Journal of Industrial and Systems Engineering
Vol. 9, No. 2, pp 1-19
Spring (April) 2016

An approximation algorithm and FPTAS for Tardy/Lost minimization with common due dates on a single machine

Kamran Kianfar[1], Ghasem Moslehi[2]*, Ali Shahandeh Nookabadi[2]
[1]*Faculty of Engineering, University of Isfahan, Isfahan, Iran*
[2]*Department of Industrial and Systems Engineering, Isfahan University of Technology, Isfahan, Iran*
k.kianfar@in.iut.ac.ir, moslehi@cc.iut.ac.ir, ali-nook@cc.iut.ac.ir

Abstract

This paper addresses the Tardy/Lost penalty minimization with common due dates on a single machine. According to this performance measure, if the tardiness of a job exceeds a predefined value, the job will be lost and penalized by a fixed value. Initially, we present a 2-approximation algorithm and examine its worst case ratio bound. Then, a pseudo-polynomial dynamic programming algorithm is developed. We show how to transform the dynamic programming algorithm to an FPTAS using the technique of "structuring the execution of an algorithm" and examine the time complexity of our FPTAS.

Keywords: Single machine scheduling, Tardy/Lost penalty, Common due date, approximation algorithm, FPTAS

1- Introduction

In this paper, we study a single machine scheduling problem related to minimizing total Tardy/Lost penalties and common due dates. Every job $i(1 \leq i \leq n)$ has a processing time, p_i, and a weight tardiness factor, w_i. The jobs have two common due dates, namely, $d_1$and d_2. In case 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; and finally, the job will be lost and a fixed amount of penalty, s_i, will be assigned if it is completed after the second due date, d_2. Based on this formulation, we can define the Tardy/Lost objective function as shown in Eq. (1), where $C_i$is the completion time of job *i* in a sequence.

$$Z_i = \begin{cases} w_i max\{0, C_i - d_1\} & if \quad C_i \leq d_2 \\ s_i = w_i(d_2 - d_1) & if \quad C_i > d_2 \end{cases} \tag{1}$$

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

*Corresponding author.

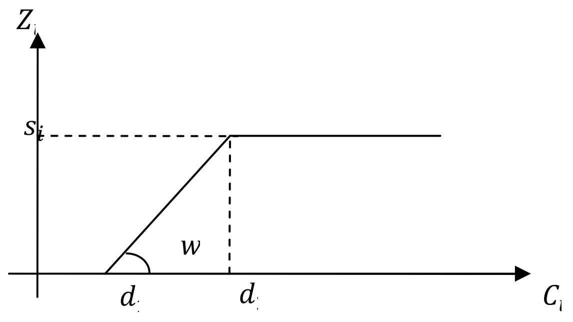

Fig. 1. Tardy/Lost penalty function with common due dates

We assume that all processing times and due dates are integers, 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 $1|d_i = d|TL$, where TL indicates the Tardy/Lost performance criteria. Yuan (1992) has shown that the tardiness minimization problem with a common due date (CD_T) is NP-hard. Since CD_T is a special case of Tardy/Lost penalty function evaluated in this paper, this function is also assumed to be NP-hard.

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

From a practical point of view, the 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 is considered; the order will be penalized if its delivery time exceeds the first due date. The penalty is increased in proportion to the delivery time until the second due date is reached. If the delivery occurs later than the second due date, the order becomes lost and gets the 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 a job i can be defined as $(d_2 - d_1)w_i$. Order acceptance assumption has been investigated widely in the literature. Engels et al. (2003) investigated the problem of minimizing weighted completion times and rejection penalties and developed some approximation algorithms. The survey papers by Slotnick (2011) and Shabtay (2013) have addressed a number of scheduling problems with order acceptance.

The performance measure we consider in this study is a kind of regular measure which is continuous and non-decreasing in the completion times of jobs. With such performance measures, jobs are penalized only for being tardy, while all inventory costs due to early completions are ignored. Pinedo(1995) indicates that in practice, the penalty function associated with a scheduling problem may follow from a function in which early jobs are assigned no penalty and those finished after their due dates are assigned a penalty that is increased at a given rate. Within the penalty function, the job reaches a point where the penalty assignment is changed and increased at a much slower pace. The function identified by Pinedo(1995) is general; however, two more specific functions that react similarly are the deferral cost function (Lawler, 1964) and the late work criterion (Potts and Vav Wassenhove, 1992a, Potts and Vav Wassenhove, 1992b).

Assume that we have an approximation algorithm that always returns near-optimal solution whose cost is at most a factor of ρ away from the optimal cost, where $\rho > 1$ is a real number. In minimization problems the near-optimal cost is at most a multiplicative factor of ρ above the optimum. Such an approximation algorithm is called a ρ-approximation algorithm. A family of (1+ε)-approximation algorithms over all $\varepsilon > 0$with polynomial running times is called a polynomial time approximation scheme, or PTAS for short. If the time complexity of a PTAS is also polynomially bounded in $1/\varepsilon$, then it is called a fully polynomial time approximation scheme, or FPTAS for short (Woeginger, 2000).

Deferral cost functions have been studied by Kahlbacher(1993). He considered general penalty functions to be monotonous with respect to absolute lateness. He also examined several specific cases of the penalty function for situations in which machine idle times are allowed (the unconstrained problem) or not allowed (the constrained problem). Using the "V-shape property", he constructed an optimal schedule with a pseudo-polynomial algorithm and extended it to an FPTAS of the order$O(n^3/\varepsilon)$, where ε denotes the precision of the approximation solution.

Federgruen and Mosheiov (1994) considered a class of single machine scheduling problems with a common due date and general earliness and tardiness penalties. In that study, some polynomial greedy algorithms were proposed for generating schedules and a small optimality gap was illustrated through numerical examples. For convex cost structures, they also established that the worst case optimality gap was bounded by 0.36 if the due date was non-restrictive. Baptiste and Sadykov (2009) considered the objective of minimizing a piecewise linear function. This class of functions is very large and can also be used to model some classical objective functions, i.e., total (weighted) completion time, total (weighted) tardiness, and (weighted) number of tardy jobs. They introduced a new Mixed Integer Programming (MIP) model based on time interval decomposition. This MIP model was closely related to the classical time-indexed MIP formulation, but used far fewer variables and constraints.

The research by Ventura and Radhakrishnan (2003) focused on scheduling jobs with varying processing times and distinct due dates on a single machine. Zhou and Cai (1997) examined two types of regular performance measures, the total cost and the maximum cost, with general cost functions. They studied a stochastic scheduling model on a single machine where processing times were random variables and the machine was subject to stochastic breakdowns. In the paper by Shabtay (2008), two continuous and non-decreasing objective functions were considered. They included penalties due to earliness, tardiness, the number of tardy jobs, and due date assignments.

In the present study, we simply observe that the Tardy/Lost penalty function can be converted to a late work criterion that estimates the quality of a solution on the basis of the duration of the late parts of jobs. This conversion is accomplished by setting $w_i = 1 \quad (\forall i = 1, \ldots, n)$and $d_{2i} = d_{1i} + p_i$, where d_{1i} and d_{2i} are the first and second due dates for each job i. The results concerning late work scheduling problems are partially presented in Chen et al. (1998) and Leung (2004), but Sterna (2011) offers the first complete review of the topic.

A number of researchers have addressed the problem of minimizing the total late work criterion on a single machine. Potts and Van Wassenhove (1992b) proposed a polynomial time algorithm based on the similarity between tardiness and late work parameters. In another study (Potts and Vav Wassenhove, 1992a), they developed a branch-and-bound algorithm for a problem forming the core of a family of approximation algorithms based on truncated enumeration. References (Sterna, 2007b, Pesch and Sterna, 2009, Sterna, 2007a, Blazewicz et al., 2008, Ren et al., 2009) may be consulted for other studies devoted to the late work criterion.

Kathley and Alidaee (2002) 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 "*modified TWLW*", which was very similar to the Tardy/Lost penalty function we have considered in this study. Kianfar and Moslehi (2013, 2014)studied some tardiness-based objective functions on a single machine with common due dates. They developed approximation algorithms as well as FPTASs for the problems. The worst-case ratio bounds are also proved in some cases.

As the Tardy/Lost penalty function is a general form of the late work criterion, most practical applications of late work can be equally defined by this penalty function. Some of the applications in problems arise in control systems (Blazewicz, 1984, Potts and Vav Wassenhove, 1992b), production and planning process (Sterna, 2000, sterna, 2006), agriculture (Alminana et al., 2010), land cultivation processes (Blazewicz et al., 2004, Sterna, 2000), and processes of planning tests for prototypes or software validation (sterna, 2006).

The rest of this paper is organized as follows. In Section 2, we propose an approximation algorithm and examine its worst case ratio bound. Section 3 describes a pseudo-polynomial dynamic programming algorithm that, in Section 4, will be converted to an FPTAS using the technique of structuring the execution of an algorithm. Concluding remarks will be presented in Section 5.

2- MWR[1] approximation algorithm

Here, a 2-approxiamtion algorithm is developed for problem $1|d_i = d|TL$. By using a numerical example in a later stage, it will be shown that the worst case ratio bound of 2 is tight for this problem. We refer to this algorithm as MWR and designate the sequence it generates as *G*. We adopt the notation $g_{[i]}$to represent the job in *i*th position in the sequence$G = (g_{[1]}, g_{[2]}, \ldots, g_{[n]})$. Heuristic *G* requires at most *n* iterations, while, in each iteration *k*, the (*n*-*k*+1)th element of *G*, $g_{[n-k+1]}$ will be determined. Furthermore, if, during an iteration, there exists an unscheduled job filling the remaining tardiness period with the minimum tardiness weight, the relevant sequence (sequence $\hat{G}$) will be saved in addition to the main sequence *G*. The algorithm returns the best of the two sequences *G* and $\hat{G}$ as the final result.

Let Z^Gdenote the penalty of sequence *G* and $Z_{g_{[r,n]}}$denote the total penalty of jobs from position *r*to*n* in this sequence. Suppose that all jobs *i*are sorted and indexed according to the non-decreasing order of w_i/p_i ratios. In the case of a tie, the job with the smallest processing time should come first. The steps of the algorithm are as follows:

Step 1. Let $U = \{1,2,..,n\}$ be a set of unscheduled jobs sorted according to their indices (according to the non-decreasing order of w_i/p_i ratios). Set $C_{[n]} = P_{sum} = \sum_{i=1}^{n} p_i$and $r = n$.

Step 2. Define $\hat{U} = \{i \in U \mid p_i \geq C_{[n]} - d_1\}$. If $\hat{U}$ is empty, then $\hat{Z}_{best} = \infty$; else, select a job $\hat{\imath}$with the minimum tardiness weight from $\hat{U}$. Let $\hat{g}_{[n]} = \hat{\imath}$and $\hat{Z}_{best} = w_{\hat{\imath}}(min\{C_{[n]}, d_2\} - d_1)$.

Step 3. Define the first job in *U* as job *k;* also, set $U = U/\{k\}$, $g_{[r]} = k$and $C_{[r-1]} = C_{[r]} - p_k$.

Step 4. If $\hat{U} = \{i \in U \mid p_i \geq C_{[r-1]} - d_1\}$ is not empty, select a job $\hat{\imath}$ with the minimum tardiness weight from $\hat{U}$. Calculate $\hat{Z} = w_{\hat{\imath}}(min\{C_{[r-1]}, d_2\} - d_1)$. If $\hat{Z} + Z_{g[r,n]} < \hat{Z}_{best}$ then we have $\hat{Z}_{best} = \hat{Z} + Z_{g[r,n]}$ and $\hat{g}_{[i]} = g_{[i]} \ \ \forall i = r, \ldots, n$and $\hat{g}_{[r-1]} = \hat{\imath}$.

Step 5. If $C_{[r-1]} > d_1$, then $r = r - 1$ and go back to Step 3; else, go to Step 6.

Step 6. Put unscheduled jobs at the beginning of each of the sequences *G*and $\hat{G}$.

Step 7. If $Z^G \leq Z^{\hat{G}}$, then return sequence $G = (g_{[1]}, g_{[2]}, \ldots, g_{[n]})$; else, return sequence $\hat{G} = (\hat{g}_{[1]}, \hat{g}_{[2]}, \ldots, \hat{g}_{[n]})$.

It can be easily verified that *G* has the same order as WSPT and $\hat{G}$is the best sequence obtained by filling the remaining tardiness period with one job in each iteration. This algorithm includes a simple sorting of *n* elements and hence, its time complexity is$O(n\ logn)$.

Now, we provide a numerical example to illustrate how the MWR algorithm works. Later, we will use a theorem to prove that the worst case ratio bound of this algorithm is equal to 2.

Example 1. Consider a problem $1|d_i = d|TL$with 4 jobs described in Table 1. Set $d_1 = 10$and $d_2 = 12$.

[1]- Minimum Weight Ratio

Table 1. Parameters of jobs in Example 1

Job i		
1	5	2
2	4	3
3	6	5
4	15	16

At the beginning, $\hat{U}$ is empty and $\hat{Z}_{best} = \infty$. The following table summarizes the results of algorithm MWR for this problem.

Table 2. Summary of applying MWR algorithm for Example 1

r=4	30	{4}	32	4	36	
r=3	25	{4}	32	10	36	
r=2	21	{4}	32	20	36	
r=1	15					(

From $Z^G = 52$and $Z^{\hat{G}} = \hat{Z}_{best} = 36$, algorithm MWR returns sequence $\hat{G}$ as the approximate solution.

Theorem 1. Algorithm MWR gives a 2-approximation for problem$1|d_i = d|TL$.

Proof. See the Appendix. ■

In the following example, we show that the worst case ratio bound of algorithm MWR for problem $1|d_i = d|TL$is a tight bound.

Example 2. Consider the problem $1|d_i = d|TL$ with $n > 2$jobs such that $d_1 = n - 1$and $d_2 = 2n - 1$. The parameters are those given in Table 3.

Table 3. Parameters of jobs in Example 2

Job i		
1	*n*	*n*+1/*n*
2 *to n*	1	1+1/*n*

Algorithm MWR returns the sequence $(2,3,\ldots,n,1)$ with a penalty value of $n^2 + 1$while the optimal sequence is $(1,2,\ldots,n)$, generating the penalty $1/2\,n^2 + 2n - 1/2$. Thus,

$$\frac{Z^G}{Z^*} = \frac{n^2 + 1}{\frac{1}{2}n^2 + 2n - \frac{1}{2}} \quad \Rightarrow \quad \lim_{n\to\infty} \frac{Z^G}{Z^*} = 2 \tag{2}$$

3- Dynamic programming algorithm

Suppose that jobs are indexed according to the non-decreasing order of $p_i/w_i$and ties are broken by first selecting a job with the minimumw_i. An optimal solution is composed of three groups of jobs. The first group includes early jobs with completion times less thand_1. The second group includes the tardy jobs with completion times between $d_1$and d_2, which must be scheduled according to their reverse order of indices (the first tardy job, called straddling job, may not adopt this order). Lost jobs with completion times greater than d_2 are placed in the third group by an arbitrary order.

The problem can be optimally solved by applying the following dynamic programming algorithm. In this algorithm, each state in the state space $v_k^{(\alpha,C_\alpha)}$ shows a particular sequence for first k jobs excluding the straddling one, where the straddling job α and its completion time, C_α, are predefined. We use the vector (t_1, t_2, f) to denote states. Variables t_1 and t_2 show the sum of processing times for jobs in the first and second groups, respectively, and f is the total penalty of the corresponding partial sequence.

In each iteration, the algorithm selects a job α as the straddling and fixes its completion time, C_α, from all possible values within the scheduling horizon. Jobs in the first group, early jobs, are continually scheduled from time zero (see Fig. 2*b*) while tardy jobs are scheduled by starting from the completion time of the straddling job (see Fig. 2*c*). Finally, lost jobs are scheduled in such a way that they are completed at time $P_{sum} = \sum_{i=1}^{n} p_i$ (see Fig. 2*d*).

Let Z^*_{α,C_α} denote the optimal objective value subject to the condition in which job α is selected as the straddling one with the completion time C_α. This algorithm can be described as follows.

Algorithm DP

```
Step 1. For each $\alpha = \{1,2,\dots,n\}$ and each $C_\alpha \in [d_1 + 1, d_1 + p_\alpha]$
    1.1. Set $v_0^{(\alpha,C_\alpha)} = \{(0,0,0)\}$
    1.2. For each $k = \{1,2,\dots,\alpha-1,\alpha+1,\dots,n\}$, consider all the states $(t_1, t_2, f)$ in $v_{k-1}^{(\alpha,C_\alpha)}$
        o If $t_1 + p_k \le C_\alpha - p_\alpha$, then add the state $(t_1 + p_k, t_2, f)$ to the state space $v_k^{(\alpha,C_\alpha)}$
        o If $C_\alpha + t_2 + p_k \le d_2$, then add the state $\left(t_1, t_2 + p_k, f + w_k(C_\alpha + t_2 + p_k - d_1)\right)$ to the state space $v_k^{(\alpha,C_\alpha)}$
        o Add $(t_1, t_2, f + s_k)$to the state space $v_k^{(\alpha,C_\alpha)}$
    1.3. For all states $(t_1, t_2, f) \in v_k^{(\alpha,C_\alpha)}$ with equal values for $t_1$and $t_2$, keep at most one state having the minimum value of $f$.
    1.4. Remove the state space $v_{k-1}^{(\alpha,C_\alpha)}$
    1.5. Set $Z^*_{\alpha,C_\alpha} = \min_{[t_1,t_2,f]\in v_{n-1}^{(\alpha,C_\alpha)}} \{f + w_\alpha.(min\{C_\alpha, d_2\} - d_1)\}$
Step 2. Return the optimal solution $Z^* = \min_{\alpha,C_\alpha} Z^*_{\alpha,C_\alpha}$
```

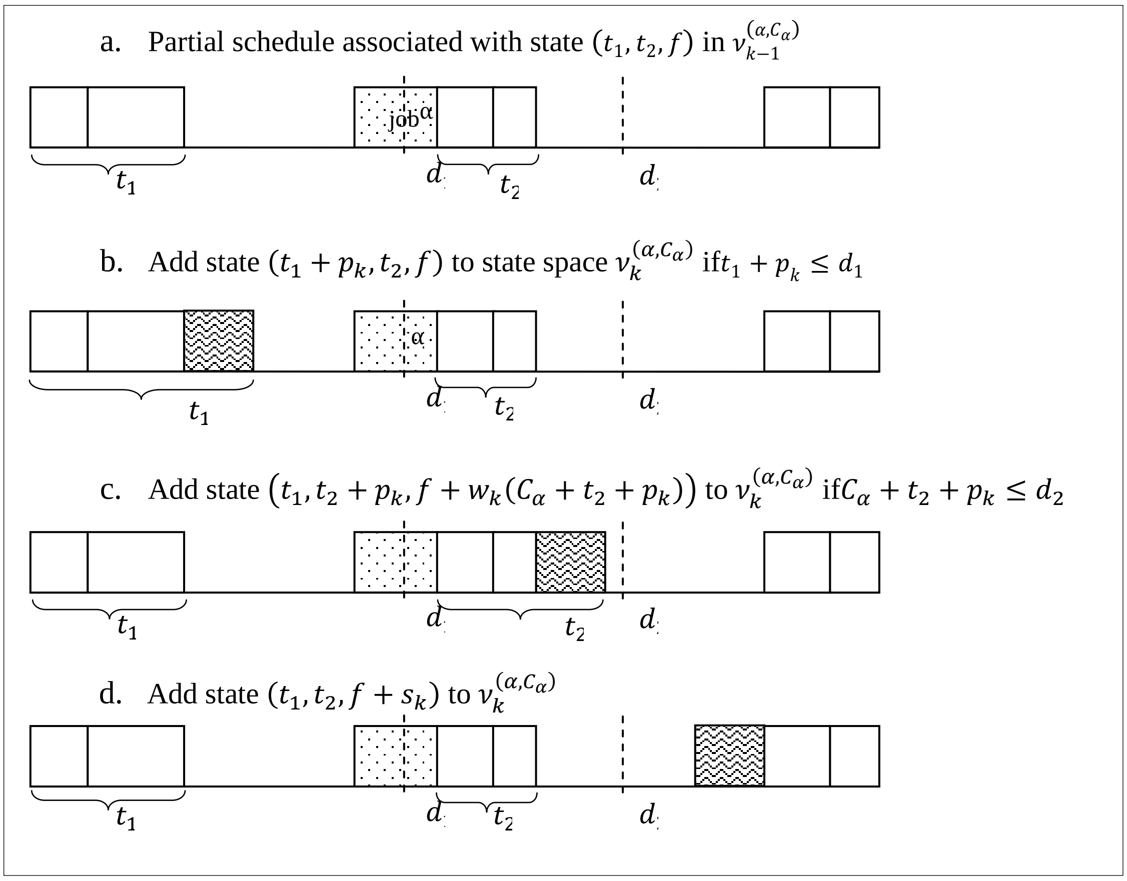
a. Partial schedule associated with state (t_1, t_2, f) in $v_{k-1}^{(\alpha,C_\alpha)}$
job α
t_1
d
t_2
d
b. Add state $(t_1 + p_k, t_2, f)$ to state space $v_k^{(\alpha,C_\alpha)}$ if$t_1 + p_k \leq d_1$
α
t_1
d
t_2
d
c. Add state $\big(t_1, t_2 + p_k, f + w_k(C_\alpha + t_2 + p_k)\big)$ to $v_k^{(\alpha,C_\alpha)}$ if$C_\alpha + t_2 + p_k \leq d_2$
t_1
d
t_2
d
d. Add state $(t_1, t_2, f + s_k)$ to $v_k^{(\alpha,C_\alpha)}$
t_1
d
t_2
d

Fig. 2 Description of the dynamic programming algorithm

To calculate the time complexity of this algorithm, as all input parameters are integers, we can restrict the number of states in each $v_k^{(\alpha,C_\alpha)}$by $d_1(d_2 - d_1)$. This is because variables t_1 and $t_2$can at most take d_1 and $d_2 - d_1$ different values, respectively. Also, in each iteration and for each combination of $t_1$and t_2, we keep at most one state with the smallest value of *f*.

Let P_{max} be the maximum processing time of jobs. The running time of Step 1.2.is proportional to $\sum_{k=1}^{n-1} \left| v_k^{(\alpha,C_\alpha)} \right|$ and hence, it is $nd_1(d_2 - d_1)$. Similarly, it can be shown that the complexity of step 1.5 is $O\big(d_1(d_2 - d_1)\big)$. Step 1 iterates at most nP_{max}times by selecting different α and C_α values and has the complexity of $O\big(n^2 P_{max} d_1(d_2 - d_1)\big)$. Finally, as Step 2 needs $O(nP_{max})$ time, the total complexity of this algorithm will be obtained as $O\big(n^2 P_{max} d_1(d_2 - d_1)\big)$. Alternatively, we can write the complexity of this algorithm as $O\big(nP_{sum} d_1(d_2 - d_1)\big)$ because the maximum number of iterations created by selecting the values of α and C_α can also be considered as P_{sum}instead of nP_{max}.

4- FPTAS algorithm

This FPTAS is based on two phases. In the first phase, algorithm MWR is used to determine an upper bound for problem $1|d_i = d|TL$and in the second phase, the execution of the algorithm DP is modified in order to reduce the number of states and the running time. One common way used 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 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

(1875) for solving the knapsack problem and it has been extended to numerous scheduling problems (see (Shabtay et al., 2012, Shabtay and bensoussan, 2010, Ji and Cheng, 2010, Steiner and Zhang, 2009, Kacem and Mahjoub, 2009)).

Let Z_H denote the objective value returned by algorithm MWR for an instance of the problem $1|d_i = d|TL$ and ε be the maximum acceptable error for the FPTAS. According to the technique of structuring the execution of an algorithm, we should restrict all factors that allow the size of the state space to grow in an uncontrolled way. These factors in our algorithm are the variables t_2 and fin the states (t_1, t_2, f) as well as the completion time of the straddling job.

By selecting a job α as the straddling, we split its completion time interval $C_\alpha \in [d_1 + 1 , d_1 + p_\alpha]$ into L_1 sub-intervals $[[d_1 + (\Delta_1)^{y-1}], [d_1 + (\Delta_1)^{y}]]$, where $\Delta_1 = 1 + \varepsilon/3$ and $y = 1,2,\dots, L_1$. The maximum number of these sub-intervals is

$$L_1 = \left\lceil log_{\Delta_1}^{p_\alpha} \right\rceil \leq \left\lceil log_{\Delta_1}^{P_{max}} \right\rceil = \lceil lnP_{max}/ln\Delta_1 \rceil \leq \lceil (1 + 3/\varepsilon) lnP_{max} \rceil \tag{3}$$

So, we have $L_1 = O(lnP_{max}/\varepsilon)$. The last inequality holds since for all $z \geq 1$, we have $ln\, z \geq (z-1)/z$ based on the Taylor expansion of $ln\, z$. Therefore,

$$\frac{1}{ln(\Delta_1)} \leq \frac{\Delta_1}{\Delta_1 - 1} = \frac{1 + \varepsilon/3}{1 + \varepsilon/3 - 1} = 1 + 3/\varepsilon \tag{4}$$

To restrict the number of different values for t_2, set $LB = Z_H/2$ and $\Delta_2 = (\varepsilon . LB)/(3n)$. Let there be m distinct values among tardiness weights w_i of n jobs. Sort these values in a decreasing order such that $w_{\pi_1} > w_{\pi_2} > \ldots > w_{\pi_m}$and create the permutation $\Pi = (\pi_1, \pi_2, \ldots, \pi_m)$. Split the interval $\left[0, Z_H/w_{\pi_m}\right]$ into m intervals

$$I_1 = \left[0, \frac{Z_H}{w_{\pi_1}}\right] \;, \; I_2 = \left[\frac{Z_H}{w_{\pi_1}}, \frac{Z_H}{w_{\pi_2}}\right] \;, \; \ldots \;, \qquad I_m = \left[\frac{Z_H}{w_{\pi_{m-1}}}, \frac{Z_H}{w_{\pi_m}}\right] \tag{5}$$

Then, split each interval I_j into the sub-intervals of the length $\Delta_2/(n \, . w_{\pi_j})$. It may appear that the last of the sub-intervals of an interval I_j is shorter than $\Delta_2/(n\, . w_{\pi_j})$. The maximum number of sub-intervals created by the above procedure is

$$\begin{aligned} L_2 &= \frac{Z_H}{w_{\pi_1}} \times \frac{n. w_{\pi_1}}{\Delta_2} + \sum_{i=2}^{m} \left[\frac{Z_H}{w_{\pi_i}} - \frac{Z_H}{w_{\pi_{i-1}}}\right] \frac{n\, . w_{\pi_i}}{\Delta_2} \\ &\leq n\, . \sum_{i=1}^{m} \left(\frac{Z_H}{w_{\pi_i}} . \frac{w_{\pi_i}}{\Delta_2}\right) \leq m\, . n\, . \frac{Z_H}{\Delta_2} \leq n^2\, . \frac{Z_H}{\Delta_2} \end{aligned} \tag{6}$$

and since $Z_H/\Delta_2 = 2LB/\Delta_2 \leq 6n/\varepsilon$, we conclude that $L_2 = O(n^3/\varepsilon)$.

In order to make the number of distinct values of f controllable, we split the related interval $[0, Z_H]$ into L_3 equal sub-intervals $I'_m = [(m-1)\Delta_3, m\Delta_3]$ of the length Δ_3 ,where$1 \leq m \leq L_3$. Considering $L_3 = \lceil 2n/\varepsilon \rceil$and $\Delta_3 = Z_H/L_3$, we get $L_3 = O(n/\varepsilon)$.

Next, the FPTAS algorithm will be designed by means of bounding the number of different values that each of variablesC_α, t_2 and fcan take in the DP algorithm. Also, the time complexity of this FPTAS as well as its worst case ratio bound will be discussed.

This FPTAS works on a reduced state space $v_k^{(\alpha, C'_\alpha)\#}$ instead of the main state space $v_k^{(\alpha, C_\alpha)}$ and returns the approximate solution of $Z^\#$. The algorithm can be described as follows.

Algorithm FPTAS

```
Step 1. For each $\alpha = \{1,2,\ldots,n\}$ and each $C'_\alpha = \lfloor d_1 + (\Delta_1)^y \rfloor$ , $y = 0,1,\ldots,L_1$
    1.1. Set $\nu_0^{(\alpha,C'_\alpha)\#} = \{(0,0,0)\}$
    1.2. For each $k = \{1,2,\ldots,\alpha-1,\alpha+1,\ldots,n\}$, consider all the states $(t_1,t_2,f)$ in $\nu_{k-1}^{(\alpha,C'_\alpha)\#}$
        o If $t_1 + p_k \le C_\alpha - p_\alpha$, then add state $(t_1 + p_k, t_2, f)$ to the state space $\nu_k^{(\alpha,C'_\alpha)\#}$
        o Add $\big(t_1, t_2 + p_k, f + w_k(min\{C_\alpha + t_2 + p_k, d_2\} - d_1)\big)$ to the state space $\nu_k^{(\alpha,C'_\alpha)\#}$
        o Add $(t_1, t_2, f + s_k)$to the state space $\nu_k^{(\alpha,C'_\alpha)\#}$
    1.3. For all states $(t_1,t_2,f) \in \nu_k^{(\alpha,C'_\alpha)\#}$ with equal values for $t_1$and $t_2$, keep at most one state
         with the minimum value of $f$.
    1.4. Remove the state space $\nu_{k-1}^{(\alpha,C'_\alpha)\#}$
    1.5. Keep at most one state with the smallest value of $t_1$ among all states $(t_1,t_2,f) \in \nu_k^{(\alpha,C'_\alpha)\#}$
         laying in common sub-intervals for $t_2$ and $f$.
    1.6. Set $Z^\#_{\alpha,C'_\alpha} = \min_{[t_1,t_2,f]\in\nu_{n-1}^{(\alpha,C'_\alpha)\#}} \{f + w_\alpha.(min\{C'_\alpha, d_2\} - d_1)\}$
Step 2. Return the approximate solution $Z^\# = \min_{\alpha,C'_\alpha} Z^\#_{\alpha,C'_\alpha}$
```

In the FPTAS algorithm, we have replaced the statement “if $C_\alpha + t_2 + p_k \le d_2$, then add state $\big(t_1, t_2 + p_k, f + w_k(C_\alpha + t_2 + p_k - d_1)\big)$” from step 1.2 of the DP algorithm with phrase “add state $\big(t_1, t_2 + p_k, f + w_k(min\{C_\alpha + t_2 + p_k, d_2\} - d_1)\big)$”. Here, we have two cases:

$$C_\alpha + t_2 + p_k \le d_2 \Rightarrow w_k(min\{C_\alpha + t_2 + p_k, d_2\} - d_1) = w_k(C_\alpha + t_2 + p_k - d_1)$$
$$C_\alpha + t_2 + p_k > d_2 \Rightarrow w_k(min\{C_\alpha + t_2 + p_k, d_2\} - d_1) = w_k(d_2 - d_1) = s_k$$

The first case shows the same condition as in the DP algorithm and in the second case, job k is scheduled in the second group, but its completion time is greater than d_2 and its tardiness penalty equals s_k. According to this, we do not need to restrict variable t_2 in the FPTAS because penalties are calculated correctly and from Eqs (5) and (6), the number of intervals for variable $t_2$is $O(n^3/\varepsilon)$, which is sufficient for the FPTAS.

4-1- Analysis of worst case ratio bound

The worst case analysis of this FPTAS will be based on comparing the steps of algorithms DP and FPTAS. We may remark that the main action of FPTAS consists of reducing the cardinality of the state spaces by splitting the intervals of C_α, t_2 and finto sub-intervals and then replacing all vectors $(t_1,t_2,f) \in \nu_k^{(\alpha,C_\alpha)\#}$ belonging to the same sub-intervals by a single approximate state with the smallest t_1. Let W^k_{max} denote the maximum tardiness weight for a tardy job in the DP between jobs *k*to*n*.

$W^k_{max} = max\{w_{k+i} \mid 0 \le i \le n-k, C_{k+i} > C_\alpha\}$ (7)

Lemma 1. Suppose (t_1,t_2,f) is an arbitrary state in $\nu_k^{(\alpha,C_\alpha)}$. Algorithm FPTAS creates at least one state $\big(t_1^\#, t_2^\#, f^\#\big)$ in $\nu_k^{(\alpha,C'_\alpha)\#}$ such that

$t_1^\# \le t_1$ (8)

$t_2^\# \le t_2 + (k.\Delta_2)/\big(n.W^k_{max}\big)$ (9)

and

$$f^{\#} \le f + k\Delta_2 + k\Delta_3 + (\varepsilon/3)f \tag{10}$$

Proof. The proof is done by induction on parameter *k*. First, for $k = 0$, the statement is trivial because$\nu_0^{(\alpha,C'_\alpha)\#} = \nu_0^{(\alpha,C_\alpha)}$. Now, assume that the lemma holds up to level *k*-1and we want to show that it is valid for level *k*. Consider an arbitrary state $(t_1, t_2, f) \in \nu_k^{(\alpha,C_\alpha)}$. Algorithm DP introduces this state into $\nu_k^{(\alpha,C_\alpha)}$ when job *k* is added to some feasible state for the first *k*-1 jobs. Let (t'_1, t'_2, f') be the above feasible state.

If $W_{max}^k = 0$, the proof is trivial because in this case, there is no tardy job in the DP between jobs *k* and *n* and the DP algorithm schedules the remaining jobs as early. In this condition, the FPTAS algorithm can also schedule the remaining jobs as early, because $t_1^{\#} \le t_1$ and there is enough empty space for the early jobs in the FPTAS. In other words, $W_{max}^k = 0$ means that the penalty for the DP remains unchanged from steps *k* to *n* and the FPTAS algorithm is also able to create a state in the final step *n* with the same penalty as state *k*. So, the proof of lemma 1 is obvious in this condition. According to the above discussion, we assume $W_{max}^k > 0$ in the rest of the proof.

Let C_α and C'_α denote the completion time of the straddling job in algorithms DP and FPTAS, respectively. Also, let T_α^* and T'_α be the tardiness of the straddling job in these two algorithms. The FPTAS algorithm considers the completion time for the straddling job as $C'_\alpha = \lfloor d_1 + (\Delta_1)^y \rfloor$, $y = 0,1,\ldots,L_1$. This divides the possible interval $[d_1 + 1, d_1 + p_\alpha]$ into L_1 integer subintervals $[\lceil d_1 + (\Delta_1)^{y-1} \rceil, \lfloor d_1 + (\Delta_1)^y \rfloor]$, $y = 1,\ldots,L_1$. Given the fact that the completion time of the straddling job in the DP, C_α, also falls in interval $[d_1 + 1, d_1 + p_\alpha]$, we have

$$\begin{aligned}
&\forall\, C_\alpha \in I_y = [\lceil d_1 + (\Delta_1)^{y-1} \rceil, \lfloor d_1 + (\Delta_1)^y \rfloor] \quad \Rightarrow \quad \exists C'_\alpha = \lfloor d_1 + (\Delta_1)^y \rfloor \qquad \forall y = 1,\ldots,L_1 \\
&\Rightarrow C_\alpha \ge d_1 + (\Delta_1)^{y-1} \quad \Rightarrow \quad C_\alpha - d_1 \ge (\Delta_1)^{y-1} \\
&\Rightarrow \Delta_1.(C_\alpha - d_1) \ge \Delta_1.(\Delta_1)^{y-1} = (\Delta_1)^y \ge \lfloor (\Delta_1)^y \rfloor = C'_\alpha - d_1 \\
&\Rightarrow \Delta_1.T_\alpha^* \ge T'_\alpha \xrightarrow{\Delta_1 = 1+\varepsilon/3} T'_\alpha \le T_\alpha^* + \frac{\varepsilon}{3} T_\alpha^*
\end{aligned} \tag{11}$$

Three cases can be distinguished here. In the first one, we have $(t_1, t_2, f) = (t'_1 + p_k, t'_2, f')$. The second possibility is $(t_1, t_2, f) = (t'_1, t'_2 + p_k, f' + w_k(min\{C_\alpha + t'_2 + p_k, d_2\} - d_1))$ and the third one is $(t_1, t_2, f) = (t'_1, t'_2, f' + s_k)$. We want to prove the statement for level *k* in these three cases.

Case A.$(t_1, t_2, f) = (t'_1 + p_k, t'_2, f')$

Since$(t'_1, t'_2, f') \in \nu_{k-1}^{(\alpha,C_\alpha)}$, we have a state $\left(t'^{\#}_1, t'^{\#}_2, f'^{\#}\right) \in \nu_{k-1}^{(\alpha,C'_\alpha)\#}$ in such a way that $t'^{\#}_1 \le t'_1$, $t'^{\#}_2 \le t'_2 + \left((k-1).\Delta_2\right)/\left(n.W_{max}^{k-1}\right)$ and $f'^{\#} \le f' + (k-1)\Delta_1 + (k-1)\Delta_2 + (\varepsilon/3)f'$. Thus, the state $\left(t'^{\#}_1 + p_k, t'^{\#}_2, f'^{\#}\right)$ is yielded by algorithm FPTAS at level k, but we may throw this state away when reducing the state space. Let (λ, μ, γ) be the remaining state in $\nu_k^{(\alpha,C'_\alpha)\#}$ that falls in the same sub-intervals of t_2 and $f$as $\left(t'^{\#}_1 + p_k, t'^{\#}_2, f'^{\#}\right)$ after reduction. So,

$$\lambda \le t'^{\#}_1 + p_k \le t'_1 + p_k = t_1 \tag{12}$$

$$\begin{aligned}
\mu &\le t'^{\#}_2 + \frac{\Delta_2}{nW_{max}^k} \\
&\le t'_2 + \frac{(k-1)\Delta_2}{nW_{max}^{k-1}} + \frac{\Delta_2}{nW_{max}^k} \\
&\le t_2 + \frac{k\Delta_2}{nW_{max}^k}
\end{aligned} \tag{13}$$

also,

$$\begin{aligned}\gamma &\le f'^{\#} + \Delta_3 \\ &\le f' + (k-1)\Delta_2 + (k-1)\Delta_3 + (\varepsilon/3)f' + \Delta_3 \\ &\le f + k\Delta_2 + k\Delta_3 + (\varepsilon/3)f\end{aligned} \tag{14}$$

The last inequality in Eq. (13) is derived from $W_{max}^{k} \le W_{max}^{k-1}$. Therefore, the statement holds for level *k* in this case.

Case B. $(t_1, t_2, f) = \left(t'_1, t'_2 + p_k, f' + w_k(min\{C_\alpha + t'_2 + p_k, d_2\} - d_1)\right)$

Since $(t'_1, t'_2, f') \in \nu_{k-1}^{(\alpha,C_\alpha)}$, we have a state $\left(t'_1{}^{\#}, t'^{\#}_2, f'^{\#}\right) \in \nu_{k-1}^{(\alpha,C'_\alpha)\#}$ in such a way that $t'_1{}^{\#} \le t'_1$, $t'^{\#}_2 \le t'_2 + \left((k-1)\,.\,\Delta_2\right)/\left(n\,.\,W_{max}^{k-1}\right)$ and $f'^{\#} \le f' + (k-1)\Delta_1 + (k-1)\Delta_2 + (\varepsilon/3)f'$. Therefore, the state $\left(t'^{\#}_1, t'^{\#}_2 + p_k, f'^{\#} + w_k\left(min\left\{C'_\alpha + t'^{\#}_2 + p_k, d_2\right\} - d_1\right)\right)$ is generated by algorithm FPTAS at level *k*. But, we may eliminate it in step 1.3. Let $(\lambda', \mu', \gamma') \in \nu_{k}^{(\alpha,C'_\alpha)\#}$ be the remaining state that is in the same sub-intervalst_2 and fas $\left(t'^{\#}_1, t'^{\#}_2 + p_k, f'^{\#} + w_k\left(min\left\{C'_\alpha + t'^{\#}_2 + p_k, d_2\right\} - d_1\right)\right)$ after reduction. So,

$$\lambda' \le t'_1{}^{\#} \le t'_1 = t_1 \tag{15}$$

$$\mu' \le t'^{\#}_2 + p_k + \frac{\Delta_2}{nW_{max}^{k}} \le t'_2 + \frac{(k-1)\Delta_2}{nW_{max}^{k-1}} + p_k + \frac{\Delta_2}{nW_{max}^{k}} \le t_2 + \frac{k\Delta_2}{nW_{max}^{k}} \tag{16}$$

$$\begin{aligned}\gamma' &\le f'^{\#} + w_k\left(min\left\{C_\alpha^{\#} + t'^{\#}_2 + p_k - d_1, d_2 - d_1\right\}\right) + \Delta_3 \\ &\le f'^{\#} + w_k\left(min\left\{T_\alpha^{\#} + t'^{\#}_2 + p_k, d_2 - d_1\right\}\right) + \Delta_3 \\ &\le f'^{\#} + w_k\left(T_\alpha^{\#} + t'^{\#}_2 + p_k\right) + \Delta_3 \\ &\le f' + (k-1)\Delta_2 + (k-1)\Delta_3 + (\varepsilon/3)f' + w_k\left(T_\alpha^{*} + (\varepsilon/3)T_\alpha^{*} + t'_2 + \frac{(k-1)\Delta_2}{n\,.\,W_{max}^{k-1}} + p_k\right) + \Delta_3 \\ &\le f' + (k-1)\Delta_2 + k\Delta_3 + (\varepsilon/3)f' + w_k\left(T_k^{*} + (\varepsilon/3)T_\alpha^{*} + \frac{(k-1)\Delta_2}{n\,.\,W_{max}^{k-1}}\right) \\ &\le f + (k-1)\Delta_2 + k\Delta_3 + (\varepsilon/3)f' + w_k\left((\varepsilon/3)T_\alpha^{*} + \frac{(k-1)\Delta_2}{n\,.\,W_{max}^{k-1}}\right) \\ &\le f + k\Delta_2 + k\Delta_3 + (\varepsilon/3)f' + w_k\left((\varepsilon/3)T_\alpha^{*}\right) \\ &\le f + k\Delta_2 + k\Delta_3 + (\varepsilon/3)f\end{aligned} \tag{17}$$

It can be easily seen that the inequalities in Eq. (17) are concluded from the following relations.

$$T_\alpha^{\#} \le T_\alpha^{*} + (\varepsilon/3)T_\alpha^{*} \tag{18}$$

$$T_k^{*} = T_\alpha^{*} + t'_2 + p_k \tag{19}$$

$$f = f' + w_k T_k^{*} \tag{20}$$

$$w_k \frac{(k-1)\Delta_2}{n\,.\,W_{max}^{k-1}} \le \frac{(k-1)\Delta_2}{n} \le \Delta_2 \tag{21}$$

Accordingly, the statement holds for level *k* in the second case.

Case C.$(t_1, t_2, f) = (t'_1, t'_2, f' + s_k)$

Since $[t'_1, t'_2, f'] \in v_{k-1}^{(\alpha, C_\alpha)}$, we have a state $\left(t'_1{}^{\#}, t'^{\#}_2, f'^{\#}\right) \in v_{k-1}^{(\alpha, C'_\alpha)\#}$ in such a way that $t'_1{}^{\#} \le t'_1$, $t'^{\#}_2 \le t'_2 + \left((k-1)\,.\,\Delta_2\right)/\left(n\,.\,W_{max}^{k-1}\right)$and $f'^{\#} \le f' + (k-1)\Delta_1 + (k-1)\Delta_2 + (\varepsilon/3)f'$. Thus, the state $\left(t'_1{}^{\#}, t'_2{}^{\#}, f'^{\#} + s_k\right)$ will be created by algorithm FPTAS at level k which may be eliminated when reducing the state space. Let $(\lambda'', \mu'', \gamma'')$ be the remaining state in $v_k^{(\alpha, C'_\alpha)\#}$ that is in the same sub-intervals of $t_2$and $f$as $\left(t'_1{}^{\#}, t'_2{}^{\#}, f'^{\#} + s_k\right)$ after reduction. So, we have

$$\lambda'' \le t'_1{}^{\#} \le t'_1 = t_1 \tag{22}$$

$$\mu'' \le t'^{\#}_2 + \frac{\Delta_2}{nW_{max}^k} \le t'_2 + \frac{(k-1)\Delta_2}{nW_{max}^{k-1}} + \frac{\Delta_2}{nW_{max}^k} \le t_2 + \frac{k\Delta_2}{nW_{max}^k} \tag{23}$$

also,

$$\begin{aligned} \gamma'' &\le f'^{\#} + s_k + \Delta_3 \\ &\le f' + (k-1)\Delta_2 + (k-1)\Delta_3 + (\varepsilon/3)f' + s_k + \Delta_3 \\ &\le f + k\Delta_2 + k\Delta_3 + (\varepsilon/3)f \end{aligned} \tag{24}$$

This indicates the correctness of Lemma 1 in this case and hence, we conclude the proof in its general form. ■

Theorem 2. For any arbitrary$\varepsilon > 0$, algorithm FPTAS generates a solution $Z^{\#}$such that $Z^{\#} - Z^{*} \le \varepsilon Z^{*}$ holds.

Proof. Suppose $\alpha^*$is the straddling job in an optimal sequence. Job α^*will also be selected as the straddling in one of the iterations of algorithm FPTAS. By definition, the optimal solution can be associated with a state (t_1, t_2, f)in $v_{n-1}^{(\alpha*, C_{\alpha^*})}$. From Lemma 2, algorithm FPTAS generates a state $\left(t_1^{\#}, t_2^{\#}, f^{\#}\right) \in v_{n-1}^{(\alpha^*, C'_{\alpha^*})\#}$ such that $t_1^{\#} \le t_1$and $f^{\#} \le f + (n-1)\Delta_2 + (n-1)\Delta_3 + (\varepsilon/3)f$. Also, there always exists a feasible state in FPTAS which is related to any feasible state generated by DP. It is because $t_1^{\#} \le t_1$ holds in all states of FPTAS and none of the states will be lost by the constraint $t + p_k \le d_1$ in Step 1.2. We also have

$$f^{\#} \le f + (n-1)\Delta_2 + (n-1)\Delta_3 + (\varepsilon/3)f \tag{25}$$

$$\left.\begin{aligned} \Delta_2 &= \frac{\varepsilon\,.\,LB}{3n} \\ \Delta_3 &= \frac{Z_H}{L_3} = \frac{LB/2}{\left\lceil \frac{2n}{\varepsilon} \right\rceil} \le \frac{\varepsilon}{3n} LB \end{aligned}\right\} \Rightarrow f^{\#} \le f + (\varepsilon/3)LB + (\varepsilon/3)LB + (\varepsilon/3)f \le (1+\varepsilon)f \tag{26}$$

Also, with regard to $T_{\alpha^*}^{\#} \le T_{\alpha^*}^{*} + (\varepsilon/3)T_{\alpha^*}^{*}$and $Z^* = f + T_{\alpha^*}^{*}$, it can be concluded that

$$\begin{aligned} Z^{\#} &= f^{\#} + T_{\alpha^*}^{\#} \\ &\le (1+\varepsilon)f + T_{\alpha^*}^{*} + (\varepsilon/3)T_{\alpha^*}^{*} \\ &\le (1+\varepsilon)Z^* \end{aligned} \tag{27}$$

and this completes the proof. ■

4-2- Time complexity of algorithm FPTAS

To complete our analysis, we try to estimate the running time of algorithm FPTAS. Before starting the first step, algorithm MWR must be applied and implemented in $O(n\, logn)$. In each iteration of Step 1.2, the state space $\nu_k^{(\alpha,C\prime_\alpha)\#}(k=\{1,2,\dots,n-1\})$ is generated. Note that algorithm FPTAS, in each step, keeps only one state with minimum t_1 among all states laying in common sub-intervals for variables t_2 and f. So, the number of states in $\nu_k^{(\alpha,C\prime_\alpha)\#}$ does not exceed $L_2.L_3$and from $L_2 = O(n^3/\varepsilon)$and $L_3 = O(n/\varepsilon)$, the running time of Step 1.2 is calculated as

$$\sum_{k=1}^{n-1} \left|\nu_k^{(\alpha,C\prime_\alpha)\#}\right| \leq n\,.L_2\,.L_3 = O(n^5/\varepsilon^2) \tag{28}$$

Since the algorithm iterates n times by each choice of the straddling job and also, guessing C_α needs at most $L_1 = O(lnP_{max}/\varepsilon)$ iterations, we can deduce that Step 1 is of the order $O(n^6 lnP_{max}/\varepsilon^3)$. Step 2 is implemented with the running time $O(nlnP_{max}/\varepsilon)$ and finally, the whole algorithm requires $O(n^6 lnP_{max}/\varepsilon^3)$ time.

5- Conclusion

In this paper, we analyzed a less studied performance measure called Tardy/Lost penalty function for scheduling problems and discussed its practical and theoretical applications. According to this performance measure, each job has two due dates and, depending on its completion time, it is classified into one of the three groups: early, tardy, or lost jobs.

An FPTAS is the strongest possible polynomial time approximation result that we can derive for an NP-hard problem. Woeginger(2000)introduced a set of conditions on optimization problems that guarantee the existence of an FPTAS. These conditions cover a relatively large subclass of the fully polynomial time approximable problems including the problem considered in this paper.

A polynomial time approximation algorithm, named MWR, was designed for problem $1|d_i = d|TL$. It was shown that its worst case ratio was bounded by 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 was found to run in $O(n^6 lnP_{max}/\varepsilon^3)$ time.

As a perspective, 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 penalty function in other scheduling environments such as parallel machines or flow-shops and extend the results to more complex problems. Development of better approximation algorithms and FPTASs is also a challenging subject.

References

Alminaya, M., Escudero, L. F., Landete, M., Monge, J. F., Rabasa, A. & Sanchez-Soriano, J. 2010. A DSS for water irrigation scheduling. *Omega,* 38**,** 492-500.

Baptiste, P. &Sadykov, R. 2009. On scheduling a single machine to minimize a piecewise linear objective function: A compact MIP formulation. *Naval Research Logistics,* 56**,** 487-502.

Blazewicz, J. 1984. Scheduling preemptible tasks on parallel processors with information loss. *Technique et Science Informatiques,* 3**,** 415-420.

Blazewicz, J., Pesch, E., Sterna, M. & Werner, F. 2004. Open shop scheduling problems with late work criteria. *Discrete Applied Mathematics,* 134**,** 1-24.

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

Chen, B., Potts, C. N. & Woeginger, G. J. 1998. A review of machine scheduling. *In:* DU, D. Z. & Pardalos, P. M. (eds.) *Handbook of combinatorial optimization.* Boston: Kluwer Academic Publishers.

Engels, D. W., Karger, D. R., Kolliopoulos, S. G., Sengupta, S., Uma, R. N. & Wein, J. 2003. Techniques for Scheduling with Rejection. *Journal of Algorithms,* 49**,** 175-191.

Federgruen, A. & Mosheiov, G. 1994. Greedy heuristics for single-machine scheduling problems with general earliness and tardiness costs. *Operations Research Letters,* 16**,** 199-208.

Ibarra, O. & Kim, C. E. 1875. Fast approximation algorithms for the knapsack and sum of subset problems. *problems, Journal of the ACM,* 22**,** 463 468.

Ji, M. & Cheng, T. C. E. 2010. Batch scheduling of simple linear deteriorating jobs on a single machine to minimize makespan. *European Journal of Operational Research,* 202**,** 90-98.

Kacem, I. & Mahjoub, A. R. 2009. 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,* 56**,** 1708-1712.

Kahlbacher, H. G. 1993. Scheduling with monotonous earliness and tardiness penalties. *European Journal of Operational Research,* 64**,** 258-277.

Kathley, R. B. & Alidaee, B. 2002. Single machine scheduling to minimize total weighted late work: a comparison of scheduling rules and search algorithms. *Computers and Industrial Engineering,* 43**,** 509-528.

Kianfar, K. & Moslehi, G. 2013. A note on ‘‘Fully polynomial time approximation scheme for the total weighted tardiness minimization with a common due date’’. *Discrete Applied Mathematics,* 161**,** 2205-2206.

Lawler, E. L. 1964. On scheduling problems with deferral costs. *management Science,* 11**,** 280-288.

Leung, J. Y. T. 2004. Minimizing total weighted error for imprecise computation tasks and related problems. *In:* LEUNG, J. Y. T. (ed.) *Handbook of scheduling: algorithms, models and performance analysis.* Boca Raton: CRC Press.

Moslehi, G. & Kianfar, K. 2014. Approximation Algorithms and an FPTAS for the Single Machine Problem with Biased Tardiness Penalty. *Journal of Applied Mathematics*.

Pesch, E. & STERNA, M. 2009. Late work minimization in flow shop by a gegnetic algorithm. *Computers and Industrial Engineering,* 57**,** 1202-1209.

Pinedo, M. 1995. *Scheduling: theory, algorithms and systems,* New Jersey, Prentice Hall.

Potts, C. N. & Vav Wassenhove, L. N. 1992a. Approximation algorithms for schrduling a single machine to minimize total late work. *Operations Research Letters,* 11**,** 261-266.

Potts, C. N. & VAV WASSENHOVE, L. N. 1992b. Single machine scheduling to minimize total late work. *Operations Research,* 40**,** 586-595.

Ren, J., Zhang, Y. & Sun, J. 2009. The NP-hardness of minimizing the total late workk on an unbounded batch machine. *Asia-Pacific Journal of Operational Research,* 26**,** 351-363.

Shabtay, D. 2008. Due date assignment and scheduling a single machine with a general earliness/tardiness cost function. *Computers and Operations Research,* 35**,** 1539-1545.

Shabtay, D. & Bensoussan, Y. 2010. Maximizing the weighted number of just-in-time jobs in several two-machine scheduling systems. *Journal of Scheduling,* 15**,** 39-47.

Shabtay, D., Bensoussan, Y. & Kaspi, M. 2012. 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,* 136**,** 67-74.

Shabtay, D., Gaspar, N. & Kaspi, M. 2013. A survey on offline scheduling with rejection. *Journal of Scheduling,* 16**,** 3-28.

Slotnick, S. A. 2011. Order acceptance and scheduling: A taxonomy and review. *European Journal of Operational Research,* 212**,** 1-11.

Steiner, G. & Zhang, R. 2009. Approximation algorithms for minimizing the total weighted number of late jobs with late deliveries in two-level supply chains. *Journal of Scheduling,* 12**,** 565-574.

Sterna, M. 2000. *Problems and algorithms in non-classical shop scheduling,* Poznan, Scientific Publishers ofthe Polish Academy of Sciences.

Sterna, M. 2006. *Late work scheduling in shop systems.* Posnan University of Technology.

Sterna, M. 2007a. Dominancec relations for two-machine flow-shop problem with late work criterion. *Bulletin of the Polish Academy of Sciences,* 55**,** 59-69.

Sterna, M. 2007b. Late work minimization in a small flexible manufacturing system. *Computers and Industrial Engineering,* 52**,** 210-228.

Sterna, M. 2011. A survey of scheduling problems with late work criteria. *Omega,* 39**,** 120-129.

Ventura, J. A. & Rahhakrishnan, S. 2003. Single machine scheduling with symmetric earliness and tardiness penalties. *European Journal of Operational Research,* 144**,** 598-612.

Woeginger, G. J. 2000. When does a dynamic programming formulation guarantee the existence of an FPTAS? *INFORMS Journal on Computing,* 12**,** 57-74.

Yuan, J. 1992. The NP-hardness of the single machine common due date weighted tardiness problem. *Systems Science and Mathematical Sciences,* 5**,** 328-333.

Zhou, X. & CAI, X. 1997. General stochastic single-machine scheduling with regular cost functions. *Mathematical and Computer Modelling,* 26**,** 95-108.

Appendix A: Proof of Theorem 1

Let *G* denote a sequence generated by algorithm MWR and each notation including an asterisk (*) be related to an optimal sequence. Algorithm MWR schedules jobs iteratively from the end of the sequence to the beginning. Through this procedure, jobs are classified into three groups, early, tardy, and lost. On the other hand, each job may be early, tardy, or lost in the optimal sequence. Thus, we get nine groups of jobs,$\{H, F, B, H', F', B', H'', F'', B''\}$, as shown below. For example, sets ${B''}^G_1, \dots, {B''}^G_{q_1}$ include jobs in sequence *G* which are early in both the sequences and sets ${B''}^*_1, \dots, {B''}^*_{q_2}$ include the same jobs in the optimal sequence. As another example, sets ${F''}^G_1, \dots, {F''}^G_{q_1}$ are some jobs in sequence *G* which are early in *G* and tardy in the optimal sequence and appear in the optimal sequence in sets ${F''}^*_1, \dots, {F''}^*_{m_2}$. Define the sum of the processing times of jobs in a sub-sequence σas p_σ.

$$sequence\ \ G = \left(\begin{array}{l} \overbrace{\left({B''}^G_{q_1}, {F''}^G_{q_1}, {H''}^G_{q_1}, \dots, {B''}^G_1, {F''}^G_1, {H''}^G_1\right)}^{early\ jobs} \overbrace{\left({B'}^G_{m_1}, {F'}^G_{m_1}, {H'}^G_{m_1}, \dots, {B'}^G_1, {F'}^G_1, {H'}^G_1\right)}^{tardy\ jobs} \\ \overbrace{\left(B^G_{k_1}, F^G_{k_1}, H^G_{k_1}, \dots, B^G_1, F^G_1, H^G_1\right)}^{lost\ jobs} \end{array} \right)$$

$$optimal\ sequence = \left(\begin{array}{l} \overbrace{\left({B''}^*_{q_2}, {F'}^*_{q_2}, F^*_{q_2}, \dots, {B''}^*_1, {F'}^*_1, F^*_1\right)}^{early\ jobs} \overbrace{\left({B'}^*_{m_2}, {F''}^*_{m_2}, H^*_{m_2}, \dots, {B'}^*_1, {F''}^*_1, H^*_1\right)}^{tardy\ jobs} \\ \overbrace{\left(B^*_{k_2}, {H''}^*_{k_2}, {H'}^*_{k_2}, \dots, B^*_1, {H''}^*_1, {H'}^*_1\right)}^{lost\ jobs} \end{array} \right)$$

First, we must compare the penalty of lost jobs in *G* (jobs in sets H^G, F^GandB^G) with some penalties in the optimal sequence; then, we should repeat this comparison for tardy jobs in sequence *G*. Given the fact that algorithm MWR selects jobs according to the non-decreasing order of their w_i/p_iratios and that jobs in ${H'}^G$and ${H''}^G$are not selected to be lost in *G*, we have

$$\frac{w_i}{p_i} \le \frac{w_j}{p_j} \forall i \in \{H^G, F^G\}, \qquad i \ne J^{\{L\}}_{first} \quad , \quad \forall j \in \{{H'}^G, {H''}^G\} \tag{A1}$$

Let $J^{\{L\}}_{first}$ be the first lost job in *G*. Thus, it is obvious that the lost jobs in *G* except $J^{\{L\}}_{first}$ cannot fill the time interval between $d_2$and P_{sum}, but the lost jobs in the optimal sequence do. So, as jobs in sets *B* are common in both the sequences, we get

$$\left. \begin{array}{l} \sum_{i=1}^{k_1} \left(p_{H^G_i} + p_{F^G_i} + p_{B^G_i}\right) - p_{J^{\{L\}}_{first}} < P_{sum} - d_2 \\ \sum_{i=1}^{k_2} \left(p_{{H'}^*_i} + p_{{H''}^*_i} + p_{B^*_i}\right) \ge P_{sum} - d_2 \end{array} \right\} \Rightarrow \sum_{i=1}^{k_1} \left(p_{H^G_i} + p_{F^G_i}\right) - p_{J^{\{L\}}_{first}} < \sum_{i=1}^{k_2} \left(p_{{H'}^*_i} + p_{{H''}^*_i}\right) \tag{A2}$$

From Eqs. (A1) and (A2), we conclude

$$\sum_{i=1}^{k_1} \sum_{j \in H^G_i} Z_j + \sum_{i=1}^{k_1} \sum_{j \in F^G_i} Z_j - Z_{J^{\{L\}}_{first}} < \sum_{i=1}^{k_2} \sum_{j \in {H'}^*_i} Z_j + \sum_{i=1}^{k_2} \sum_{j \in {H''}^*_i} Z_j \tag{A3}$$

The above inequality can be summarized as $Z_{H^G} + Z_{F^G} - Z_{J^{\{L\}}_{first}} < Z_{H'^*} + Z_{H''^*}$. Let $Z^G_{\{L\}}$ and $Z^*_{\{L\}}$ denote the penalty of the lost jobs in *G* and in the optimal sequence, respectively. So,

$$Z^G_{\{L\}} = Z_{H^G} + Z_{F^G} + Z_{B^G} \\ < Z_{H'^*} + Z_{H''^*} + Z_{J^{\{L\}}_{first}} + Z_{B^*} \tag{A4}$$

In order to evaluate the penalty of tardy jobs in *G*, we should compare the penalty related to sets H'^G, F'^G and $B'^G$with some penalties in the optimal sequence.

$$Z^G_{\{T\}} = \sum_{i=1}^{m_1} \sum_{j \in H'^G_i} [w_j(C^G_j - d_1)] + \sum_{i=1}^{m_1} \sum_{j \in F'^G_i} [w_j(C^G_j - d_1)] + \sum_{i=1}^{m_1} \sum_{j \in B'^G_i} [w_j(C^G_j - d_1)] \tag{A5}$$

Jobs in sets *H* are penalized only in the optimal sequence and so, the worst case ratio occurs when the tardiness weights, w_i, are equal to zero for these sets. Therefore, according to the WSPT order for tardy jobs in the optimal sequence, we conclude that sets *H* must be scheduled after sets B' and F'' in the optimal sequence. Therefore,

$$optimal\ sequence = \begin{pmatrix} (B''^*_{q_2}, F'^*_{q_2}, F^*_{q_2}, \dots, B''^*_1, F'^*_1, F^*_1)(B'^*_{m_2}, F''^*_{m_2}, \dots, B'^*_1, F''^*_1, H^*_{m_2}, \dots, H^*_1) \\ (B^*_{k_2}, H''^*_{k_2}, H'^*_{k_2}, \dots, B^*_1, H''^*_1, H'^*_1) \end{pmatrix}$$

In the following, we will consider the three expressions in (A5) one by one. Let $S_{F'^G_i}$ and $S_{B'^G_i}$ show the starting times of sets F'^G_i and B'^G_i in sequence *G*, respectively. As algorithm MWR selects jobs in F'^G_i after jobs in H'^G_i for each $i \in \{1,2,\dots,m_1\}$, we have

$$\begin{aligned} \sum_{i=1}^{m_1} \sum_{j \in F'^G_i} [w_j(C^G_j - d_1)] &\le \sum_{i=1}^{m_1} \sum_{j \in F'^G_i} \left[w_j \left(\sum_{r=i+1}^{m_1} \left(p_{F'^G_r} + p_{H'^G_r} \right) + \sum_{r=i}^{m_1} p_{B'^G_r} + \left(C^G_j - S_{F'^G_i} \right) \right) \right] \\ &= \sum_{i=1}^{m_1} \sum_{j \in F'^G_i} \left[w_j \left(\sum_{r=i+1}^{m_1} p_{F'^G_r} + \sum_{r=i}^{m_1} p_{B'^G_r} + \left(C^G_j - S_{F'^G_i} \right) \right) \right] + \sum_{i=1}^{m_1} \sum_{j \in F'^G_i} \left[w_j \sum_{r=i+1}^{m_1} p_{H'^G_r} \right] \\ &\le \sum_{i=1}^{m_1} \sum_{j \in F'^G_i} \left[w_j \left(\sum_{r=i+1}^{m_1} p_{F'^G_r} + \sum_{r=i}^{m_1} p_{B'^G_r} + \left(C^G_j - S_{F'^G_i} \right) \right) \right] + \sum_{i=1}^{m_1} \sum_{j \in H'^G_i} \left[w_j \sum_{r=1}^{i} p_{F'^G_r} \right] \end{aligned} \tag{A6}$$

Similarly, we conclude the following inequalities for sets B'^G_i.

$$\begin{aligned} \sum_{i=1}^{m_1} \sum_{j \in B'^G_i} [w_j(C^G_j - d_1)] &\le \sum_{i=1}^{m_1} \sum_{j \in B'^G_i} \left[w_j \left(\sum_{r=i+1}^{m_1} \left(p_{F'^G_r} + p_{H'^G_r} + p_{B'^G_r} \right) + \left(C^G_j - S_{B'^G_i} \right) \right) \right] \\ &= \sum_{i=1}^{m_1} \sum_{j \in B'^G_i} \left[w_j \left(\sum_{r=i+1}^{m_1} \left(p_{F'^G_r} + p_{B'^G_r} \right) + \left(C^G_j - S_{B'^G_i} \right) \right) \right] + \sum_{i=1}^{m_1} \sum_{j \in B'^G_i} \left[w_j \sum_{r=i+1}^{m_1} p_{H'^G_r} \right] \\ &\le \sum_{i=1}^{m_1} \sum_{j \in B'^G_i} \left[w_j \left(\sum_{r=i+1}^{m_1} \left(p_{F'^G_r} + p_{B'^G_r} \right) + \left(C^G_j - S_{B'^G_i} \right) \right) \right] + \sum_{i=1}^{m_1} \sum_{j \in H'^G_i} \left[w_j \sum_{r=1}^{i} p_{B'^G_r} \right] \end{aligned} \tag{A7}$$

Next, considering the direct penalty of sets H'^G_i and their penalty in (A6) and (A7), we get

$$\begin{aligned}
&\sum_{i=1}^{m_1}\sum_{j\in H'^G_i}\left[w_j.\left(C_j-d_1\right)\right]+\sum_{i=1}^{m_1}\sum_{j\in H'^G_i}\left[w_j\sum_{r=1}^{i}p_{F'^G_r}\right]+\sum_{i=1}^{m_1}\sum_{j\in H'^G_i}\left[w_j\sum_{r=1}^{i}p_{B'^G_r}\right]\\
&\quad=\sum_{i=1}^{m_1}\sum_{j\in H'^G_i}\left[w_j\left(C_j+\sum_{r=1}^{i}p_{F'^G_r}+\sum_{r=1}^{i}p_{B'^G_r}-d_1\right)\right]\\
&\quad\le\sum_{i=1}^{m_1}\sum_{j\in H'^G_i}\left[w_j\left(d_2-d_1\right)\right]\\
&\quad=\sum_{i=1}^{k_2}Z_{H'^*_i}
\end{aligned}\tag{A8}$$

And so,

$$\begin{aligned}
Z^G_{\{T\}}&=\sum_{i=1}^{m_1}\sum_{j\in F'^G_i}\left[w_j\left(C^G_j-d_1\right)\right]+\sum_{i=1}^{m_1}\sum_{j\in B'^G_i}\left[w_j\left(C^G_j-d_1\right)\right]+\sum_{i=1}^{m_1}\sum_{j\in H'^G_i}\left[w_j\left(C^G_j-d_1\right)\right]\\
&\le\sum_{i=1}^{m_1}\sum_{j\in F'^G_i}\left[w_j\left(\sum_{r=i+1}^{m_1}p_{F'^G_r}+\sum_{r=i}^{m_1}p_{B'^G_r}+\left(C^G_j-S_{F'^G_i}\right)\right)\right]\\
&\quad+\sum_{i=1}^{m_1}\sum_{j\in B'^G_i}\left[w_j\left(\sum_{r=i+1}^{m_1}\left(p_{F'^G_r}+p_{B'^G_r}\right)+\left(C^G_j-S_{B'^G_i}\right)\right)\right]+\sum_{i=1}^{k_2}Z_{H'^*_i}\\
&=Z^{\left[B'^G_{m_1},F'^G_{m_1},\dots,B'^G_2,F'^G_2,B'^G_1,F'^G_1\right]}_{\left(d_1+\sum_{i=1}^{m_1}\left(p_{F'^G_i}+p_{B'^G_i}\right)\right)}+\sum_{i=1}^{k_2}Z_{H'^*_i}
\end{aligned}\tag{A9}$$

Suppose $J^{\{T\}}_{first}$ be the first tardy job in sequence G. Thus,

$$\left.\begin{aligned}
&p_{H^G}+p_{F^G}+p_{B^G}+p_{H'^G}+p_{F'^G}+p_{B'^G}-p_{J^{\{T\}}_{first}}<P_{sum}-d_1\\
&p_{H'^*}+p_{H''^*}+p_{B^*}+p_{H^*}+p_{F''^*}+p_{B'^*}-\left(d_1-S_{B'^*_{m_2}}\right)\ge P_{sum}-d_1
\end{aligned}\right\}$$
$$\Rightarrow\quad p_{F'^G}+p_{B'^G}<p_{H''^*}+p_{F''^*}+p_{B'^*}-\left(d_1-S_{B'^*_{m_2}}\right)+p_{J^{\{T\}}_{first}}\tag{A10}$$

Given the fact that jobs in H''^* and F''^* are not selected as tardy or lost by algorithm MWR, we can conclude

$$\frac{w_i}{p_i}\le\frac{w_j}{p_j}\forall i\in\{F'^G\}\quad,\quad\forall j\in\{F''^*,H''^*\}\tag{A11}$$

From (A10) and (A11), we derive

$$\begin{aligned}
Z^{\left[B'^G_{m_1},F'^G_{m_1},\dots,B'^G_2,F'^G_2,B'^G_1,F'^G_1\right]}_{\left(d_1+\sum_{i=1}^{m_1}\left(p_{F'^G_i}+p_{B'^G_i}\right)\right)}&<Z^{\left[J^{\{T\}}_{first},B'^*_{m_2},F''^*_{m_2},\dots,B'^*_2,F''^*_2,B'^*_1,F''^*_1\right]}_{\left(d_1+\sum_{i=1}^{m_2}\left(p_{B'^*_i}+p_{F''^*_i}\right)-\left(d_1-S_{B'^*_{m_2}}\right)+p_{J^{\{T\}}_{first}}\right)}+Z_{H''^*}\\
&\le Z_{B'^*}+Z_{F''^*}+p_{J^{\{T\}}_{first}}.\left(\sum_{i=1}^{m_2}\sum_{j\in B'^*_i}w_j+\sum_{i=1}^{m_2}\sum_{j\in F''^*_i}w_j\right)+Z_{H''^*}\\
&\le 2Z_{B'^*}+2Z_{F''^*}+Z_{H''^*}
\end{aligned}\tag{A12}$$

The last inequality in Eq. (A12) is derived from the following relations and the fact that in each iteration, algorithm MWR checks if there exists a job filling the remaining tardiness period.

$$p_{J_{first}^{\{T\}}} \leq C_j^* - d_1 \qquad \forall j \in \{B'^*, F''^*\} \tag{A13}$$

$$p_{J_{first}^{\{T\}}} \cdot \left(\sum_{i=1}^{m_2} \sum_{j \in B'^*_i} w_j + \sum_{i=1}^{m_2} \sum_{j \in F''^*_i} w_j \right) \leq \sum_{i=1}^{m_2} \sum_{j \in B'^*_i} \left[w_j (C_j^* - d_1) \right] + \sum_{i=1}^{m_2} \sum_{j \in F''^*_i} \left[w_j (C_j^* - d_1) \right] = Z_{B'^*} + Z_{F''^*} \tag{A14}$$

Now, by summarizing (A4), (A9), and (A12), we get

$$\left. \begin{aligned} Z_{\{L\}}^G &\leq Z_{H'^*} + Z_{H''^*} + 2Z_{B^*} \\ Z_{\{T\}}^G &\leq Z_{H'^*} + Z_{H''^*} + 2Z_{B'^*} + 2Z_{F''^*} \end{aligned} \right\} \Rightarrow Z_{\{L\}}^G + Z_{\{T\}}^G \leq 2Z_{H'^*} + 2Z_{H''^*} + 2Z_{B'^*} + 2Z_{F''^*} + 2Z_{B^*}$$
$$\Rightarrow Z^G \leq 2Z^*$$

If the first lost job, $J_{first}^{\{L\}}$, fills the entire tardiness period, then no tardy job is created by algorithm MWR. In this case, job $J_{first}^{\{L\}}$ has the smallest weight among unscheduled jobs and sequence $\widehat{G}$ is the related sequence. The main sequence in MWR is denoted by G and it has already been proved that it gives a worst case ratio of 2. Step 7 of the algorithm selects the best of sequences G and $\widehat{G}$ and hence, the final result of MWR is a 2-approximation which completes the proof. ■