Information Processing Letters 112 (2012) 72–76

Contents lists available at SciVerse ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

A note on a single machine scheduling problem with generalized total tardiness objective function

Evgeny R. Gafarov [a], Alexander A. Lazarev [a,b], Frank Werner [c,*]

[a] Institute of Control Sciences of the Russian Academy of Sciences, Profsoyuznaya st. 65, 117997 Moscow, Russia
[b] Lomonosov Moscow State University, Higher School of Economics, Moscow Institute of Physics and Technology, Russia
[c] Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, PSF 4120, 39016 Magdeburg, Germany

ARTICLE INFO

Article history:
Received 22 April 2010
Received in revised form 26 September 2011
Accepted 14 October 2011
Available online 20 October 2011
Communicated by F.Y.L. Chin

Keywords:
Scheduling
Single machine
Total tardiness
Number of tardy jobs
Total late work
Pseudo-polynomial algorithm
Graphical algorithm

ABSTRACT

In this note, we consider a single machine scheduling problem with generalized total tardiness objective function. A pseudo-polynomial time solution algorithm is proposed for a special case of this problem. Moreover, we present a new graphical algorithm for another special case, which corresponds to the classical problem of minimizing the weighted number of tardy jobs on a single machine. The latter algorithm improves the complexity of an existing pseudo-polynomial algorithm by Lawler. Computational results are presented for both special cases considered.

1. Introduction

Two classical single machine scheduling problems are the problem of minimizing total tardiness and the problem of minimizing the number of tardy jobs which can be formulated as follows.

We are given a set $N = \{1, 2, \ldots, n\}$ of n independent jobs that must be processed on a single machine. Job preemption is not allowed. The machine can handle only one job at a time. All jobs are assumed to be available for processing at time 0. For each job $j \in N$, a processing time $p_j > 0$ and a due date d_j are given.

A feasible solution is described by a permutation $\pi = (j_1, j_2, \ldots, j_n)$ of the jobs of the set N from which the corresponding schedule can be uniquely determined by starting each job as early as possible. Let $C_{j_k}(\pi) = \sum_{l=1}^{k} p_{j_l}$ be the completion time of job j_k in schedule π. If $C_j(\pi) > d_j$, then job j is tardy and we have $U_j = 1$, otherwise $U_j = 0$. If $C_j(\pi) \leqslant d_j$, then job j is said to be on-time. Moreover, let $T_j(\pi) = \max\{0, C_j(\pi) - d_j\}$ be the tardiness of job j in the schedule resulting from the sequence π. For the problem of minimizing the number of tardy jobs $1\|\sum U_j$, the objective is to find an optimal job sequence π^* that minimizes the value $\sum_{j=1}^{n} U_j(\pi)$ and for the problem of minimizing total tardiness $1\|\sum T_j$, the objective is to find an optimal job sequence π^* that minimizes the value $\sum_{j=1}^{n} T_j(\pi)$.

Problem $1\|\sum U_j$ can be solved in $O(n \log n)$ time by Moore's algorithm [1]. Problem $1\|\sum T_j$ is NP-hard in the ordinary sense [2,3]. A pseudo-polynomial dynamic programming algorithm of time complexity $O(n^4 \sum p_j)$ has been proposed by Lawler [4]. A summary of polynomially and pseudo-polynomially solvable special cases can be found e.g. in [5].

* Corresponding author.
E-mail addresses: axel73@mail.ru (E.R. Gafarov), jobmath@mail.ru (A.A. Lazarev), frank.werner@mathematik.uni-magdeburg.de (F. Werner).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2011.10.013

In this note, we consider a generalization of these two problems. In addition to the above data, a quota of tardiness $b_j \geqslant 0$, a coefficient of normal penalty $v_j \geqslant 0$ and a coefficient of abnormal penalty $w_j \geqslant 0$ are given for each job $j \in N$. We define the generalized tardiness as follows:

$$GT_j(\pi) = \begin{cases} 0, & \text{if } C_j(\pi) - d_j \leqslant 0, \\ v_j \cdot (C_j(\pi) - d_j), & \text{if } 0 < C_j(\pi) - d_j \leqslant b_j, \\ w_j, & \text{if } b_j < C_j(\pi) - d_j, \end{cases}$$

where $w_j \geqslant v_j b_j$ for all $j \in N$, and we define

$$F(\pi) = \sum_{j=1}^{n} GT_j(\pi).$$

This means that, from a certain level of tardiness described by parameter b_j for job j, the penalty w_j for exceeding the due date d_j is constant and does no longer depend on the concrete value of the tardiness. The objective is to find an optimal job sequence π^* that minimizes the function $F(\pi)$. We will denote this problem by $1\|\sum GT_j$. It is obvious that this problem is NP-hard. For the special case of $b_j = 0$, we have the classical problem $1\|\sum w_j U_j$ which is NP-hard in the ordinary sense [6]. For problem $1\|\sum w_j U_j$, there exists a pseudo-polynomial solution algorithm with time complexity $O(nd_{\max})$ [7], where $d_{\max}$ is the maximal due date of the jobs. Moreover, it is easy to show that already the special case of problem $1\|\sum GT_j$ with $b_j \in Z_+$ is also NP-hard.

In this note, we consider a special case of the generalized total tardiness problem with

$$b_j = p_j, \qquad v_j = 1, \qquad w_j = p_j, \tag{1}$$

i.e., $GT_j(\pi) = \min\{\max\{0, C_j(\pi) - d_j\}, p_j\}$ for all $j \in N$. This problem corresponds to the minimization of late work considered e.g. in [8,9]. In [9], a pseudo-polynomial algorithm of complexity $O(nUB)$ has been given, where UB denotes an upper bound on the total late work. An excellent overview of recent developments on problems with total late work criteria has been given by Sterna [10]. In Section 3, we give a pseudo-polynomial algorithm with time complexity $O(nd_{\max})$ for this special case which can be realized by a graphical algorithm in a more efficient way. In Section 3, we present another pseudo-polynomial algorithm with time complexity $O(nd_{\max})$ for the problem $1\|\sum w_j U_j$ and its graphical modification, which improves the running time and the complexity of the latter algorithm. Some computational results with the graphical variants of the two algorithms are presented in Section 4.

2. A solution algorithm for the special case (1)

In this section, we present an exact pseudo-polynomial algorithm for the special case (1).

Lemma 1. *There exists an optimal job sequence π for the special case* (1) *that can be represented as a concatenation* (G, H)*, where all jobs $j \in H$ are tardy and $GT_j(\pi) = p_j$. For all jobs $i \in G$, we have $0 \leqslant GT_i(\pi) < p_i$. All jobs from the set G are processed in EDD* (*earliest due date*) *order and all jobs from the set H are processed in LDD* (*last due date*) *order.*

Proof. 1) Assume that there exists an optimal job sequence $\pi^* = (\pi_1, j, \pi_2)$. If $GT_j(\pi) = p_j$, then sequence $\pi' = (\pi_1, \pi_2, j)$ is optimal, too. Thus, there exists an optimal sequence of the type $\pi = (G, H)$, where all jobs $j \in H$ are tardy and $GT_j(\pi) = p_j$. For all jobs $i \in G$, we have $0 \leqslant GT_i(\pi) < p_i$.

2) We consider an optimal job sequence $\pi = (G, H)$, where all jobs $j \in H$ are tardy and $GT_j(\pi) = p_j$. For all jobs $i \in G$, we have $0 \leqslant GT_i(\pi) < p_i$. Now we prove that all jobs $i \in G$ are processed according to EDD order.

Assume that there exists an optimal sequence $\pi = (\pi_1, \alpha, \beta, \pi_2)$, where jobs $\alpha, \beta \in G$ and $d_\alpha > d_\beta$. Then inequalities $C_\alpha(\pi) - d_\alpha < p_\alpha$ and $C_\beta(\pi) - d_\beta < p_\beta$ hold.

We consider sequence $\pi' = (\pi_1, \beta, \alpha, \pi_2)$. Denote $C = C_\beta(\pi) = C_\alpha(\pi')$. Then

$$\begin{aligned} F(\pi) - F(\pi') &= \big(GT_\alpha(\pi) - GT_\alpha(\pi')\big) \\ &\quad + \big(GT_\beta(\pi) - GT_\beta(\pi')\big) \\ &= -\min\big\{p_\alpha, \max\{0, C - d_\alpha\}\big\} \\ &\quad + \min\big\{p_\alpha, \max\{0, C - d_\beta\}\big\} \geqslant 0 \end{aligned}$$

and sequence π' is optimal as well.

3) We consider an optimal job sequence $\pi = (G, H)$, where all jobs $j \in H$ are tardy and $GT_j(\pi) = p_j$. For all jobs $i \in G$, we have $0 \leqslant GT_i(\pi) < p_i$. Now, we prove that all jobs $j \in H$ can be processed in an LDD order in an optimal sequence. For all jobs $j \in H$, we have $d_j \leqslant \sum_{l=1}^{n} p_l - \sum_{k \in H} p_k$, otherwise, if $d_j > \sum_{l=1}^{n} p_l - \sum_{k \in H} p_k$, then sequence $\pi' = (G, j, H \setminus \{j\})$ is better, and we have a contradiction. Therefore, the jobs from H can be processed in any order. □

The following algorithm is based on Lemma 1.

Algorithm 1.

1. Enumerate the jobs according to non-increasing due dates: $d_1 \geqslant d_2 \geqslant \cdots \geqslant d_n$.
2. $\pi_1(t) := (1)$, $F_1(t) := \min\{p_1, \max\{0, p_1 + t - d_1\}\}$ for all $t \in Z \cap [0, \sum_{j=2}^{n} p_j]$;
3. FOR $l := 2$ TO n DO
   FOR $t := 0$ TO $\sum_{j=l+1}^{n} p_j$ $(t \in Z)$ DO
   $\pi^1 := (l, \pi_{l-1}(t + p_l))$, $\pi^2 := (\pi_{l-1}(t), l)$;
   $F(\pi^1) := \min\{p_l, \max\{0, p_l + t - d_l\}\} + F_{l-1}(t + p_l)$;
   $F(\pi^2) := F_{l-1}(t) + \min\{p_l, \max\{0, \sum_{j=1}^{l} p_j + t - d_l\}\}$;
   $F_l(t) := \min\{F(\pi^1), F(\pi^2)\}$;
   $\pi_l(t) := \arg\min\{F(\pi^1), F(\pi^2)\}$.
4. $\pi_n(0)$ is an optimal sequence with the objective function value $F_n(0)$.

$\pi_l(t)$ represents the best partial sequence of the jobs $1, 2, \ldots, l$ when the first job starts at time t, and $F_l(t)$ denotes the corresponding generalized total tardiness.

Theorem 1. *Algorithm* 1 *constructs an optimal sequence for the special case* (1) *in $O(n \sum p_j)$ time.*

Proof. We prove the theorem indirectly. Assume that there exists an optimal sequence of the form $\pi^* = (EDD, LDD)$, where $F(\pi^*) < F(\pi_n(0)) = F_n(0)$.

Let $\pi' := \pi^*$. For each $l = 1, 2, \ldots, n$, we successively consider the part $\bar{\pi}_l \in \pi'$, $\{\bar{\pi}_l\} = \{1, \ldots, l\}$ of the sequence. Let $\pi' = (\pi_\alpha, \bar{\pi}_l, \pi_\beta)$. If $\bar{\pi}_l \neq \pi_l(t = \sum_{i\in\pi_\alpha} p_i)$ (for the notation, see the last row in Step 3 of Algorithm 1), then $\pi' := (\pi_\alpha, \pi_l(\sum_{i\in\pi_\alpha} p_i), \pi_\beta)$. It is obvious that $F((\pi_\alpha, \bar{\pi}_l, \pi_\beta)) \geqslant F((\pi_\alpha, \pi_l(\sum_{i\in\pi_\alpha} p_i), \pi_\beta))$. Analogously, step by step, we modify the partial sequences $\bar{\pi}_l$ corresponding to the subsequent values l. At the end, we have $F(\pi^*) \geqslant F(\pi') = F_n(0)$. Thus, sequence $\pi_n(0)$ is also optimal.

Obviously, the time complexity of Algorithm 1 is equal to $O(n \sum p_j)$. □

We can improve the running time of Algorithm 1, if for each $l = 1, 2, \ldots, n$, we consider only the interval $[0, d_l]$ instead of $[0, \sum_{j=l+1}^{n} p_j]$ since for each $t \geqslant d_l$, job l is tardy in any partial sequence π_l, where $\pi_l(t)$ represents a partial sequence of the jobs $1, 2, \ldots, l$ when the first job starts at time t. Moreover, for $t \geqslant d_l$, we have $GT_l = p_l$. Thus, the partial sequence $\pi^2 := (\pi_{l-1}(t), l)$ (for the notation, see the first row in Step 3 of Algorithm 1) is optimal. The time complexity of the modified Algorithm 1 is equal to $O(nd_{\max})$.

For a practical realization of the algorithm, we can use the idea from [11,12] resulting in a graphical algorithm with the same complexity but often reducing the running time of Algorithm 1 (a brief sketch of this graphical approach is described for another special case in Section 3, where it reduces the complexity).

We also note that the well-known algorithm by Lawler [4] for problem $1\|\sum T_j$ with time complexity $O(n^4 \sum p_j)$ is not exact for the special case (1) since the known rule by Emmons (if $d_i < d_j$, $p_i < p_j$, then $i \to j$) [4] does not hold.

3. A graphical algorithm for the special case $1\|\sum w_j U_j$

We can propose a similar algorithm for the special case $1\|\sum w_j U_j$. The following lemma is an immediate consequence from [1,7].

Lemma 2. *There exists an optimal job sequence π for problem $1\|\sum w_j U_j$ that can be represented as a concatenation (G, H), where all jobs $j \in H$ are tardy and all jobs $i \in G$ are on-time. All jobs from the set G are processed in EDD (earliest due date) order and all jobs from the set H are processed in LDD (last due date) order.*

Note that in an optimal sequence, the on-time jobs can be scheduled in EDD order while the tardy jobs can be scheduled in arbitrary order [1,7]. The following algorithm for problem $1\|\sum w_j U_j$ is based on Lemma 2.

Algorithm 2.

1. Enumerate the jobs according to non-increasing due dates: $d_1 \geqslant d_2 \geqslant \cdots \geqslant d_n$.
2. $\pi_1(t) := (1)$. For each $t \in Z \cap [0, \sum_{j=2}^{n} p_j]$, we compute: if $p_1 + t - d_1 > 0$, then $F_1(t) := w_1$ else $F_1(t) := 0$;
3. FOR $l := 2$ TO n DO
   FOR $t := 0$ TO $\sum_{j=l+1}^{n} p_j$ $(t \in Z)$ DO
   $\pi^1 := (l, \pi_{l-1}(t + p_l))$, $\pi^2 := (\pi_{l-1}(t), l)$;
   If $p_l + t - d_l > 0$, then $F(\pi^1) := w_l + F_{l-1}(t + p_l)$ else $F(\pi^1) := F_{l-1}(t + p_l)$;
   If $\sum_{j=1}^{l} p_j + t - d_l > 0$, then $F(\pi^2) := F_{l-1}(t) + w_l$ else $F(\pi^2) := F_{l-1}(t)$;
   $F_l(t) := \min\{F(\pi^1), F(\pi^2)\}$;
   $\pi_l(t) := \arg\min\{F(\pi^1), F(\pi^2)\}$.
4. $\pi_n(0)$ is an optimal sequence with the objective function value $F_n(0)$.

Analogously to the proof of Theorem 1, we can prove the following theorem.

Theorem 2. *Algorithm 2 constructs an optimal sequence for problem $1\|\sum w_j U_j$ in $O(n \sum p_j)$ time.*

Analogously to the modification of Algorithm 1, we can propose a modification of Algorithm 2, where for each $l = 1, 2, \ldots, n$, we consider only the interval $[0, d_l - p_l]$ instead of $[0, \sum_{j=l+1}^{n} p_j]$ since for each $t > d_l - p_l$, job l is tardy in any partial sequence $\pi_l(t)$ and the partial sequence $\pi^2 := (\pi_{l-1}(t), l)$ is optimal. Thus, the time complexity of the modified Algorithm 2 is equal to $O(nd_{\max})$.

Using the idea of the graphical approach from [11,12], we obtain an exact algorithm, which improves the running time and the complexity of Algorithm 2. The idea of such a modified graphical algorithm is as follows.

In each step of the graphical algorithm, we store function $F_l(t)$ in tabular form as given in Table 1, where $t_1 < t_2 < \cdots < t_m$ and $W_1 < W_2 < \cdots < W_m$.

The above data means the following. For each value $t \in (t_k, t_{k+1}]$, $1 \leqslant k < m$, we have an optimal partial sequence $\pi_k = (G, H) = (EDD, LDD)$ and the objective function value $F_l(t) = W_k = \sum_{j\in H} w_j$. The points t_k are called the *break points*, i.e., we have $F_l(t') < F_l(t'')$ for $t' \leqslant t_k < t''$.

In the next step $l + 1$, we transform function $F_l(t)$ into functions $F^1(t)$ and $F^2(t)$ according to Step 3 of Algorithm 2 in $O(m)$ operations. In each of the tables for $F^1(t)$ and $F^2(t)$, we have at most $m + 1$ break points. Then we compute a new table of the function $F_{l+1}(t) = \min\{F^1(t), F^2(t)\}$ in $O(m)$ operations. In the new table of function $F_{l+1}(t)$, there are at most $2m + 2$ break points (usually, this number is smaller). In fact, we do not consider all points t from the interval $[0, \min\{d_l - p_l, \sum_{j=l+1}^{n} p_j\}]$, but only points from the interval in which the objective function value changes.

In the graphical algorithm, in each step $l = 1, 2, \ldots, n$, we have to consider at most $\min\{2^l, d_l - p_l, \sum_{j=l+1}^{n} p_j, \sum_{j=1}^{l} w_j, F_{opt}\}$ break points. Thus, the time complexity of

Table 1
Function $F_l(t)$.

t	t_1	t_2	...	t_m
$F_l(t)$	W_1	W_2	...	W_m
Optimal partial sequence	π_1	π_2	...	π_m

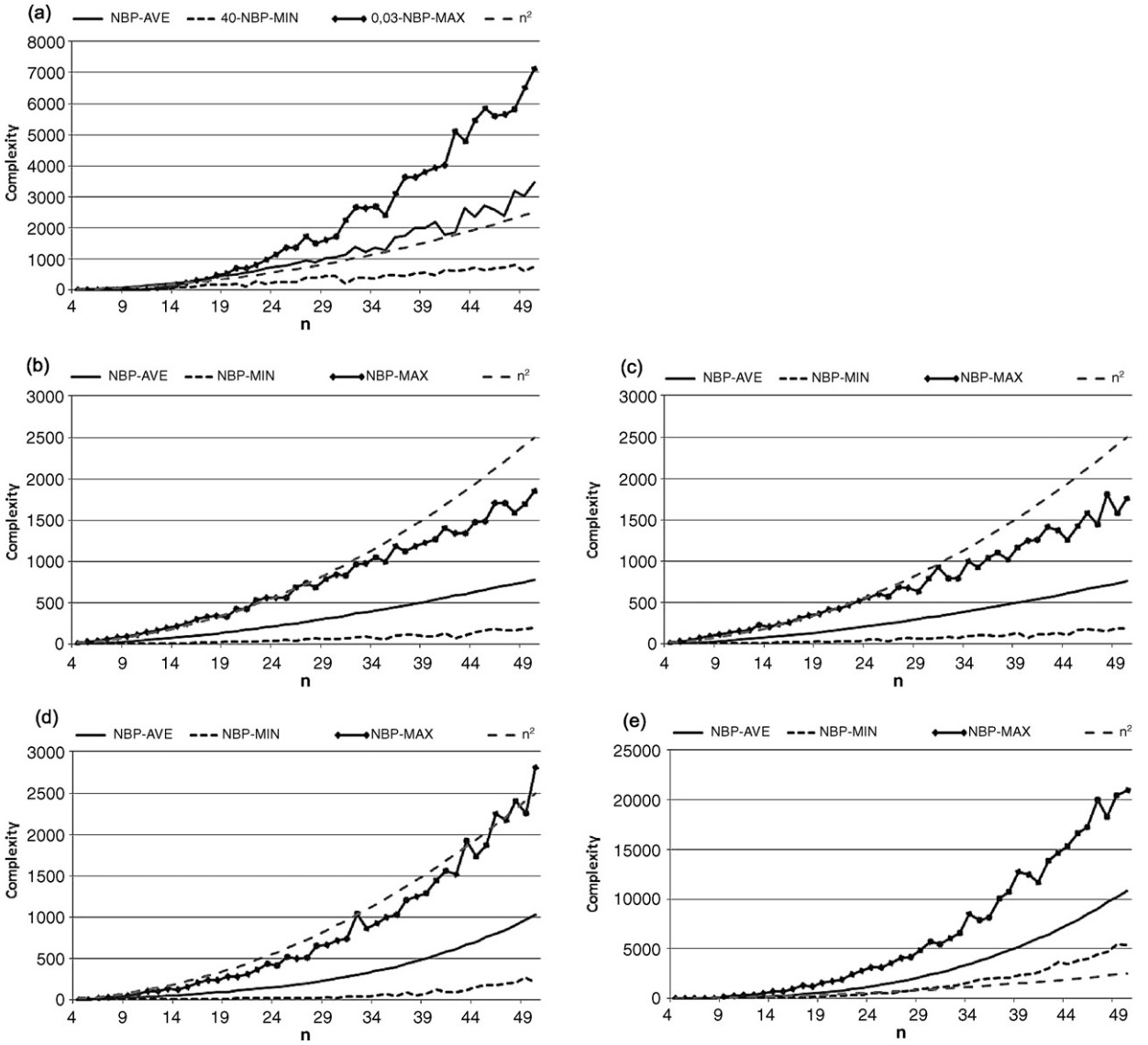

Fig. 1. Computational results.

the graphical algorithm is $O(n \min\{d_{\max}, 2^n, F_{opt}\})$, where F_{opt} is the optimal objective function value.

Note that the running time of the algorithm is the same for the instance with the parameters $\{p_j, d_j, w_j, j \in N\}$ and for the instance with the parameters $\{p'_j = p_j \times 10^6 \pm 1, d_j \times 10^6, w_j\}$. This is in contrast to usual dynamic programming, where the running time for the second instance is larger than for the first one. Moreover, the graphical algorithm can also solve instances with $p_j \notin Z$.

Finally, we only note that a numerical example for illustrating the graphical algorithm for the single machine total tardiness *maximization* problem has been presented in [12].

4. Computational results

We have implemented the versions of the graphical algorithm for both problems.

For the special case (1) of the problem $1\|\sum GT_j$, we have used the following set of instances, based on Potts and Van Wassenhove's scheme [13]. The processing times are randomly generated from the interval $[1, 100]$, and the due dates are randomly generated from the interval

$$\left[\sum_{j=1}^{n} p_j(1 - TF - RDD/2), \sum_{j=1}^{n} p_j(1 - TF + RDD/2)\right].$$

The parameters *TF* (average tardiness factor) and *RDD* (relative range of due dates) are taken from the set $\{0.2, 0.4, 0.6, 0.8, 1\}$. For each combination of the parameters (TF, RDD) and $n \in \{4, 5, 6, \ldots, 50\}$, 100 instances were generated, i.e., 2500 instances for each value of $n \in \{4, 5, 6, \ldots, 50\}$. For each instance, we have computed the minimal (NBP-MIN), average (NBP-AVE) and maximal (NBP-MAX) number of break points. The results are summarized in Fig. 1(a). The results show that NBP-AVE is sub-

stantially smaller than NBP-MAX and that NBP-AVE grows approximately until 3500 for the large instances. Roughly speaking, NBP-AVE grows approximately as n^2.

For the problem $1\|\sum w_jU_j$, we have run two sets of instances for testing the graphical variant of Algorithm 2. The first set is as follows. The processing times are randomly generated from the interval $[p_{\min}, p_{\max}]$, the weights are randomly generated from the interval $[1, w_{\max}]$, and the due dates are randomly generated from the interval $[p_j, p_j + m_{\max}]$. The following values of the parameters were used:

$(p_{\min}, p_{\max})$: $(0, 100), (25, 75)$
$w_{\max}$: $1, 10, 100$
$m_{\max}$: $50, 200, 350, 500, 650$

For each combination of the parameters and $n \in \{4, 5, \ldots, 50\}$, a series of 2500 test instances were generated.

The second set is generated as follows. The processing times are randomly generated from the interval $[0, 100]$, the weights are randomly generated from the interval $[1, w_{\max}]$, and the due dates are randomly generated from the interval $[p_j, p_j + Kn]$. The following values of parameters were used:

$w_{\max}$: $10, 99$
K: $1, 5, 10, 20$

For each combination of the parameters and $n \in \{4, 5, \ldots, 50\}$, a series of 2500 test instances has been generated.

For each instance, we have computed the minimal (NBP-MIN), average (NBP-AVE) and maximal (NBP-MAX) number of break points. Fig. 1(b) (instances with $p_{\min} = 25, p_{\max} = 75, w_{\max} = 10, m_{\max} = 50$) and Fig. 1(c) (instances with $p_{\min} = 25, p_{\max} = 75, w_{\max} = 10, m_{\max} = 50$) present the results for two representative variants of the first set, and Fig. 1(d) (instances with $w_{\max} = 10, K = 1$) and Fig. 1(e) (instances with $w_{\max} = 10, K = 10$) present the results for two representative variants of the second set. For the instances of the first type presented here, the number of break points only moderately increases, and NBP-AVE is lower than 1000 even for the large problems. For both variants, even NBP-MAX grows less that n^2. Figs. 1(d) and 1(e) demonstrate the influence of the parameter K on the number of break points. For $K = 10$, the number of break points is roughly ten times as large as for $K = 1$. While for the instances with $K = 1$, NBP-MAX grows approximately as n^2, the number of break points is usually larger than n^2 for the instances with $K = 10$.

From our detailed results we report the following additional observations. It follows that, as expected, the lengths of the intervals considered has the strongest influence on the number of break points. The largest numbers of break points were observed for instances with $m_{\max} = 650$ in the first set and for instances with $K = 20$ in the second set. We have also observed that in the first set of instances in the case of $w_{\max} = 1$, i.e., $w_j = 1$ for all $j \in N$, the number of break points may substantially differ from those for the other test series. For instance, the average number of break points in the series with $w_{\max} = 1, m_{\max} = 50$ is much smaller than the average number of break points for the case $w_{\max} = 10, m_{\max} = 50$. When $m_{\max}$ is large, e.g. when $m_{\max} = 500$ or $m_{\max} = 650$, the number of break points in the series with $w_{\max} = 1$ is considerably smaller than in the series with other values of $w_{\max}$.

References

[1] J.M. Moore, An n job, one machine sequencing algorithm for minimizing the number of late jobs, Management Sci. 15 (1) (1968) 102–109.
[2] J. Du, J.Y.-T. Leung, Minimizing total tardiness on one processor is NP-hard, Math. Oper. Res. 15 (1990) 483–495.
[3] A.A. Lazarev, E.R. Gafarov, Special case of the single-machine total tardiness problem is NP-hard, J. Comput. Systems Sci. Internat. 45 (3) (2006) 450–458.
[4] E.L. Lawler, A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness, Ann. Discrete Math. 1 (1977) 331–342.
[5] A.A. Lazarev, F. Werner, Algorithms for special cases of the single machine total tardiness problem and an application to the even–odd partition problem, Math. Comput. Modelling 49 (9–10) (2009) 2061–2072.
[6] J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems, Ann. Discrete Math. 1 (1977) 343–362.
[7] E.L. Lawler, J.M. Moore, A functional equation and its application to resource allocation and sequencing problems, Management Sci. 16 (1969) 77–84.
[8] J. Blazewicz, Scheduling preemptive tasks on parallel processors with information loss, Tech. Sci. Inform. 3 (6) (1984) 415–420.
[9] C.N. Potts, L.N. van Wassenhove, Single machine scheduling to minimize total late work, Oper. Res. 40 (3) (1991) 586–595.
[10] M. Sterna, A survey of scheduling problems with late work criteria, Omega 39 (2) (2011) 120–129.
[11] A.A. Lazarev, F. Werner, A graphical realization of the dynamic programming method for solving NP-hard combinatorial problems, Comput. Math. Appl. 58 (4) (2009) 619–631.
[12] E.R. Gafarov, A.A. Lazarev, F. Werner, A polynomial time graphical algorithm for maximizing total tardiness on a single machine, preprint 12/10, FMA, Otto-von-Guericke-Universität Magdeburg, 2010.
[13] C.N. Potts, L.N. van Wassenhove, A decomposition algorithm for the single machine total tardiness problem, Oper. Res. Lett. 1 (5) (1982) 177–181.