Transforming a pseudo-polynomial algorithm for the single machine total tardiness maximization problem into a polynomial one

Evgeny R. Gafarov, Alexander A. Lazarev & Frank Werner

Annals of Operations Research

ISSN 0254-5330

Ann Oper Res
DOI 10.1007/s10479-011-1055-4

Springer

Ann Oper Res
DOI 10.1007/s10479-011-1055-4

Transforming a pseudo-polynomial algorithm for the single machine total tardiness maximization problem into a polynomial one

Evgeny R. Gafarov · Alexander A. Lazarev · Frank Werner

Abstract We consider the problem of maximizing total tardiness on a single machine, where the first job starts at time zero and idle times between the processing of jobs are not allowed. We present a modification of an exact pseudo-polynomial algorithm based on a graphical approach, which has a polynomial running time. This result settles the complexity status of the problem under consideration which was open.

Keywords Scheduling · Single machine problems · Maximization problems · Total tardiness · Polynomial algorithm

1 Introduction

Most papers in scheduling theory deal with problems, where a specific objective function has to be minimized. For instance, the minimization of makespan is a very popular optimization criterion. In the case when a sum function is considered, often total completion time, total tardiness or the number of tardy jobs has to be minimized. In this paper, we consider a single machine problem with the *opposite* criterion, namely, we consider the maximization of total tardiness.

In detail, the problem under consideration 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.

E.R. Gafarov · A.A. Lazarev
Institute of Control Sciences of the Russian Academy of Sciences, Profsoyuznaya st. 65, 117997 Moscow, Russia

E.R. Gafarov
e-mail: axel73@mail.ru

A.A. Lazarev
e-mail: jobmath@mail.ru

F. Werner (✉)
Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, PSF 4120, 39016 Magdeburg, Germany
e-mail: frank.werner@mathematik.uni-magdeburg.de

Published online: 10 January 2012

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.

We assume that each feasible schedule starts at time 0 and does not have any idle time between the processing of the jobs (note that the maximization problem considered in this paper would be trivial when allowing inserted idle times, since the maximal objective function value can become arbitrarily large in this case). Thus, 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 the schedule resulting from the sequence π. If $C_j(\pi) > d_j$, then job j is tardy. If $C_j(\pi) \leq d_j$, then job j is 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 sequence π. The objective is to find an optimal job sequence π^* that maximizes total tardiness, i.e., $F(\pi) = \sum_{j=1}^{n} T_j(\pi)$. We denote this problem by $1(\textit{no-idle})||\max \sum T_j$. In addition, let $E_j(\pi) = \max\{0, d_j - C_j(\pi)\}$, $U_j(\pi) = 1$ if $T_j(\pi) > 0$ and $U_j(\pi) = 0$ otherwise.

In the following, the notation $\{\pi\}$ denotes the set of jobs contained in the sequence π. The notation $i \in \pi$ means $i \in \{\pi\}$. For the sequence $\pi = (\pi_1, i, \pi_2)$, the notation $\pi \setminus \{i\}$ means (π_1, π_2).

On the one hand, the investigation of a particular problem with the *maximum* criterion is an important theoretical task. Algorithms for such a problem with the maximum criterion can be used to cut bad sub-problems in the branching tree of branch-and-bound algorithms. So the maximal number of tardy jobs determined by the algorithm for the problem $1(\textit{no-idle})||\max \sum U_j$ from Gafarov et al. (2010a) can be used to reduce the search for an optimal solution of the problem $1(\textit{no-idle})||\max \sum T_j$. Moreover, we can use the value $\max \sum T_j$ for the computation of an upper bound on the optimum in the problem $1||\min(\alpha \sum E_j + \beta \sum T_j)$, i.e., if $\alpha > \beta$, then we can 'fix' the maximal value $\beta \sum T_j$ and search for an optimal schedule for the criterion $\min \sum E_j$. In Aloulou and Artigues (2010), maximization problems were used for solving bi-criteria problems by a branch and bound method. In addition, it is interesting whether polynomially solvable cases of the minimization problem are also easy when maximizing the corresponding objective function, or vice versa.

On the other hand, such problems have also practical interpretations and applications. For example, an installation team has to mount wind turbines in various regions of a large country. The required time for mounting the turbines in a particular region (which is a job) depends on the number of turbines (i.e., each job j has a particular processing time p_j) and is not dependent on weather or climatic conditions. However, weather influences some costs (e.g., covering of snow may increase fuel consumption, the salary of the workers and/or the cost of accommodation for the workers which might be higher in winter). Such additional costs begin to decrease after the beginning of a particular season. For each region of the country (i.e., for each corresponding job j), it is known in which week this season can be expected to start (interpreted as a due date d_j). All jobs must be executed in the time interval $[0, \sum p_j]$, where $\sum p_j$ is the total processing time of the jobs. The objective is the minimization of these additional costs. In fact, this optimization criterion can be formulated as $\max \sum \max\{0, S_j - d'_j\}$, where $S_j = C_j - p_j$ and $d'_j = d_j - p_j$, i.e., the problem $1(\textit{no-idle})||\max \sum T_j$ is obtained. Another situation arises when the company is considered as a customer, and one wants to know the worst variant of a schedule, which is computed in a 'black box' (e.g. in a plant).

It is easy to show that each of the problems $1(\textit{no-idle})||\max \sum T_j$ and $1(\textit{no-idle})||\max \sum E_j$ can be polynomially reduced to the other one. Assume that we

have an instance of the problem $1(\textit{no-idle})||\max\sum E_j$ and an optimal job sequence $\pi=(1,2,\ldots,n)$. We construct an instance of the problem $1(\textit{no-idle})||\max\sum T_j$ as follows. The set of jobs and the processing times are the same. In addition, we set $d'_j=\sum_{i=1}^{n}p_i-d_j+p_j$. Then the sequence $\pi'=(n,n-1,\ldots,1)$ is optimal for the above instance of the problem $1(\textit{no-idle})||\max\sum T_j$. We have $C_j(\pi')-p_j=\sum p_i-C_j(\pi)$. Therefore, $C_j(\pi')-d'_j=(\sum p_i-C_j(\pi)+p_j)-(\sum p_i-d_j+p_j)=d_j-C_j(\pi)$.

We note that for the problem $1(\textit{no-idle})||\max\sum E_j$, we can drop the assumption that 'idle times are not allowed'. In addition, we can give the following interpretation for this problem. A worker has to complete a set of jobs. If he completes a job j before the *due date* d_j, then he obtains a bonus which is proportional to d_j-C_j. The objective is to maximize the total bonus. Both the problems $1(\textit{no-idle})||\max\sum E_j$ and $1(\textit{no-idle})||\max\sum T_j$ were mentioned in Lawler and Moore (1969), where a pseudo-polynomial algorithm with time complexity $O(nd_{\max})$ was presented for the more general problem $1(\textit{no-idle})||\max\sum w_jT_j$ (Here w_j is the weight of job $j\in N$ and $d_{\max}$ is the maximal due date). However, the complexity status of this problem was open for a long time, and only in Gafarov et al. (2010b) a proof of NP-hardness of the problem $1(\textit{no-idle})||\max\sum w_jT_j$ was presented. For the problem $1||\min\sum w_jT_j$, a branch and bound algorithm was given in Babu et al. (2004) and a simulated annealing algorithm was presented in Matsuo et al. (1989).

The problem $1||\min\sum T_j$ is NP-hard in the ordinary sense (Du and Leung 1990; Lazarev and Gafarov 2006a). A pseudo-polynomial dynamic programming algorithm of time complexity $O(n^4\sum p_j)$ was proposed by Lawler (1977). The algorithms by Szwarc et al. (1999) can solve special instances (Potts and Van Wassenhove 1982) of this problem for $n\le 500$ jobs. A summary of polynomially and pseudo-polynomially solvable special cases can be found, e.g., in Lazarev and Werner (2009b). A pseudo-polynomial time algorithm for the problem $1(\textit{no-idle})||\max\sum T_j$ was presented in Gafarov et al. (2010a). In Gafarov et al. (2010a, 2010c), Aloulou et al. (2004, 2007), the complexity of single machine scheduling problems with classical and maximum optimization criteria was investigated. In Posner (1990), the reducibility among single machine weighted completion time scheduling problems including both minimization and maximization problems was studied. The constraints considered include release dates, deadlines, and continuous machine processing.

For a practical realization of some pseudo-polynomial algorithms, one can use the idea from Lazarev and Werner (2009a). This modification of pseudo-polynomial algorithms for combinatorial problems with Boolean variables (e.g. problems, where a job can be on-time or tardy, or an item is put into the knapsack or not) is called a *graphical approach*.

In this paper, we present such a graphical modification of a pseudo-polynomial algorithm for the problem $1(\textit{no-idle})||\max\sum T_j$, and we prove that the suggested graphical algorithm has a polynomial time complexity. This is in contrast to the pseudo-polynomial complexity of the best known exact algorithm for the problem of minimizing total tardiness on a single machine.

The rest of this paper is organized as follows. In Sect. 2, an exact pseudo-polynomial algorithm for the problem $1(\textit{no-idle})||\max\sum T_j$ is presented. A modification of this pseudo-polynomial algorithm for the problem $1(\textit{no-idle})||\max\sum T_j$ and the proof of its polynomial running time are given in Sect. 3. In Sect. 4, a numerical example for illustrating this algorithm is presented.

2 A pseudo-polynomial time algorithm for problem $1(\textit{no-idle})||\max\sum T_j$

In this section, we present a property of an optimal sequence and an exact algorithm for the problem under consideration which are the base for the modification in Sect. 3.

Lemma 1 (Gafarov et al. 2010a) *There exists an optimal job sequence π for the problem $1(\textit{no-idle})||\max\sum T_j$ 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 SPT (shortest processing time) order and all jobs from the set H are processed in LPT (longest processing time) order.*

Assume that the jobs are numbered as follows: $p_1 \geq p_2 \geq \cdots \geq p_n$. As a corollary from Lemma 1, for each $l \in \{1, 2, \ldots, n\}$ there exists an optimal schedule in which all jobs $j \in \{1, 2, \ldots, l\}$ are processed from time t one by one, and there is no job $i \in \{l+1, l+2, \ldots, n\}$ which is processed between these jobs. Thus, we can present a dynamic programming algorithm based on Lemma 1. At each stage l, $1 \leq l \leq n$, we construct a best partial sequence $\pi_l(t)$ for the set of jobs $\{1, 2, \ldots, l\}$ and for each possible starting time t of the first job (which represents a possible state in the dynamic programming algorithm). $F_l(t)$ denotes the total tardiness value of the job sequence $\pi_l(t)$. $\Phi^1(t)$ and $\Phi^2(t)$ are temporary functions, which are used to compute $F_l(t)$.

Algorithm 1

1. Number the jobs according to non-increasing processing times: $p_1 \geq p_2 \geq \cdots \geq p_n$. If $p_i = p_{i+1}$, then $d_i \geq d_{i+1}$;
2. FOR $t := 0$ TO $\sum_{j=2}^{n} p_j$ DO
   $\pi_1(t) := (1)$, $F_1(t) := \max\{0, p_1 + t - d_1\}$;
3. FOR $l := 2$ TO n DO
   FOR $t := 0$ TO $\sum_{j=l+1}^{n} p_j$ DO
   $\pi^1 := (l, \pi_{l-1}(t + p_l))$, $\pi^2 := (\pi_{l-1}(t), l)$;
   $\Phi^1(t) := \max\{0, p_l + t - d_l\} + F_{l-1}(t + p_l)$;
   $\Phi^2(t) := F_{l-1}(t) + \max\{0, \sum_{j=1}^{l} p_j + t - d_l\}$;
   IF $\Phi^1(t) > \Phi^2(t)$ THEN $F_l(t) := \Phi^1(t)$ and $\pi_l(t) := \pi^1$,
   ELSE $F_l(t) := \Phi^2(t)$ and $\pi_l(t) := \pi^2$;
4. $\pi_n(0)$ is an optimal schedule with the objective function value $F_n(0)$.

Theorem 1 *Algorithm 1 constructs an optimal job sequence in $O(n\sum p_j)$ time.*

Proof We prove the theorem indirectly. Assume that there exists an optimal schedule of the form $\pi^* = (G, H)$, 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 SPT (shortest processing time) order and all jobs from the set H are processed in LPT (longest processing time) order. Assume that $F(\pi^*) > F(\pi_n(0)) = F_n(0)$.

Let $\pi' := \pi^*$. For each $l = 1, 2, \ldots, n$, we successively consider the part $\bar{\pi}_l \in \pi'$ of the schedule with $\{\bar{\pi}_l\} = \{1, 2, \ldots, l\}$. Let $\pi' = (\pi_\alpha, \bar{\pi}_l, \pi_\beta)$ and $t^* = \sum_{i \in \pi_\alpha} p_i$. If $\bar{\pi}_l \neq \pi_l(t^*)$, then $\pi' := (\pi_\alpha, \pi_l(t^*), \pi_\beta)$. It is obvious that $F((\pi_\alpha, \bar{\pi}_l, \pi_\beta)) \leq F((\pi_\alpha, \pi_l(t^*), \pi_\beta))$. Applying this procedure to subsequent values l, we have $F(\pi^*) \leq F(\pi') = F_n(0)$ at the end. Thus, the schedule $\pi_n(0)$ is also optimal.

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

3 A polynomial time algorithm for problem $1(\textit{no-idle})||\max\sum T_j$

In this section, we derive a new graphical algorithm for this problem which is based on an idea from Lazarev and Werner (2009a). The graphical algorithm is a modification of Algorithm 1, in which function $F_l(t)$ is defined for any $t \in (-\infty, +\infty)$ (not only for integer t).

Table 1 Function $F_l(t)$

k	1	2	...	w_l+1
Interval k	$(-\infty, t_l^1]$	$(t_l^1, t_l^2]$	...	$(t_l^{w_l}, +\infty)$
b_l^k	0	b_l^2	...	$b_l^{w_l+1}$
u_l^k	0	u_l^2	...	$u_l^{w_l+1}$
π_l^k	π_l^1	π_l^2	...	$\pi_l^{w_l+1}$

However, we need to compute these values only at the *break* points separating intervals in which function $F_l(t)$ is a linear function of the form $F_l(t) = F_l^k(t) = u_l^k \cdot (t - t_l^{k-1}) + b_l^k$. In Theorem 2, we are to prove that $F_l(t)$ is a continuous piecewise linear convex function whose parameters can be described in a tabular form (like in Table 1). In contrast to Algorithm 1 having a pseudo-polynomial complexity, the number of such points (and of the resulting intervals, respectively) is polynomially bounded in the number of jobs for the problem under consideration, as we show below. Together with the graphical algorithm in this section, we present a numerical example in Sect. 4 to illustrate the graphical algorithm.

In each step of the graphical algorithm, we store the information on function $F_l(t)$ for a number of intervals (characterized by the same best partial sequence and by the same number of tardy jobs) in a tabular form as given in Table 1.

In Table 1, k denotes the number of the current interval, whose values range from 1 to $w_l + 1$ (where the number of intervals $w_l + 1$ is defined for each $l = 1, 2, \ldots, n$), $(t_l^{k-1}, t_l^k]$ is the kth interval (where $t_l^0 = -\infty$, $t_l^{w_l+1} = \infty$), b_l^k, u_l^k are the parameters of the linear function $F_l^k(t)$ defined in the kth interval, and π_l^k is the best sequence of the first l jobs if they are processed from time $t \in (t_l^{k-1}, t_l^k]$.

This data means the following. For each above interval, we store the parameters b_l^k and u_l^k for describing function $F_l(t)$ and the resulting best partial sequence if the first job starts in this interval. For each starting time $t \in (t_l^{k-1}, t_l^k]$ $(t_l^0 = -\infty)$ of the first job, we have a best partial sequence π_l^k of the jobs $1, 2, \ldots, l$ with u_l^k tardy jobs and the function value $F_l(t) = u_l^k \cdot (t - t_l^{k-1}) + b_l^k$ (see Fig. 1), $F_l(t) = 0$ for $t \in (t_l^0, t_l^1]$. Recall that function $F_l(t)$ is defined not only for integers t, but also for real numbers t. For simplicity of the following description, we consider the whole t-axis, i.e., $t \in (-\infty, +\infty)$. In Theorem 2, we prove that this table describes a continuous, piecewise-linear and convex function $F_l(t)$. The points $t_l^1, t_l^2, \ldots, t_l^{w_l}$ are called *break points*, since there is a change from value u_l^{k-1} to u_l^k (which

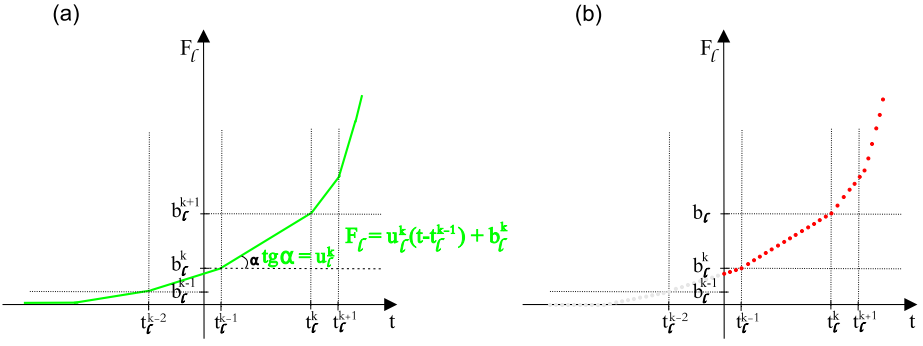

Fig. 1 Function $F_l(t)$ in the graphical algorithm and in Algorithm 1

Table 2 Function $F_1(t)$

k	1	2
Interval k	$(-\infty, d_1 - p_1]$	$(d_1 - p_1, +\infty)$
b_1^k	0	0
u_1^k	0	1
π_1^k	(1)	(1)

Table 3 Function $F_{l-1}(t)$

k	1	2	...	$w_{l-1}+1$
Interval k	$(-\infty, t_{l-1}^1]$	$(t_{l-1}^1, t_{l-1}^2]$	...	$(t_{l-1}^{w_{l-1}}, +\infty)$
b_{l-1}^k	0	b_{l-1}^2	...	$b_{l-1}^{w_{l-1}+1}$
u_{l-1}^k	0	u_{l-1}^2	...	$u_{l-1}^{w_{l-1}+1}$
π_{l-1}^k	π_{l-1}^1	π_{l-1}^2	...	$\pi_{l-1}^{w_{l-1}+1}$

means that the slope of the piecewise-linear function changes). Note that some of the break points t_l^k can be non-integer. For describing each linear segment, we store its slope u_l^k and its function value $b_l^k = F_l(t)$ at the point $t = t_l^{k-1}$.

In the graphical algorithm, the functions $F_l(t)$ reflect the same functional equations as in Algorithm 1, i.e., for each $t \in Z \cap [0, \sum_{j=2}^{n} p_j]$, the function $F_l(t)$ has the same value as in Algorithm 1 (see Fig. 1), but these functions are now defined for any $t \in (-\infty, +\infty)$. As a result, often a large number of integer states is combined into one interval (describing a new state in the resulting algorithm) with the same best partial sequence. In Fig. 1(a), the function $F_l(t)$ from the graphical algorithm is presented and in Fig. 1(b), the function $F_l(t)$ from Algorithm 1 is displayed.

Next, we describe the graphical algorithm. The core is Step 3, where we describe how the states at stage $l, l > 1$, are obtained if the states at stage $l-1$ are known.

Graphical algorithm

Step 1. We number the jobs as follows: $p_1 \geq p_2 \geq \cdots \geq p_n$. If $p_i = p_{i+1}$, then $d_i \geq d_{i+1}$;

Step 2. We set $l := 1$, $\pi_1^1(t) := (1)$, $F_1(t) := \max\{0, p_1 + t - d_1\}$ for all t. We represent function $F_1(t)$ in a tabular form as given in Table 2. For both intervals, we have the same best partial sequence (1). For $t \in (-\infty, d_1 - p_1]$, there is no tardy job in the schedule corresponding to sequence (1) when this job is started at time t and for $t \in (d_1 - p_1, +\infty)$, there is one tardy job.

Step 3. Let $l > 1$ and assume that function $F_{l-1}(t)$ and the best partial sequences of the jobs $\{1, 2, \ldots, l-1\}$ for all resulting intervals given in Table 3 are known.
In the following, we describe how function $F_l(t)$ is obtained by means of function $F_{l-1}(t)$. We note that we can store the temporary functions $\Phi^1(t)$ and $\Phi^2(t)$ determined in Steps 3.1 and 3.2 also in a tabular form as in Table 1.

Step 3.1. The function $\Phi^1(t)$ is obtained from function $F_{l-1}(t)$ by the following operations. We shift the diagram of function $F_{l-1}(t)$ to the left by the value p_l and in the table for function $F_{l-1}(t)$, we add a column which results from the new break point $t' = d_l - p_l$. If $t_{l-1}^s - p_l < t' < t_{l-1}^{s+1} - p_l$, $s+1 \leq w_{l-1}$, then we have two new intervals of t in the table for

$\Phi^1(t)$: $(t_{l-1}^s - p_l, t']$ and $(t', t_{l-1}^{s+1} - p_l]$ (for $s = w_{l-1}$, we have $(t_{l-1}^{w_{l-1}} - p_l, t']$ and (t', ∞)). This means that we first replace each interval $(t_{l-1}^k, t_{l-1}^{k+1}]$ by $(t_{l-1}^k - p_l, t_{l-1}^{k+1} - p_l]$ in the table for $\Phi^1(t)$, and then we replace the column with the interval $(t_{l-1}^s - p_l, t_{l-1}^{s+1} - p_l]$ by two new columns with the intervals $(t_{l-1}^s - p_l, t']$ and $(t', t_{l-1}^{s+1} - p_l]$.

Moreover, we increase the values $u_{l-1}^{s+1}, u_{l-1}^{s+2}, \ldots, u_{l-1}^{w_{l-1}+1}$ by 1, i.e., the number of tardy jobs (and thus the slope of the function) increases. The corresponding partial sequences π^1 are obtained by adding job l as the first job to each previous partial sequence. In this way, we obtain the information on function $\Phi^1(t)$ together with the corresponding partial sequences given in Table 4, which also includes the calculation of the corresponding b values.

Step 3.2. The function $\Phi^2(t)$ is obtained from function $F_{l-1}(t)$ by the following operations. In the table for $F_{l-1}(t)$, we add a column which results from the new break point $t'' = d_l - \sum_{i=1}^{l} p_i$. If $t_{l-1}^h < t'' < t_{l-1}^{h+1},\ h+1 \le w_{l-1}$, then we have two new intervals of t in the table for $\Phi^2(t)$: $(t_{l-1}^h, t'']$ and $(t'', t_{l-1}^{h+1}]$ (for $h = w_{l-1}$, we have $(t_{l-1}^{w_{l-1}}, t'']$ and (t'', ∞)). This means that we replace the column with the interval $(t_{l-1}^h, t_{l-1}^{h+1}]$ by two new columns with the intervals $(t_{l-1}^h, t'']$ and $(t'', t_{l-1}^{h+1}]$.

Moreover, we increase the values $u_{l-1}^{h+1}, u_{l-1}^{h+2}, \ldots, u_{l-1}^{w_{l-1}+1}$ by 1, i.e., the number of tardy jobs increases. The corresponding partial sequences π^2 are obtained by adding job l at the end to each previous partial sequence. In this way, we obtain the information on function $\Phi^2(t)$ together with the corresponding partial sequences given in Table 5.

Step 3.3. Now we construct a table that corresponds to the function

$$F_l(t) = \max\{\Phi^1(t), \Phi^2(t)\}.$$

We consider all resulting intervals from both tables and search for intersection points of the diagrams of functions $\Phi^1(t)$ and $\Phi^2(t)$. Then we construct function $F_l(t)$ as the maximum of both functions obtained.

To be more precise, we construct a list $t_1, t_2, \ldots, t_e,\ t_1 < t_2 < \cdots, t_e$, of all break points t from the tables for $\Phi^1(t)$ and $\Phi^2(t)$, which are left / right boundary points of the intervals given in these tables. Then we consider each interval $(t_i, t_{i+1}],\ i = 1, 2, \ldots, e-1$, and compare the two functions $\Phi^1(t)$ and $\Phi^2(t)$ over this interval. Let the interval $(t_i, t_{i+1}]$ be contained in the interval $(t_{x-1}, t_x]$ from the table for $\Phi^1(t)$ and in the interval $(t_{y-1}, t_y]$ from the table for $\Phi^2(t)$. Then $\Phi^1(t) = (t - t_{x-1}) \cdot u_x + b_x$ and $\Phi^2(t) = (t - t_{y-1}) \cdot u_y + b_y$. We choose the column from both tables corresponding to the maximum of the two functions in the interval $(t_i, t_{i+1}]$ and insert this column into the table for $F_l(t)$. If there exists an intersection point t''' of $\Phi^1(t)$ and $\Phi^2(t)$ in this interval, then we insert two columns with the intervals $(t_i, t''']$ and $(t''', t_{i+1}]$.

This step requires $O(w_{l-1})$ operations. If we obtain two subsequent intervals with the same number of tardy jobs in the table for function $F_l(t)$ (see Table 6), we delete the column corresponding to interval $(t_l^k, t_l^{k+1}]$ and combine both intervals, i.e., we set $t_l^k := t_l^{k+1}$ (the values b_l^k, u_l^k and the sequence π_l^k in the column corresponding to the interval $(t_l^{k-1}, t_l^k]$ remain the same).

In Theorem 2, we prove that for each stage l the number of intervals is less than or equal to $l + 1 \le n + 1$.

Step 3.4. If $l = n$, then GOTO 4 else $l := l + 1$ and GOTO 3.

Step 4. In the table corresponding to function $F_n(t)$ we determine the column $(t_n^k, t_n^{k+1}]$, where $t_n^k < 0 \le t_n^{k+1}$. Then we have an optimal sequence $\pi^* = \pi_n^{k+1}$ and the optimal function value $F(\pi^*) = b_n^{k+1} + (0 - t_n^k) \cdot u_n^{k+1}$.

Table 4 Function $\Phi^1(t)$

Interval k	$(-\infty, t^1_{l-1} - p_l]$	$(t^1_{l-1} - p_l, t^2_{l-1} - p_l]$	$\ldots$	$(t^s_{l-1} - p_l, t']$	$(t', t^{s+1}_{l-1} - p_l]$	$(t^{s+1}_{l-1} - p_l, t^{s+2}_{l-1} - p_l]$	$\ldots$	$(t^{w_{l-1}}_{l-1} - p_l, +\infty)$
b_k	0	b^2_{l-1}	$\ldots$	b^{s+1}_{l-1}	b'	$\overline{b}^{s+2}_{l-1}$	$\ldots$	$\overline{b}^{w_{l-1}+1}_{l-1}$
u_k	0	u^2_{l-1}	$\ldots$	u^{s+1}_{l-1}	$u^{s+1}_{l-1} + 1$	$u^{s+2}_{l-1} + 1$	$\ldots$	$u^{w_{l-1}+1}_{l-1} + 1$
Best partial sequence π^1	(l, π^1_{l-1})	(l, π^2_{l-1})	$\ldots$	(l, π^{s+1}_{l-1})	(l, π^{s+1}_{l-1})	(l, π^{s+2}_{l-1})	$\ldots$	$(l, \pi^{w_{l-1}+1}_{l-1})$

$b' = b^{s+1}_{l-1} + (t' - (t^s_{l-1} - p_l)) \cdot u^{s+1}_{l-1}, \overline{b}^{s+2}_{l-1} = b' + ((t^{s+1}_{l-1} - p_l) - t') \cdot (u^{s+1}_{l-1} + 1), \ldots, \overline{b}^{w_{l-1}+1}_{l-1} = \overline{b}^{w_{l-1}}_{l-1} + (t^{w_{l-1}}_{l-1} - t^{w_{l-1}-1}_{l-1}) \cdot (u^{w_{l-1}}_{l-1} + 1)$

Table 5 Function $\Phi^2(t)$

Interval k	$(-\infty, t^1_{l-1}]$	$(t^1_{l-1}, t^2_{l-1}]$	$\ldots$	$(t^h_{l-1}, t'']$	$(t'', t^{h+1}_{l-1}]$	$(t^{h+1}_{l-1}, t^{h+2}_{l-1}]$	$\ldots$	$(t^{w_{l-1}}_{l-1}, +\infty)$
b_k	0	b^2_{l-1}	$\ldots$	b^{h+1}_{l-1}	b''	$\overline{b}^{h+2}_{l-1}$	$\ldots$	$\overline{b}^{w_{l-1}+1}_{l-1}$
u_k	0	u^2_{l-1}	$\ldots$	u^{h+1}_{l-1}	$u^{h+1}_{l-1} + 1$	$u^{h+2}_{l-1} + 1$	$\ldots$	$u^{w_{l-1}+1}_{l-1} + 1$
Best partial sequence π^2	(π^1_{l-1}, l)	(π^2_{l-1}, l)	$\ldots$	(π^{h+1}_{l-1}, l)	(π^{h+1}_{l-1}, l)	(π^{h+2}_{l-1}, l)	$\ldots$	$(\pi^{w_{l-1}+1}_{l-1}, l)$

$b'' = b^{h+1}_{l-1} + (t'' - t^h_{l-1}) \cdot u^{h+1}_{l-1}, \overline{b}^{h+2}_{l-1} = b'' + (t^{h+1}_{l-1} - t'') \cdot (u^{h+1}_{l-1} + 1), \ldots, \overline{b}^{w_{l-1}+1}_{l-1} = \overline{b}^{w_{l-1}}_{l-1} + (t^{w_{l-1}}_{l-1} - t^{w_{l-1}-1}_{l-1}) \cdot (u^{w_{l-1}}_{l-1} + 1)$

Table 6 Deletion of a column

Interval k	...	$(t_l^{k-1}, t_l^k]$	$(t_l^k, t_l^{k+1}]$	...
b_l^k	...	...	...	...
u_l^k	...	u_l^k	$u_l^{k+1} = u_l^k$	...
π_l^k	...	...	...	...

Table 7 Function $F_l(t)$

k	1	2	... s	$s+1$	... w_l+1
Interval k	$(-\infty, t_l^1]$	$(t_l^1, t_l^2]$	... $(t_l^{s-1}, t_l^s]$	$(t_l^s, t_l^{s+1}]$	... $(t_l^{w_l}, +\infty)$
b_l^k	0	b_l^2	... b_l^s	b_l^{s+1}	... $b_l^{w_l+1}$
u_l^k	0	u_l^2	... u_l^s	u_l^{s+1}	... $u_l^{w_l+1}$
π_l^k	π_l^1	π_l^2	... π_l^s	π_l^{s+1}	... $\pi_l^{w_l+1}$

Theorem 2 *The graphical algorithm constructs an optimal job sequence for the problem* $1(\textit{no-idle})||\max\sum T_j$ *in* $O(n^2)$ *time.*

Proof First, we prove that all functions $F_l(t),\ l = 1, 2, \ldots, n$, defined at the beginning of Sect. 3 are continuous, piecewise linear and convex functions.

It is obvious that function $F_1(t)$ is a continuous, piecewise linear and convex function with one break point. According to the operations described in Step 3, both functions $\Phi^1(t)$ and $\Phi^2(t)$ are also continuous, piecewise linear and convex functions. Thus, function

$$F_2(t) = \max\{\Phi^1(t), \Phi^2(t)\}$$

is a continuous, piecewise linear and convex function as well. Continuing in this way, we obtain that all functions $F_l(t), l = 1, 2, \ldots, n$, have the above properties.

Now assume that we have obtained Table 7 for some function $F_l(t)$ in Step 3.

We prove that $u_l^1 < u_l^2 < \cdots < u_l^s < u_l^{s+1} < \cdots < u_l^{w_l+1}$ holds. Assume that we have $u_l^s > u_l^{s+1}$. Let $\overline{F}(\pi, t)$ be the total tardiness value of the sequence π when the first job starts at time t. For each $t \in (t_l^s, t_l^{s+1}]$, we have $\overline{F}(\pi_l^s, t) > \overline{F}(\pi_l^{s+1}, t)$, since for the number of tardy jobs we have $u_l^s > u_l^{s+1}$ which gives a contradiction[1]. We recall that function $F_l(t)$ is a continuous, piecewise linear function and that $b_l^{s+1} = b_l^s + (t_l^s - t_l^{s-1}) \cdot u_l^s$. If $u_l^s = u_l^{s+1}$, then we delete the column with u_l^{s+1} and combine both intervals.

For the numbers of tardy jobs, we have $u_l^1 < u_l^2 < \cdots < u_l^s < u_l^{s+1} < \cdots < u_l^{w_l+1}$, where $w_l + 1 \le l + 1 \le n + 1$. Thus, we have at most l break points (and at most $l + 1 \le n + 1$ intervals) in the table for each function $F_l(t),\ l = 1, 2, \ldots, n$. Therefore, Step 3 requires $O(n)$ operations for each $l = 1, 2, \ldots, n$.

[1]Note that we can prove that $u_l^1 < \cdots < u_l^{w_l+1}$ holds also in another way. Functions $\Phi^1(t)$ and $\Phi^2(t)$ are convex and thus, function

$$F_l(t) = \max\{\Phi^1(t), \Phi^2(t)\}$$

is convex as well. The value u_l^s corresponds to $\tan\alpha$, where α is the angle between the t-axis and the linear segment of $F_l(t)$ (see Fig. 1).

According to the graphical algorithm, it is obvious that at each integer point t, the value $F_l(t)$ is equal to that determined by Algorithm 1 (since in the graphical algorithm, the functions $F_l(t)$ represent the same equations as in Algorithm 1 for the integer values t considered). Therefore, the graphical algorithm returns the optimal value $F_n(0)$ and constructs an optimal sequence in $O(n^2)$ time. □

In Lazarev and Gafarov (2006b), the authors use the same idea to improve a pseudo-polynomial algorithm for the NP-hard special case B-1 (Lazarev and Gafarov 2006a) of the problem $1||\min\sum T_j$. However, in this case the graphical modification of this algorithm remains pseudo-polynomial since in that paper, function

$$F_l(t) = \min\{\Phi^1(t), \Phi^2(t)\}$$

is used which is not convex.

4 A numerical example

For illustration, we consider an instance of the problem $1(\textit{no-idle})||\max\sum T_j$ with the data given in Table 8.

Now we consider subsequently all stages $l = 1, \ldots, 4$ of the graphical algorithm and store all necessary information on the states used in the graphical algorithm in a tabular form.

Consider the stage $l = 1$, i.e., we consider job 1. We obtain the only break point $d_1 - p_1 = 2$, since this is the starting time after which job 1 becomes tardy. We get two columns depending on whether job 1 is on-time or tardy and store the information on Function $F_1(t)$ given in Table 9 (it corresponds to Table 2 presented in Step 2 in Sect. 3). The diagram of function $F_1(t)$ is displayed in Fig. 2(a).

Now we consider the stage $l = 2$, i.e., we consider job 2 and obtain the following results. To determine function $\Phi^1(t)$ for $l = 2$, we sequence job 2 as the first job. According to Step 3.1 of the graphical algorithm, we shift all intervals given in Table 9 for function $F_1(t)$ to the left by $p_2 = 22$, i.e., in the resulting table we have two columns with the intervals $(-\infty, -20]$ and $(-20, +\infty)$. Then we compute the new break point $t' = d_2 - p_2 = 13$. For each sequence of the first l jobs, where the first job $l = 2$ starts at time t, the job $l = 2$ is on-time for each $t \le t'$, and the job $l = 2$ is tardy for each $t > t'$. This means that at this point t', the number of tardy jobs increases (i.e., the slope of the piecewise-linear function $\Phi^1(t)$ changes). Then we split the column corresponding to the interval $(-20, +\infty)$ into two

Table 8 Data of the instance

j	1	2	3	4
p_j	30	22	12	5
d_j	32	35	38	40

Table 9 Function $F_1(t)$

Interval k	$(-\infty, 2]$	$(2, +\infty)$
b_1^k	0	0
u_1^k	0	1
π_1^k	(1)	(1)

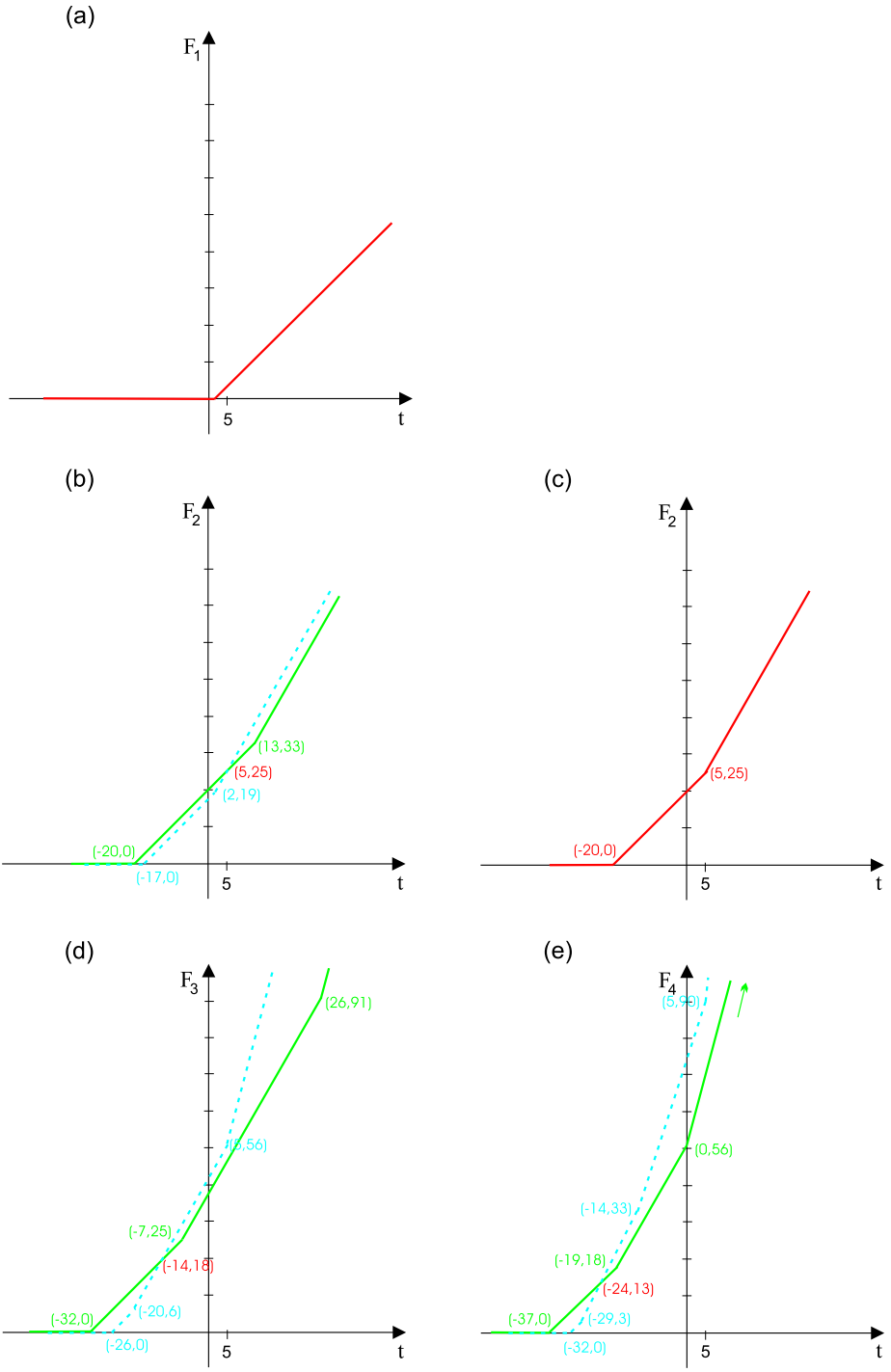

Fig. 2 Example

Table 10 Function $\Phi^1(t)$ for $l=2$

Interval k	$(-\infty,-20]$	$(-20,13]$	$(13,+\infty)$
b_k	0	0	33
u_k	0	1	2
π^1	(2, 1)	(2, 1)	(2, 1)

Table 11 Function $\Phi^2(t)$ for $l=2$

Interval k	$(-\infty,-17]$	$(-17,2]$	$(2,+\infty)$
b_k	0	0	19
u_k	0	1	2
π^2	(1,2)	(1,2)	(1,2)

Table 12 Function $F_2(t)$

Interval k	$(-\infty,-20]$	$(-20,5]$	$(5,+\infty)$
b_2^k	0	0	25
u_2^k	0	1	2
π_2^k	(2, 1)	(2, 1)	(1, 2)

columns representing the intervals $(-20, 13]$ and $(13, +\infty)$ since $t' = 13 \in (-20, +\infty)$. Thus, we get the information given on function $\Phi^1(t)$ together with the corresponding best partial sequences in Table 10 which has three columns (i.e., the number of tardy jobs can be 0, 1 or 2).

To determine function $\Phi^2(t)$ for $l = 2$, we add job 2 to the end of a partial sequence. According to Step 3.2 of the graphical algorithm, we compute the new break point $t'' = d_2 - (p_1 + p_2) = -17$. For any sequence of the first l jobs, where the first job starts at time t and job $l = 2$ is the last job, job 2 is tardy for $t > t''$ with $t'' = -17 \in (-\infty, 2]$. Therefore, we split the column corresponding to the interval $(-\infty, 2]$ into two columns with the intervals $(-\infty, -17]$ and $(-17, 2]$. We obtain the information on function $\Phi^2(t)$ for $l = 2$ together with the corresponding partial sequences given in Table 11.

Now according to Step 3.3, we combine the tables for functions $\Phi^1(t)$ and $\Phi^2(t)$. We consider each of the intervals $(-\infty, -20]$, $(-20, -17]$, $(-17, 2]$, $(2, 13]$ and $(13, +\infty)$, which result from the break points given in the two tables and check whether there exist intersection points of functions $\Phi^1(t)$ and $\Phi^2(t)$ in these intervals. We remind that functions $\Phi^1(t)$ and $\Phi^2(t)$ are defined as functions of the type $u_k \cdot (t - t_{k-1}) + b_k$.

For $t \in (2, 13]$, we obtain from $\Phi^1(t) = (t + 20) \cdot 1 + 0 = \Phi^2(t) = (t - 2) \cdot 2 + 19$ the intersection point $(5, 25)$ (for the other intervals, intersection points do not exist). For $t \leq 5$, we have $\Phi^1(t) = \max\{\Phi^1(t), \Phi^2(t)\}$, and for $t > 5$, we have $\Phi^2(t) = \max\{\Phi^1(t), \Phi^2(t)\}$, i.e., for $t \leq 5$, the partial sequence (2,1) corresponding to $\Phi^1(t)$ is better, while for $t > 5$, the partial sequence $(1, 2)$ corresponding to $\Phi^2(t)$ is better. Therefore, to construct the table for function $F_2(t)$, we get the columns with the intervals $(-\infty, -20]$ and $(-20, 5]$ from the table for $\Phi^1(t)$ and the column $(5, +\infty) \subset (2, +\infty)$ from the table for $\Phi^2(t)$.

Thus, combining the results from the two tables for $\Phi^1(t)$ and $\Phi^2(t)$, we obtain the information on function $F_2(t)$ together with the best partial sequences for the corresponding intervals given in Table 12. The diagram of function $F_2(t)$ is displayed in Fig. 2(c) obtained from Fig. 2(b).

Now we consider the stage $l = 3$, i.e., we sequence job 3 and obtain the following results. To determine function $\Phi^1(t)$ for $l = 3$, we sequence job 3 as the first one and shift all

Table 13 Function $\Phi^1(t)$ for $l = 3$

Interval k	$(-\infty, -32]$	$(-32, -7]$	$(-7, 26]$	$(26, +\infty)$
b_k	0	0	25	91
u_k	0	1	2	3
π^1	(3,2,1)	(3,2,1)	(3,1,2)	(3,1,2)

Table 14 Function $\Phi^2(t)$ for $l = 3$

Interval k	$(-\infty, -26]$	$(-26, -20]$	$(-20, 5]$	$(5, +\infty)$
b_k	0	0	6	56
u_k	0	1	2	3
π^2	(2,1,3)	(2,1,3)	(2,1,3)	(1,2,3)

Table 15 Function $F_3(t)$

Interval k	$(-\infty, -32]$	$(-32, -14]$	$(-14, 5]$	$(5, +\infty)$
b_3^k	0	0	18	56
u_3^k	0	1	2	3
π_3^k	(3,2,1)	(3,2,1)	(2,1,3)	(1,2,3)

Table 16 Function $\Phi^1(t)$ for $l = 4$

Interval k	$(-\infty, -37]$	$(-37, -19]$	$(-19, 0]$	$(0, 35]$	$(35, +\infty)$
b_k	0	0	18	56	161
u_k	0	1	2	3	4
π^1	(4,3,2,1)	(4,3,2,1)	(4,2,1,3)	(4,1,2,3)	(4,1,2,3)

intervals given in Table 12 to the left by $p_3 = 12$. We get the new break point $t' = d_3 - p_3 = 26$ and obtain the information on function $\Phi^1(t)$ together with the corresponding partial sequences for $l = 3$ given in Table 13.

To determine function $\Phi^2(t)$ for $l = 3$, we sequence job 3 as the last one and get the new break point $t'' = d_3 - (p_1 + p_2 + p_3) = -26$. We obtain the information on function $\Phi^2(t)$ for $l = 3$ together with the corresponding partial sequences given in Table 14.

The only intersection point of the diagrams of functions $\Phi^1(t)$ and $\Phi^2(t)$ is $(-14, 18)$. Thus, we obtain the information on function $F_3(t)$ and the best partial sequences given in Table 15. The diagram of function $F_3(t)$ is displayed in Fig. 2(d).

Finally, we consider the stage $l = 4$, i.e., we sequence job 4 and obtain the following results. To determine function $\Phi^1(t)$ for $l = 4$, we sequence job 4 as the first one and get the new break point $t' = d_4 - p_4 = 35$. We obtain the information on function $\Phi^1(t)$ for $l = 4$ together with the corresponding partial sequences given in Table 16.

To determine function $\Phi^2(t)$ for $l = 4$, we sequence job 4 as the last one and get the new break point $t'' = d_4 - (p_1 + p_2 + p_3 + p_4) = -29$. We obtain the information on function $\Phi^2(t)$ together with the corresponding partial sequences for $l = 4$ given in Table 17.

The only intersection point of the diagrams of functions $\Phi^1(t)$ and $\Phi^2(t)$ is $(-24, 13)$. Thus, we obtain the information on function $F_4(t)$ and the corresponding best partial sequences given in Table 18. The diagram of function $F_4(t)$ is displayed in Fig. 2(e).

The optimal objective function value is $F_4(0) = 33 + 14 \cdot 3 = 75$, and the resulting optimal job sequence is $\pi^* = (2, 1, 3, 4)$.

Table 17 Function $\Phi^2(t)$ for $l=4$

Interval k	$(-\infty,-32]$	$(-32,-29]$	$(-29,-14]$	$(-14,5]$	$(5,+\infty)$
b_k	0	0	3	33	90
u_k	0	1	2	3	4
π^2	(3,2,1,4)	(3,2,1,4)	(3,2,1,4)	(2,1,3,4)	(1,2,3,4)

Table 18 Function $F_4(t)$

Interval k	$(-\infty,-37]$	$(-37,-24]$	$(-24,-14]$	$(-14,5]$	$(5,+\infty)$
b_4^k	0	0	13	33	90
u_4^k	0	1	2	3	4
π_4^k	(4,3,2,1)	(4,3,2,1)	(3,2,1,4)	(2,1,3,4)	(1,2,3,4)

5 Concluding remarks

In this paper, we used a graphical approach to improve a known pseudo-polynomial algorithm for the problem $1(\textit{no-idle})||\max\sum T_j$ in such a way that the resulting algorithm has a running time of $O(n^2)$. This polynomial algorithm also settled the complexity status of this problem which was open up to now. The polynomial solvability of this problem is in contrast to the minimization version of this problem which is known to be NP-hard.

The idea of the algorithm presented in this paper can also be used to solve the NP-hard problem $1(\textit{no-idle})||\max\sum w_jT_j$ considered in Lawler and Moore (1969), Gafarov et al. (2010b). The graphical modification of the pseudo-polynomial dynamic programming algorithm has a time complexity of $O(\min\{2^n, n\cdot\min\{F, d_{\max}\}\})$ instead of $O(nd_{\max})$ for the algorithm from Lawler and Moore (1969), where $F\le\sum w_j$ denotes the total weight of the tardy jobs in an optimal schedule.

The graphical approach can be applied to problems, where a pseudo-polynomial algorithm exists and Boolean variables are used in the sense that yes/no decisions have to be made. However, for the knapsack problem, the graphical algorithm mostly reduces substantially the number of states to be considered but the time complexity of the algorithm remains pseudo-polynomial. For the single machine problem of maximizing total tardiness, the graphical algorithm improved the complexity from $O(n\sum p_j)$ to $O(n^2)$. Thus, the graphical approach is not only of a practical but also of a theoretical importance.

Acknowledgements This work was partially supported by DAAD (Deutscher Akademischer Austauschdienst): A/08/80442/Ref. 325. The authors are grateful to the referees for their constructive comments, which substantially improved the presentation of the results.

References

Aloulou, M. A., & Artigues, C. (2010). Flexible solutions in disjunctive scheduling: general formulation and study of the flow-shop case. *Computers & Operations Research*, *37*(5), 890–898.

Aloulou, M. A., Kovalyov, M. Y., & Portmann, M.-C. (2004). Maximization problems in single machine scheduling. *Annals of Operations Research*, *129*, 21–32.

Aloulou, M. A., Kovalyov, M. Y., & Portmann, M.-C. (2007). Evaluation flexible solutions in single machine scheduling via objective function maximization: the study of computational complexity. *RAIRO. Recherche Opérationnelle*, *41*, 1–18.

Babu, P., Peridy, L., & Pinson, E. (2004). A branch and bound algorithm to minimize total weighted tardiness on a single processor. *Annals of Operations Research*, *129*, 33–46.

Du, J., & Leung, J. Y.-T. (1990). Minimizing total tardiness on one processor is NP-hard. *Mathematics of Operations Research*, *15*, 483–495.

Gafarov, E. R., Lazarev, A. A., & Werner, F. (2010a). Algorithms for maximizing the number of tardy jobs or total tardiness on a single machine. *Automation and Remote Control*, *71*(10), 2070–2084.

Gafarov, E. R., Lazarev, A. A., & Werner, F. (2010b). *A modification of dynamic programming algorithms to reduce the running time or/and complexity*. Preprint 20/10, FMA, Otto-von-Guericke-Universität Magdeburg.

Gafarov, E. R., Lazarev, A. A., & Werner, F. (2010c). *Classical combinatorial and single machine scheduling problems with opposite optimality criteria*. Preprint 11/10, FMA, Otto-von-Guericke-Universität Magdeburg.

Lawler, E. L. (1977). A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, *1*, 331–342.

Lawler, E. L., & Moore, J. M. (1969). A functional equation and its application to resource allocation and sequencing problems. *Management Science*, *16*(1), 77–84.

Lazarev, A. A., & Gafarov, E. R. (2006a). Special case of the single-machine total tardiness problem is NP-hard. *Journal of Computer and Systems Sciences International*, *45*(3), 450–458.

Lazarev, A. A., & Gafarov, E. R. (2006b). *Scheduling theory. Total tardiness problem*. Moscow: Computing Center of the Russian Academy of Sciences, 128 pp. (in Russian).

Lazarev, A. A., & Werner, F. (2009a). A graphical realization of the dynamic programming method for solving NP-hard combinatorial problems. *Computers and Mathematics with Applications*, *58*, 619–631.

Lazarev, A. A., & Werner, F. (2009b). Algorithms for special cases of the single machine total tardiness problem and an application to the even-odd partition problem. *Mathematical and Computer Modelling*, *49*(9–10), 2061–2072.

Matsuo, H., Suh, C. J., & Sullivan, R. S. (1989). A controlled search simulated annealing method for the single machine weighted tardiness problem. *Annals of Operations Research*, *21*, 85–108.

Posner, M. E. (1990). Reducibility among weighted completion time scheduling problems. *Annals of Operations Research*, *26*, 91–101.

Potts, C. N., & Van Wassenhove, L. N. (1982). A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters*, *1*, 363–377.

Szwarc, W., Della Croce, F., & Grosso, A. (1999). Solution of the single machine total tardiness problem. *Journal of Scheduling*, *2*, 55–71.