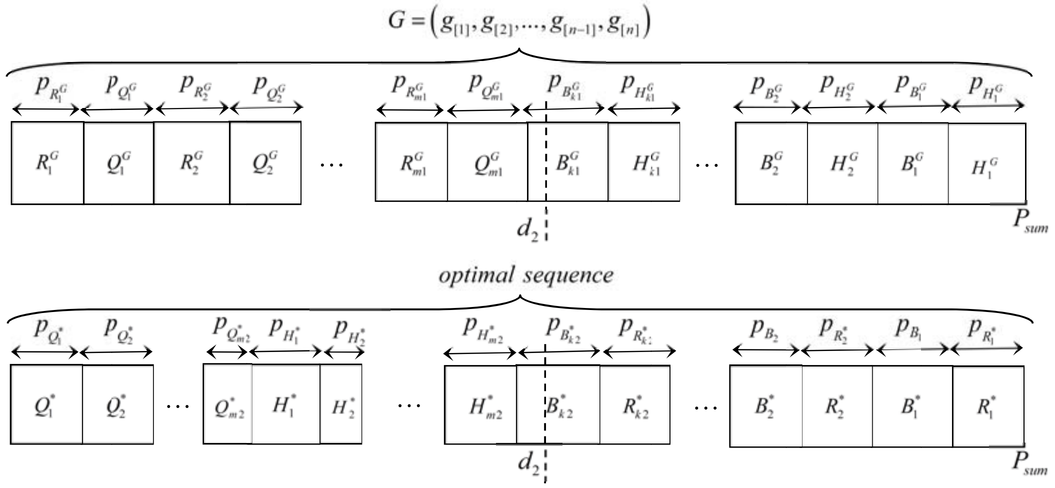
$G = (g_{[1]}, g_{[2]}, \ldots, g_{[n-1]}, g_{[n]})$
$p_{R_1^G}$ $p_{Q_1^G}$ $p_{R_2^G}$ $p_{Q_2^G}$ $p_{R_{m1}^G}$ $p_{Q_{m1}^G}$ $p_{B_{k1}^G}$ $p_{H_{k1}^G}$ $p_{B_2^G}$ $p_{H_2^G}$ $p_{B_1^G}$ $p_{H_1^G}$
R_1^G Q_1^G R_2^G Q_2^G … R_{m1}^G Q_{m1}^G B_{k1}^G H_{k1}^G … B_2^G H_2^G B_1^G H_1^G
d_2
P_{sum}
optimal sequence
$p_{Q_1^*}$ $p_{Q_2^*}$ $p_{Q_{m2}^*}$ $p_{H_1^*}$ $p_{H_2^*}$ $p_{H_{m2}^*}$ $p_{B_{k2}^*}$ $p_{R_{k2}^*}$ p_{B_2} $p_{R_2^*}$ p_{B_1} $p_{R_1^*}$
Q_1^* Q_2^* … Q_{m2}^* H_1^* H_2^* … H_{m2}^* B_{k2}^* R_{k2}^* … B_2^* R_2^* B_1^* R_1^*
d_2
P_{sum}

Fig. 3. Sequence of jobs in G and the optimal sequence

First, we compare the penalty of tardy jobs in G (jobs in sets $\boldsymbol{R^G}$ and $\boldsymbol{Q^G}$) with some penalties in the optimal sequence; then, we should repeat this comparison for lost jobs in sequence G. Since jobs in sets *H* are penalized by tardiness weights only in the optimal sequence, the worst-case ratio occurs when the tardiness weight $\boldsymbol{w_i}$ is equal to zero for these sets. Therefore, according to the WSPT ordering for tardy jobs in the optimal sequence, we conclude that sets *H* must be scheduled after sets $\boldsymbol{Q}$ in the optimal sequence. Let $\boldsymbol{Z^G_{\{T\}}}$ and $\boldsymbol{Z^*_{\{T\}}}$ denote the sum penalties for tardy jobs in sequence *G* and the optimal sequence, respectively. Also, $\boldsymbol{w_\sigma}$ indicates the sum tardiness weights of jobs in a sub-sequence $\boldsymbol{\sigma}$.

$$Z^*_{\{T\}} = \sum_{i=1}^{m_2}\sum_{j\in Q^*_i}(w_jC^*_j) + \sum_{i=1}^{m_2}\sum_{j\in H^*_i}(w_jC^*_j) \geq \sum_{i=1}^{m_2}\sum_{j\in Q^*_i}(w_jC^*_j) \tag{21}$$

$$\begin{aligned} Z^G_{\{T\}} &= \textstyle\sum_{i=1}^{m_1}\sum_{j\in Q^G_i}(w_jC^G_j) + \sum_{i=1}^{m_1}\sum_{j\in R^G_i}(w_jC^G_j) \\ &= \textstyle\sum_{i=1}^{m_2}\sum_{j\in Q^*_i}(w_jC^*_j) + \sum_{i=1}^{m_1}\left(w_{Q^G_i}\sum_{j=1}^{i}p_{R^G_j}\right) \\ &\quad + \sum_{i=1}^{m_1}\left(w_{R^G_i}\left(\sum_{j=1}^{i}p_{R^G_j} + \sum_{j=1}^{i-1}p_{Q^G_j}\right)\right) \end{aligned} \tag{22}$$

According to the fact that tardy jobs are arranged according to the WSPT ordering in both sequences *G* and optimal, we get:

$$\begin{aligned} &\frac{p_j}{w_j} \leq \frac{p_{j'}}{w_{j'}} \forall i = 1,2,\dots,m_1 \quad , \quad \forall j \in R^G_i \ , \qquad j' \in Q^G_i \\ &\Rightarrow p_jw_{j'} \leq p_{j'}w_j \forall i = 1,2,\dots,m_1 \quad , \quad \forall j \in R^G_i \ , \qquad j' \in Q^G_i \\ &\Rightarrow p_{R^G_i}w_{Q^G_i} \leq p_{Q^G_i}w_{R^G_i} \forall i = 1,2,\dots,m_1 \end{aligned} \tag{23}$$

and so, we will get

$$\begin{aligned} Z^G_{\{T\}} &\leq \sum_{i=1}^{m_2}\sum_{j\in Q^*_i}(w_jC^*_j) + \sum_{i=1}^{m_1}\left(w_{R^G_i}\sum_{j=i}^{m_1}p_{Q^G_j}\right) + \sum_{i=1}^{m_1}\left(w_{R^G_i}\left(\sum_{j=1}^{i}p_{R^G_j} + \sum_{j=1}^{i-1}p_{Q^G_j}\right)\right) \\ &\leq Z^*_{\{T\}} + \sum_{i=1}^{m_1}\left(w_{R^G_i}\left(\sum_{j=1}^{i}p_{R^G_j} + \sum_{j=1}^{m_1}p_{Q^G_j}\right)\right) \\ &\leq Z^*_{\{T\}} + \sum_{i=1}^{m_1}\sum_{j\in R^G_i}s_j \end{aligned} \tag{24}$$

Now, we compare the penalty of lost jobs in *G* (jobs in sets H^G and B^G) with some penalties in the optimal sequence. Let $Z^G_{\{L\}}$ and $Z^*_{\{L\}}$ denote the sum penalties for lost jobs in the sequence *G* and the optimal sequence, respectively. Suppose $\hat{\jmath}$ as the first lost job in sequence *G*, i.e., $\hat{\jmath} := \left\{\hat{\jmath} \in B^G_{k1}, C^G_{\hat{\jmath}} < C^G_i \quad \forall i \in B^G_{k1}\right\}$. Thus,

$$\left.\begin{aligned} &\sum_{i=1}^{k_1}p_{H^G_i} + \sum_{i=1}^{k_1}p_{B^G_i} - p_{\hat{\jmath}} < P_{sum} - d_2 \\ &\sum_{i=1}^{k_2}p_{R^*_i} + \sum_{i=1}^{k_2}p_{B^*_i} \geq P_{sum} - d_2 \end{aligned}\right\} \rightarrow$$

$$\sum_{i=1}^{k_1} p_{H_i^G} + \sum_{i=1}^{k_1} p_{B_i^G} - p_{\hat{j}} < \sum_{i=1}^{k_2} p_{R_i^*} + \sum_{i=1}^{k_2} p_{B_i^*} \xrightarrow{\sum_{i=1}^{k1} p_{B_i^G} = \sum_{i=1}^{k2} p_{B_i^*}} \sum_{i=1}^{k_1} p_{H_i^G} < \sum_{i=1}^{k_2} p_{R_i^*} + p_{\hat{j}} \tag{25}$$

In Step 3 of the algorithm, *MLCR* jobs are selected according to the non-decreasing order of θ_i values. So, according to the fact that job $\hat{j}$ is selected after all other lost jobs by *MLCR* and regarding that jobs in sets *R* are not selected as lost jobs by *MLCR*,

$$\frac{s_j}{min(p_j, C_j^G - d_2)} \le \frac{s_{\hat{j}}}{min(p_{\hat{j}}, C_{\hat{j}}^G - d_2)} \forall i = 1,\dots,k_1 \quad , \quad \forall j \in H_i^G \tag{26}$$

$$\frac{s_j}{min(p_j, C_j^G - d_2)} \le \frac{s_{j'}}{min(p_{j'}, C_{\hat{j}}^* - d_2)} \forall i = 1,\dots,k_1 \quad , \quad \forall j \in H_i^G \quad \forall i' = 1,\dots,k_2 \quad , \quad \forall j' \in R_{i'}^* \tag{27}$$

By summarizing (25), (26), and (27) and from Theorem 5, we conclude that:

$$\sum_{i=1}^{k_1} \sum_{j \in H_i^G} s_j \le \sum_{i=1}^{k_2} \sum_{j' \in R_i^*} s_{j'} + s_{\hat{j}} \tag{28}$$

$$Z_{\{L\}}^G = \sum_{i=1}^{k_1} \sum_{j \in H_i^G} s_j + \sum_{i=1}^{k_1} \sum_{j \in B_i^G} s_j$$
$$\Rightarrow Z_{\{L\}}^G \le \sum_{i=1}^{k_2} \sum_{j' \in R_i^*} s_{j'} + s_{\hat{j}} + \sum_{i=1}^{k_1} \sum_{j \in B_i^G} s_j \tag{29}$$

Finally, from (24) and (29), it will be concluded that

$$\begin{aligned} Z^G &= Z_{\{T\}}^G + Z_{\{L\}}^G \\ &\le Z_{\{T\}}^* + \sum_{i=1}^{k_2} \sum_{j \in R_i^G} s_j + \sum_{i=1}^{k_2} \sum_{j' \in R_i^*} s_{j'} + s_{\hat{j}} + \sum_{i=1}^{k_1} \sum_{j \in B_i^G} s_j \\ &= Z_{\{T\}}^* + Z_{\{L\}}^* + \sum_{i=1}^{k_2} \sum_{j' \in R_i^*} s_{j'} + s_{\hat{j}} \\ &\le 2Z^* \end{aligned} \tag{30}$$

This completes the proof.

5-2. Dynamic programming algorithm

Again, we propose a dynamic programming algorithm and convert it to an FPTAS by the technique of constructing the execution of an algorithm. Suppose that jobs are indexed according to the WSPT ordering. An optimal solution is composed of two groups of jobs; the first group includes tardy jobs with completion times no more than d_2, while the second group includes lost jobs with completion times greater than d_2.

In the following dynamic programming algorithm, jobs in the first group are continually scheduled from time zero and according to their order of indices, while lost jobs are scheduled from the end of the sequence to the beginning so that they complete exactly at time P_{sum}. The steps of this algorithm are as follows:

Algorithm DP2

```
1. Set $v_0 = \{[0,0]\}$
2. For each $k = \{1,2,\ldots,n\}$, consider all states $[t,f]$ in $v_{k-1}$
   o If $t + p_k \le d_2$, add the state $[t + p_k, f + w_k\,.(t + p_k)]$ to states space $v_k$
   o Add $[t, f + s_k]$ to states space $v_k$
   2.1. For all the states $[t,f] \in v_k$ with equal value for $t$, keep at most one state with minimum f.
   2.2. Remove states space $v_{k-1}$
3. Return the optimal solution $Z^* = \min_{[t,f]\in v_n} \{f\}$
```

The time complexity of this algorithm is $O(n.d_2)$ which can be verified through the same deduction as algorithm *DP1*.

5-3. FPTAS algorithm

This FPTAS is based on two phases. In the first phase, algorithm *MLCR* is used to determine an upper bound for problem $P3$ and, in the second phase, the execution of algorithm *DP2* is modified in order to reduce the number of states and running time. Recall that Z^G denotes the objective value returned by the algorithm *MLCR* for an instance of problem $P3$.

The FPTAS algorithm works on reduced state space $v_k^\#$ instead of v_k. To reduce the number of states in each iteration, we split the feasible interval $[0, Z^G]$ related to the second coordinate of state $[t,f]$ into L equal sub-intervals $I_m = [(m-1)\Delta, m\Delta]$ $,1 \le m \le L$ of length Δ. For each of the resulting sub-intervals I_m, we keep at most one state with the smallest value *t*. Given an arbitrary $\varepsilon > 0$, define

$$LB = \frac{Z^G}{2} \quad , \quad L = \left\lceil \frac{2n}{\varepsilon} \right\rceil, \Delta = \frac{Z^G}{L} \qquad (31)$$

Since $L = O(n/\varepsilon)$, the complexity of the FPTAS algorithm is obtained as $O(n^2/\varepsilon)$. This algorithm can be described as follows:

Algorithm FPTAS2

```
1. Set $v_o^\# = \{[0,0]\}$
2. For each $k = \{1,2,\ldots,n\}$, consider all states $[t,f]$ in $v_{k-1}^\#$
   o If $t + p_k \le d_2$, add the state $[t + p_k, f + w_k\,.(t + p_k)]$ to states space $v_k^\#$
   o Add $[t, f + s_k]$ to the states space $v_k^\#$
   2.1. For all the states $[t,f] \in v_k^\#$ with equal value for $t$, keep at most one state with minimum f.
   2.2. Let $[t,f]_m$ be a state in $v_k^\#$ such that $f \in I_m$ and t has the minimum value. Set $v_k^\# = \{[t,f]_m \mid 1 \le m \le L\}$
   2.3. Remove states space $v_{k-1}^\#$
3. Return the final solution $Z^\# = \min_{[t,f]\in v_n^\#} \{f\}$
```

Example 6. This example describes the steps of algorithm *FPTAS*2. Consider the data from Table 2. Suppose that $d_1 = 0$ and $d_2 = 15$; also, let $\epsilon = 0.5$. Algorithm *MLCR* generates sequence 1-3-2 with total penalty $Z^G = 118$. We have $LB = \frac{118}{2} = 59$, $L = \left\lceil \frac{6}{0.5} \right\rceil = 12$ and $\Delta = \frac{118}{12} = 9.83$. Jobs 1, 2, and 3 are selected, respectively, based on the WSPT ordering and are scheduled during the iterations. From step 1, we have $v_0^\# = [0,0]$. At the first iteration, we get $v_1^\# = \{[6,24], [0,46]\}$. Three states are generated by the second iteration making the state space $v_2^\# = \{[6,76], [11,123], [0,98]\}$. Job 3 is scheduled at the third iteration where five states are generated and, after removing states [15,121] and [9,125], the remaining state space is $v_3^\# = \{[6,118], [11,165], [0,140]\}$. So, the final solution is $Z^\# = 118$.

5-4. Worst-case analysis of algorithm FPTAS2

The worst-case analysis is based on comparing the execution of algorithms *DP2* and *FPTAS2*. First, a lemma is provided, and then, the main Theorem will be proved.

Lemma 2. Let $[t,f] \in v_k$ be an arbitrary state in *DP2*. Algorithm *FPTAS2* generates at least one state $[t^\#, f^\#]$ in $v_k^\#$, such that $t^\# \le t$ and $f^\# \le f + k\Delta$.

Proof. Similar to the proof of Lemma 1, this lemma is proved by induction on *k*. Again, two cases can be distinguished where, in the first case, $[t,f] = [t' + p_k, f' + w_k(t' + p_k)]$ and, in the second case, $[t,f] = [t', f' + s_k]$. The procedure is mainly the same as the proof of Lemma 1; however, the only difference is in Eqs. (15) and (17) that must be changed to Eqs. (32) and (33), respectively.

$$\begin{aligned}\mu &\le f'^{\#} + w_k.\left(t'^{\#} + p_k\right) + \varDelta \\ &\le f' + (k-1)\varDelta + w_k.(t' + p_k) + \varDelta \\ &= f + k\varDelta \qquad (32)\end{aligned}$$

$$\begin{aligned}\mu' &\le f'^{\#} + s_k + \varDelta \\ &\le f' + (k-1)\varDelta + s_k + \varDelta \\ &= f + k\varDelta \qquad (33)\end{aligned}$$

Theorem 6. Given an arbitrary $\varepsilon > 0$, algorithm *FPTAS2* outputs a sequence with $Z^{\#}$ penalty for problem $P3$ such that $Z^{\#} - Z^{*} \le \varepsilon\ Z^{*}$.

Proof. By definition, the optimal sequence can be related to a state $[t^{*}, f^{*}]$ in v_n. According to Lemma 2, algorithm *FPTAS2* generates a state $[t^{\#}, f^{\#}]$ in $\mathrm{v}_{\mathrm{n}}^{\#}$ such that $t^{\#} \le t^{*}$ and

$$\begin{aligned}f^{\#} &\le f + n\varDelta \\ &= f + n\frac{Z^G}{L} = f + n\frac{2\,.LB}{\lceil 2n/\varepsilon\rceil} \\ &\le f + \varepsilon\,.LB \\ &\le (1+\varepsilon)f \qquad (34)\end{aligned}$$

Since $t^{\#} \le t$ holds in all states of the FPTAS algorithm, it is clear that there always exists a feasible state in *FPTAS2* related to any feasible state generated by *DP2*. This completes the proof. ■

6. Conclusion

In this paper, we considered a less studied performance criteria for scheduling problems, named Tardy/Lost, and analyzed two of its special cases from the approximability point of view. We showed that many of popular objective functions for scheduling problems, such as weighted tardiness, late work, and tardiness with rejection, are special cases for the proposed Tardy/Lost performance criteria and, hence, the results of this study are applicable to them. Two polynomial-time approximation algorithms were developed for the problem of minimizing *TL* measure with common due dates on a single machine. The worst-case ratio bounds of these algorithms were proved, and their tightness was also shown through some numerical examples.

The first special case occurs when all jobs have the same tardiness weights. For this case, after proving the complexity of the problem, an FPTAS was developed that runs in $O(n^3/\varepsilon)$ time. In the second special case, none of the jobs becomes early under any arbitrary sequence. We have developed a 2-approximation algorithm for this case and verified its tightness. Next, a dynamic programming algorithm was developed and converted to an FPTAS of $O(n^2/\varepsilon)$ time complexity.

In future research, we aim to adjust our scheme to handle the problem with any fixed number of distinct due dates. Another interesting research goal is to study the Tardy/Lost measure in more complex scheduling environments such as parallel machine or flow shops and extend the results. The development of better approximation algorithms and FPTASs is also a challenging subject.

References

[1] J. Yuan, "The NP-hardness of the single machine common due date weighted tardiness problem," *Systems Science and Mathematical Sciences,* Vol. 5, (1992), pp. 328-333.

[2] D. Shabtay, N. Gaspar, and M. Kaspi, "A survey on offline scheduling with rejection," *Journal of Scheduling,* Vol. 16, (2013), pp. 3-28.

[3] G. J. Woeginger, "When does a dynamic programming formulation guarantee the existence of an FPTAS?," *INFORMS Journal on Computing,* Vol. 12, (2000), pp. 57-74.

[4] R. B. Kethley and B. Alidaee, "Single machine scheduling to minimize total weighted late work: a comparison of scheduling rules and search algorithms," *Computers and Industrial Engineering,* Vol. 43, (2002), pp. 509-528.

[5] D. W. Engels, D. R. Karger, S. G. Kolliopoulos, S. Sengupta, R. N. Uma, and J. Wein, "Techniques for Scheduling with Rejection," *Journal of Algorithms,* Vol. 49, (2003), pp. 175-191.

[6] D. Shabtay, N. Gaspar, and L. Yedidsion, "A bicriteria approach to scheduling a single machine with job

rejection and positional penalties," *Journal of Combinatorial Oprimization,* Vol. 23, (2012), pp. 395-424.

[7] S. A. Slotnick, "Order acceptance and scheduling: A taxonomy and review," *European Journal of Operational Research,* Vol. 212, (2011), pp. 1-11.

[8] J. K. Lenstra, A. H. G. Rinnoy-Kan, and P. Brucker, "Complexity of machine scheduling problems," *Annals of Discrete Mathematics,* Vol. 1, (1977), pp. 343-362.

[9] S. G. Kolliopoulos and G. Steiner, "Approximation algorithms for minimizing the total weighted tardiness on a single machine," *Theoretical Computer Science,* Vol. 355, (2006), pp. 261-273.

[10] T. C. E. Cheng, C. T. Ng, J. J. Yuan, and Z. H. Liu, "Single machine scheduling to minimize total weighted tardiness," *European Journal Operations Research,* Vol. 165, (2005), pp. 423-443.

[11] 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," presented at the The 15th International Computing and Combinatorics conference, Niagara Falls, (2009).

[12] E. L. Lawler and J. M. Moore, "A functional equation and its application to resource allocation and sequencing problems," *Management Science,* Vol. 16, (1969), pp. 77-84.

[13] Y. Fathi and H. W. L. Nuttle, "Heuristics for the Common due date weighted tardiness problem," *lIE Transactions,* Vol. 22, (1990), pp. 215-225.

[14] 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, (2006), pp. 230-238.

[15] K. Kianfar and G. Moslehi, "A note on “Fully polynomial time approximation scheme for the total weighted tardiness minimization with a common due date”," *Discrete Applied Mathematics,* Vol. 161, (2013), pp. 2205-2206.

[16] J. Du and J. Y.-T. Leung, "Minimizing total tardiness on one machine is NP-hard," *Mathematics of Operations Resarch,* Vol. 15, (1990), pp. 483-495.

[17] E. L. Lawler, "A psedupolynomial algorithm for scheduling jobs to minimize total tardiness," *Annals of Discrete Mathematics,* Vol. 1, (1977), pp. 331-342.

[18] E. L. Lawler, "A fully polynomial approximation scheme for the total tardiness problem," *Operations Research Letters,* Vol. 1, (1982), pp. 207-208.

[19] C. Koulamas, "A faster fully polynomial approximation scheme for the single-machine total tardiness problem," *European Journal of Operational Research,* Vol. 193, (2009), pp. 637-638.

[20] M. Y. Kovalyov, "Improving the complexities of approximation algorithms for optimization problems," *Operations Research Letters,* Vol. 17, (1995), pp. 85-87.

[21] F. Della Croce, A. Grosso, and V. T. Paschos, "Lower bounds on the approximation ratios of leading heuristics for the single-machine total tardiness problem," *Journal of Scheduling,* Vol. 7, (2004), pp. 85-91.

[22] 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, (2002), pp. 415-428.

[23] C. N. Potts and L. N. Van Wassenhove, "Single machine scheduling to minimize total late work," *Operations Research,* Vol. 40, (1992)pp. 586-595.

[24] C. N. Potts and L. N. Van Wassenhove, "Approximation algorithms for schrduling a single machine to minimize total late work," *Operations Research Letters,* Vol. 11, (1992), pp. 261-266.

[25] M. Sterna, "Late work minimization in a small flexible manufacturing system," *Computers and Industrial Engineering,* Vol. 52, (2007), pp. 210-228.

[26] E. Pesch and M. Sterna, "Late work minimization in flow shop by a genetic algorithm," *Computers and Industrial Engineering,* Vol. 57, (2009), pp. 1202-1209.

[27] M. Sterna, "Dominancec relations for two-machine flow-shop problem with late work criterion," *Bulletin of the Polish Academy of Sciences,* Vol. 55, (2007), pp. 59-69.

[28] J. Blazewicz, E. Pesch, M. Sterna, and F. Werner, "Methaheuristic approaches for the two-machine flow-shop problem with weighted late work criterion and common due date," *Computers and Operations Research,* Vol. 35, (2008), pp. 574-599.

[29] J. Ren, Y. Zhang, and J. Sun, "The NP-hardness of minimizing the total late workk on an unbounded batch machine," *Asia-Pacific Journal of Operational Research,* Vol. 26, (2009), pp. 351-363.

[30] B. Chen, C. N. Potts, and G. J. Woeginger, "A review of machine scheduling," in *Handbook of combinatorial optimization*, D. Z. Du and P. M. Pardalos, Eds., ed Boston: Kluwer Academic Publishers, (1998), pp. 21-169.

[31] J. Y. T. Leung, "Minimizing total weighted error for imprecise computation tasks and related problems," in *Handbook of scheduling: algorithms, models and performance analysis*, J. Y. T. Leung, Ed., ed Boca Raton: CRC Press, (2004), pp. 34.1-16.

[32] M. Sterna, "A survey of scheduling problems with late work criteria," *Omega,* Vol. 39, (2011), pp. 120-129.

[33] K. Kianfar, G. Moslehi, and A. S. Nookabadi, "Exact and heuristic algorithms for minimizing Tardy/Lost penalties on a single-machine scheduling problem," *Computational and Applied Mathematics,*(2016).

[34] K. Kianfar, G. Moslehi, and A. S. Nookabadi, "An approximation algorithm and FPTAS for Tardy/Lost minimization with common due dates on a single machine," *Journal of Industrial and Systems Engineering,* Vol. 9, (2016), pp. 1-19.

[35] G. Moslehi and K. Kianfar, "Approximation Algorithms and an FPTAS for the Single Machine Problem with Biased Tardiness Penalty," *Journal of Applied Mathematics,* (2014).

[36] M. Pinedo, *Scheduling: theory, algorithms and systems*. New Jersey: Prentice Hall, (1995).

[37] E. L. Lawler, "On scheduling problems with deferral costs," *management Science,* Vol. 11, (1964), pp. 280-288.

[38] H. G. Kahlbacher, "Scheduling with monotonous earliness and tardiness penalties," *European Journal of Operational Research,* Vol. 64, (1993), pp. 258-277.

[39] A. Federgruen and G. Mosheiov, "Greedy heuristics for single-machine scheduling problems with general earliness and tardiness costs,"

Operations Research Letters, Vol. 16, (1994), pp. 199-208.

[40] P. Baptiste and R. Sadykov, "On scheduling a single machine to minimize a piecewise linear objective function: A compact MIP formulation," *Naval Research Logistics,* Vol. 56, (2009), pp. 487-502.

[41] X. Zhou and X. Cai, "General stochastic single-machine scheduling with regular cost functions," *Mathematical and Computer Modelling,* Vol. 26, (1997), pp. 95-108.

[42] D. Shabtay, "Due date assignment and scheduling a single machine with a general earliness/tardiness cost function," *Computers and Operations Research,* Vol. 35, (2008), pp. 1539-1545.

[43] J. A. Ventura and S. Rahhakrishnan, "Single machine scheduling with symmetric earliness and tardiness penalties," *European Journal of Operational Research,* Vol. 144, (2003), pp. 598-612.

[44] P. Baptiste, D. Rebaine, and M. Zouba, "FPTAS for the two identical parallel machine problem with a single operator under the free changing mode," *European Journal of Operational Research,* Vol. 256, (2017), pp. 55–61.

[45] J. Dong, W. Tong, T. Luo, and X. Wang, "An FPTAS for the parallel two-stage flowshop problem," *Theoretical Computer Science,* Vol. 657, (2017), pp. 64–72.

[46] R. Rudek, "Scheduling on parallel processors with varying processing times," *Computers & Operations Research,* Vol. 81, (2017), pp. 90–101.

[47] Y. Yin, S. R. Cheng, T. C. E. Cheng, D. J. Wang, and C. C. Wu, "Just-in-time scheduling with two competing agents on unrelated parallel machines," *Omega,* Vol. 63, (2017), pp. 41-47.

[48] G. Wachtel and A. Elalouf, "Using the “Floating Patients” method to balance crowding between the hospital emergency department and other departments," *Computers & Industrial Engineering,* Vol. 110, (2017). pp. 289–296.

[49] O. Ibarra and C. E. Kim, "Fast approximation algorithms for the knapsack and sum of subset problems," *problems, Journal of the ACM,* Vol. 22, (1875), pp. 463-468.

[50] 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, (2012), pp. 67-74.

[51] 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, (2010), pp. 39-47.

[52] 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, (2010), pp. 90-98.

[53] 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,* (2009).

[54] 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, (2009), pp. 1708-1712.

Follow This Article at The Following Site

Kianfar K, Moslehi G. Minimizing a General Penalty Function on a Single Machine via Developing Approximation Algorithms and FPTASs. IJIEPR. 2017; 28 (3) :221-240 DOI: 10.22068/ijiepr.28.3.221
URL: http://ijiepr.iust.ac.ir/article-1-710-en.html