A Polynomial Time Graphical Algorithm for Maximizing Total Tardiness on a Single Machine

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

Institute of Control Sciences of the Russian Academy of Sciences,
Profsoyuznaya st. 65, 117997 Moscow, Russia,
email: [a] *axel73@mail.ru,* [b] *jobmath@mail.ru*

Frank Werner

Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg,
PSF 4120, 39016 Magdeburg, Germany,
email: frank.werner@mathematik.uni-magdeburg.de

April 20, 2010

Abstract

In this paper, 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.

Keywords: Scheduling, Single machine problems, Maximization problems, Total tardiness

MSC classification: 90 B 35

1 Introduction

Typically, in scheduling theory problems are considered, where a specific objective function has to be minimized. For instance, the minimization of makespan is a very popular optimization criterion. When a sum function is considered, often total completion time, total tardiness or the number of tardy jobs have to be minimized. In this paper, we consider a single machine problem with an *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. Preemptions of a job are not allowed. The machine can handle only one job at a time. All the 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 *idle times are not allowed* (note that otherwise the maximization problem considered in this paper would be trivial). Thus, a schedule $\pi = (j_1, j_2, \ldots, j_n)$ is uniquely determined by a permutation of the jobs of set N. Each feasible schedule starts at time 0 and does not have any idle time between the processing of jobs. 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. 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 schedule π. The objective is to find an optimal schedule π^* that maximizes total tardiness, i.e., $F(\pi) = \sum_{j=1}^{n} T_j(\pi)$. We will denote this problem by $1||\max \sum T_j$.

In this paper, notation $\{\pi\}$ denotes the set of jobs contained in sequence π. Notation $i \in \pi$ means $i \in \{\pi\}$. For $\pi = (\pi_1, i, \pi_2)$, notation $\pi \setminus \{i\}$ means (π_1, π_2). Notation $j \rightarrow i$ expresses that the processing of job j precedes the processing of job i, and notation $(j \rightarrow i)_\pi$ means that this holds in schedule π.

On one side, investigations of problems with *opposite* optimization criteria itself are an important theoretical task. For instance, they can be used for proposing algorithms and deriving properties of an optimal schedule of this opposite problem which might be taken to compute upper bounds for the original problems. Algorithms for such problems with opposite optimization criterion can be used to compute parts of optimal schedules for the original problems, or to cut bad subproblems in the branching tree of branch-and-bound algorithms. For example, the algorithm for problem $1||\max \sum U_j$ from [7] can be used to compute the maximal number of tardy jobs, and later we can use this number to reduce the search for an optimal solution of problem $1||\max \sum T_j$. Moreover, we can use the value $\max \sum T_j$ for the computation of lower and upper bounds for the optimal objective function value of problem $1||(\alpha \sum E_j + \beta \sum T_j)$, i.e., if $\alpha > \beta$, then we can

'fix' the maximal value $\beta \sum T_j$ and search for an optimal schedule for criterion $\sum E_j$. In addition, it is interesting from a theoretical point whether polynomially solvable cases for the minimization problem are also easy for the maximization case, or vice versa.

On the other side, such problems separately have practical interpretations and applications. For example, an installation team has to mount wind turbines in various regions of a 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 has a particular processing time) 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 rapidly after the snowmelt. For each region of the country (i.e., for each corresponding job), there exists a forecast in which week snowmelt can be expected (interpreted as a due date). The objective function is the minimization of these additional costs. In fact, this objective function 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., problem $1||\max \sum T_j$ 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 the problems $1||\max \sum T_j$ and $1||\max \sum E_j$ are identical. Assume that we have an instance of problem $1||\max \sum E_j$ and an optimal schedule $\pi = (1, 2, \ldots, n)$. We construct an instance of problem $1||\max \sum T_j$ 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 schedule $\pi' = (n, n-1, \ldots, 1)$ is optimal for the above instance of problem $1||\max \sum T_j$. 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 problem $1||\max \sum E_j$, 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 $d_j - C_j$.

Problem $1 \,||\, \min \sum T_j$ is NP-hard in the ordinary sense [1, 2]. A pseudo-polynomial dynamic programming algorithm of time complexity $O(n^4 \sum p_j)$ has been proposed by Lawler [3]. The state-of-the-art algorithms by Szwarc et al. [4] can solve special instances [5] of this problem for $n \leq 500$ jobs. A summary of polynomially and pseudo-polynomially solvable special cases can be found e.g. in [6].

In [7, 8], the authors investigated the complexity of single machine scheduling problems with classical objective functions and the opposite

optimization criteria. Moreover, a practical interpretation, a proof of NP-hardness and a pseudo-polynomial time algorithm for the knapsack minimization problem have been presented. A pseudo-polynomial time algorithm for problem $1||\max\sum T_j$ has been presented in [7].

For a practical realization of some pseudo-polynomial algorithms, one can use the idea from [9]. The modification of pseudo-polynomial algorithms for the combinatorial problems with Boolean variables (i.e., the variables represent a yes/no answer, e.g. a job is on-time or tardy, or an item is put into the knapsack or not) is called *graphical approach*. As computational experiments for the partition problem show, for a substantial part of instances, the time complexity time of such a modification is polynomially bounded.

In this paper, we propose such a graphical modification of a pseudo-polynomial algorithm for problem $1||\max\sum T_j$, and we prove that the suggested graphical algorithm has a polynomial time complexity. This is in contrast to the problem of minimizing total tardiness.

The rest of this paper is organized as follows. In Section 2, the exact pseudo-polynomial algorithm for problem $1||\max\sum T_j$ is presented. A graphical modification of this pseudo-polynomial algorithm for problem $1||\max\sum T_j$ and the proof of its polynomial running time are given in Section 3. In Section 4, a numerical example for illustrating the suggested algorithm is presented.

2 A Pseudo-Polynomial Time Algorithm for Problem $1||\max\sum T_j$

In this subsection, we present a property of an optimal schedule and an exact algorithm for the problem under consideration.

Lemma 1 *There exists an optimal schedule $\pi = (G, H) = (SPT, LPT)$, 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.*

Proof.
1) Assume that there exists an optimal schedule $\pi = (\pi_1, j, \pi_2, i, \pi_3)$, where job j is tardy and job i is on-time. For schedule $\pi' = (\pi_1, i, j, \pi_2, \pi_3)$, we have: $F(\pi') - F(\pi) \geq (T_j(\pi') - T_j(\pi)) + (T_i(\pi') - T_i(\pi)) = p_i + 0 > 0$. Therefore, we have a contradiction since schedule π' has a larger value of total tardiness, i.e., π' is better and π is not optimal.

2) We consider an optimal schedule $\pi = (G, H)$, where all jobs $j \in H$ are tardy and all jobs $i \in G$ are on-time. Now we prove that

all jobs from H are processed according to an LPT order. Assume that there exists an optimal schedule $\pi = (\pi_1, j_1, j_2, \pi_2)$, where j_1 and j_2 are tardy and $p_{j_1} < p_{j_2}$. For schedule $\pi' = (\pi_1, j_2, j_1, \pi_2)$, we have $F(\pi') - F(\pi) = (T_{j_1}(\pi') - T_{j_1}(\pi)) + (T_{j_2}(\pi') - T_{j_2}(\pi)) \geq p_{j_2} - \min\{p_{j_1}, T_{j_2}(\pi)\} > 0$. Therefore, we have a contradiction and $\pi = (\pi_1, j_1, j_2, \pi_2)$ is not optimal.

3) We consider an optimal schedule $\pi = (G, H)$, where all jobs $j \in H$ are tardy and all jobs $i \in G$ are on-time. Now, we prove that all jobs $i \in G$ can be processed in an SPT order in an optimal schedule. For all jobs $i \in G$, we have $d_i \geq \sum_{k \in G} p_k$, otherwise, if $d_i < \sum_{k \in G} p_k$, then schedule $\pi' = (G \setminus \{i\}, i, H)$ is better, and we have a contradiction. Therefore, the jobs from G can be processed in any order.

□

The following algorithm is based on Lemma 1.

Algorithm 1

1. Enumerate the jobs according to non-increasing processing times: $p_1 \geq p_2 \geq \ldots \geq p_n$. If $p_i = p_{i+1}$, then $d_i \geq d_{i+1}$;
2. $\pi_1(t) := (1)$, $F_1(t) := \max\{0, p_1 + t - d_1\}$ for all $t \in Z$ with $t \in [0, \sum_{j=2}^{n} p_j]$;
3. FOR $l := 2$ TO n DO

   FOR $t := 0$ TO $\sum_{j=l+1}^{n} p_j$ $(t \in Z)$ DO

   $\pi^1 := (l, \pi_{l-1}(t + p_l))$, $\pi^2 := (\pi_{l-1}(t), l)$;

   $F(\pi^1, t) := \max\{0, p_l + t - d_l\} + F_{l-1}(t + p_l)$;

   $$F(\pi^2, t) := F_{l-1}(t) + \max\left\{0, \sum_{j=1}^{l} p_j + t - d_l\right\};$$

   $F_l(t) := \max\{F(\pi^1, t), F(\pi^2, t)\}$;

   $\pi_l(t) := \arg\max\{F(\pi^1, t), F(\pi^2, t)\}$.
4. $\pi_n(0)$ is an optimal schedule with the objective function value $F_n(0)$.

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

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

Proof. We prove the theorem indirectly. Assume that there is an optimal schedule of the form $\pi^* = (SPT, LPT)$, 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 schedule. 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 $F((\pi_\alpha, \bar{\pi}_l, \pi_\beta)) \leq F((\pi_\alpha, \pi_l(\sum_{i \in \pi_\alpha} p_i), \pi_\beta))$. Analogously, step by step, we modify the partial schedules $\bar{\pi}_l$ corresponding to the subsequent values l. At the end, we have $F(\pi^*) \leq F(\pi') = F_n(0)$. Thus, 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 $1||\max \sum T_j$

In this section, we derive a new graphical algorithm for this problem, which is based on an idea from [9].

In each step of the graphical algorithm, we store function $F_l(t)$ as a table of the form:

Table 1: Function $F_l(t)$

t	$(-\infty, t_1]$	$(t_1, t_2]$	$\ldots$	$(t_m, +\infty)$
$F_l(t)$	$b_1 = 0$	b_2	$\ldots$	b_{m+1}
number of tardy jobs	$u_1 = 0$	u_2	$\ldots$	u_{m+1}
optimal partial schedule	π_1	π_2	$\ldots$	π_{m+1}

The above data means the following. For each value $t \in (t_k, t_{k+1}]$, we have an optimal partial schedule π_{k+1} with u_{k+1} tardy jobs and the function value $F_l(t) = b_{k+1} + (t - t_k) \cdot u_{k+1}$. In Theorem 2, we prove that this table represents a continuous, piecewise-linear and convex function $F_l(t)$. The points $t_1, t_2, \ldots, t_m$ are called the *break points* since there is a change from value u_k to u_{k+1} (which means that the slope of the piecewise-linear function changes). For describing each linear segment, we store its slope u_{k+1} and its function value b_{k+1} at point $t = t_k$. Then the graphical algorithm is as follows.

Graphical algorithm

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

Step 2. $l := 1$, $\pi_1(t) := (1)$, $F_1(t) := \max\{0, p_1 + t - d_1\}$ for all $t \in Z$. We represent function $F_1(t)$ in tabular form as given in Table

2.

Table 2: Function $F_1(t)$

t	$(-\infty, d_1 - p_1]$	$(d_1 - p_1, +\infty)$
$F_1(t)$	0	0
number of tardy jobs	0	1
optimal partial schedule	(1)	(1)

Step 3. Now, let $l > 1$, and we assume that function $F_{l-1}(t)$ is known and described by Table 3.

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

t	$(-\infty, t_1]$	$(t_1, t_2]$	$\ldots$	$(t_m, +\infty)$
$F_{l-1}(t)$	$b_1 = 0$	b_2	$\ldots$	b_{m+1}
number of tardy jobs	$u_1 = 0$	u_2	$\ldots$	u_{m+1}
optimal partial schedule	π_1	π_2	$\ldots$	π_{m+1}

In the following, we describe how function $F_l(t)$ is determined by means of function $F_{l-1}(t)$.

Step 3.1. A function $F(\pi^1, t)$ is obtained from function $F_{l-1}(t)$ by the following operations. We shift the graph 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 $t_k - p_l < t' < t_{k+1} - p_l,\ k+1 \leq m$, then in the new table for $F(\pi^1, t)$, we have two new intervals of t: $(t_k - p_l, t']$ and $(t', t_{k+1} - p_l]$. Moreover, we increase the values $u_{k+1}, u_{k+2}, \ldots, u_{m+1}$ by 1, i.e., the number of tardy jobs (and thus the slope of the corresponding function) increases. The corresponding partial sequences π^1 are obtained by adding job l as the first job to each previous partial sequence. Then we obtain Table 4 for function $F(\pi^1, t)$.

Step 3.2. Function $F(\pi^2, t)$ 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 $t' = d_l - \sum_{i=1}^{l} p_l$. If $t_k < t' < t_{k+1},\ k+1 \leq m$, then in the new table, we have two new intervals of t: $(t_k, t']$ and $(t', t_{k+1}]$. Moreover, we increase the values $u_{k+1}, u_{k+2}, \ldots, u_{m+1}$ 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 Table 5 for function $F(\pi^2, t)$.

Step 3.3. Now we construct a table that corresponds to the function $F_l(t) = \max\{F(\pi^1, t), F(\pi^2, t)\}$. We compare the intervals from

Table 4: Function $F(\pi^1, t)$

t	$(-\infty, t_1 - p_l]$	$(t_1 - p_l, t_2 - p_l]$	$\ldots$	$(t_k - p_l, t']$	$(t', t_{k+1} - p_l]$	$(t_{k+1} - p_l, t_{k+2} - p_l]$	$\ldots$	$(t_m - p_l, +\infty)$
$F(\pi^1, t)$	$b_1 = 0$	b_2	$\ldots$	b_{k+1}	b'	b'_{k+2}	$\ldots$	b'_{m+1}
number of tardy jobs	$u_1 = 0$	u_2	$\ldots$	u_{k+1}	$u_{k+1} + 1$	$u_{k+2} + 1$	$\ldots$	$u_{m+1} + 1$
partial schedule π^1	(l, π_1)	(l, π_2)	$\ldots$	(l, π_{k+1})	(l, π_{k+1})	(l, π_{k+2})	$\ldots$	(l, π_{m+1})

$b' = b_{k+1} + (t' - (t_k - p_l)) \cdot u_{k+1}$, $b'_{k+2} = b' + ((t_{k+1} - p_l) - t') \cdot (u_{k+1} + 1)$, $\ldots$, $b'_{m+1} = b_m{}' + ((t_m - t_{m-1}) \cdot (u_m + 1)$

Table 5: Function $F(\pi^2, t)$

t	$(-\infty, t_1]$	$(t_1, t_2]$	$\ldots$	$(t_k, t']$	$(t', t_{k+1}]$	$(t_{k+1}, t_{k+2}]$	$\ldots$	$(t_m, +\infty)$
$F(\pi^2, t)$	$b_1 = 0$	b_2	$\ldots$	b_{k+1}	b'	b'_{k+2}	$\ldots$	b'_{m+1}
number of tardy jobs	$u_1 = 0$	u_2	$\ldots$	u_{k+1}	$u_{k+1} + 1$	$u_{k+2} + 1$	$\ldots$	$u_{m+1} + 1$
partial schedule π^2	(π_1, l)	(π_2, l)	$\ldots$	(π_{k+1}, l)	(π_{k+1}, l)	(π_{k+2}, l)	$\ldots$	(π_{m+1}, l)

$b' = b_{k+1} + (t' - t_k) \cdot u_{k+1}$, $b'_{k+2} = b' + ((t_{k+1}) - t') \cdot (u_{k+1} + 1)$, $\ldots$, $b'_{m+1} = b_m{}' + ((t_m - t_{m-1}) \cdot (u_m + 1)$

Table 7: Function $F_l(t)$

t	$(-\infty, t_1]$	$(t_1, t_2]$	$\ldots$	$(t_{k-1}, t_k]$	$(t_k, t_{k+1}]$	$\ldots$	$(t_m, +\infty)$
$F_l(t)$	$b_1 = 0$	b_2	$\ldots$	b_k	b_{k+1}	$\ldots$	b_{m+1}
number of tardy jobs	$u_1 = 0$	u_2	$\ldots$	u_k	u_{k+1}	$\ldots$	u_{m+1}
optimal partial schedule	π_1	π_2	$\ldots$	π_k	π_{k+1}	$\ldots$	π_{m+1}

both tables and search for intersection points of the graphs of functions $F(\pi^1, t)$ and $F(\pi^2, t)$. This step requires $O(m)$ operations. If in the table for function $F_l(t)$, we have the situation displayed in Table 6, we cut the column, which corresponds to interval $(t_k, t_{k+1}]$, and combine both intervals, i.e., we set $t_k := t_{k+1}$.

Table 6: Deletion of a column

t	...	$(t_{k-1}, t_k]$	$(t_k, t_{k+1}]$	...
$F_l(t)$	...	...	...	...
number of tardy jobs	...	u_k	$u_{k+1} = u_k$	...
optimal partial schedule π_l	...	...	...	...

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

Step 3.3. If $l = n$, then GOTO 4 else $l := l + 1$ and GOTO 3.

Step 4. In the table, which corresponds to function $F_n(t)$, we determine the column $(t_k, t_{k+1}]$, where $t_k < 0 \leq t_{k+1}$. Then we have an optimal schedule $\pi^* = \pi_{k+1}$ and the optimal function value $F(\pi^*) = b_{k+1} + (0 - t_k) \cdot u_{k+1}$.

It is obvious that the graphical algorithm corresponds to Algorithm 1. In Algorithm 1, we consider all points $t \in Z$ with $t \in [0, \sum_{j=2}^{n} p_j]$ and in the graphical algorithm, we consider only the break points from this interval. Thus, the graphical algorithm is exact.

Theorem 2 *The graphical algorithm constructs an optimal schedule in $O(n^2)$ time.*

Proof.

First, we prove that all functions $F_l(t),\ l = 1, 2, \ldots, n$, are convex, 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 in Step 3, both functions $F(\pi^1, t)$ and $F(\pi^2, t)$ are also continuous, piecewise linear and convex functions. Thus, function

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

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

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

We prove that $u_1 < u_2 < \ldots < u_k < u_{k+1} < \ldots < u_{m+1}$ holds. Assume that we have $u_k > u_{k+1}$. Then, for each $t \in (t_k, t_{k+1}]$, where t is the starting time of the processing of the jobs from the partial schedule π_k or π_{k+1}, we have $F(\pi_k) > F(\pi_{k+1})$ since for the number of tardy jobs, we have $u_k > u_{k+1}$. We note that function $F_l(t)$ is a continuous, piecewise linear function and $b_{k+1} = b_k + (t_k - t_{k-1}) \cdot u_k$. Thus, we have a contradiction. If $u_k = u_{k+1}$, then we cut the column with u_{k+1} and combine both intervals.

Note that we can prove this fact also in another way. Functions $F(\pi^1, t)$ and $F(\pi^2, t)$ are convex and thus, function

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

is convex, too. The value u_k denotes $\tan\alpha$, where α is the angle between the x-axis and the linear segment of $F_l(t)$.

We have $u_1 < u_2 < \ldots < u_k < u_{k+1} < \ldots < u_{m+1}$, where $u_i,\ i = 1, 2, \ldots, m+1$, is the number of tardy jobs. Thus, we have $m+1 \leq l+1 \leq n+1$. As a consequence, we have at most l break points and thus, at most $l+1 \leq n+1$ columns for each function $F_l(t),\ l = 1, 2, \ldots, n$.

Therefore, Step 3 requires $O(n)$ operations for each $l = 1, 2, \ldots, n$, and thus, the graphical algorithm constructs an optimal schedule in $O(n^2)$ time.

□

In [10], the authors use the same idea to improve a pseudo-polynomial algorithm for the NP-hard special case B-1 [2] of problem $1||\sum T_j$. However, in this case the graphical modification of this algorithm remains pseudo-polynomial since in that paper, function

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

is used which is not convex.

4 A Numerical Example

For illustration, we consider an instance of problem $1||\max\sum T_j$ with the data given in Table 8.

Table 8: Data of the instance

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

Now we describe the data stored in each step.

Let $l = 1$**.** We store the information given in Table 9.

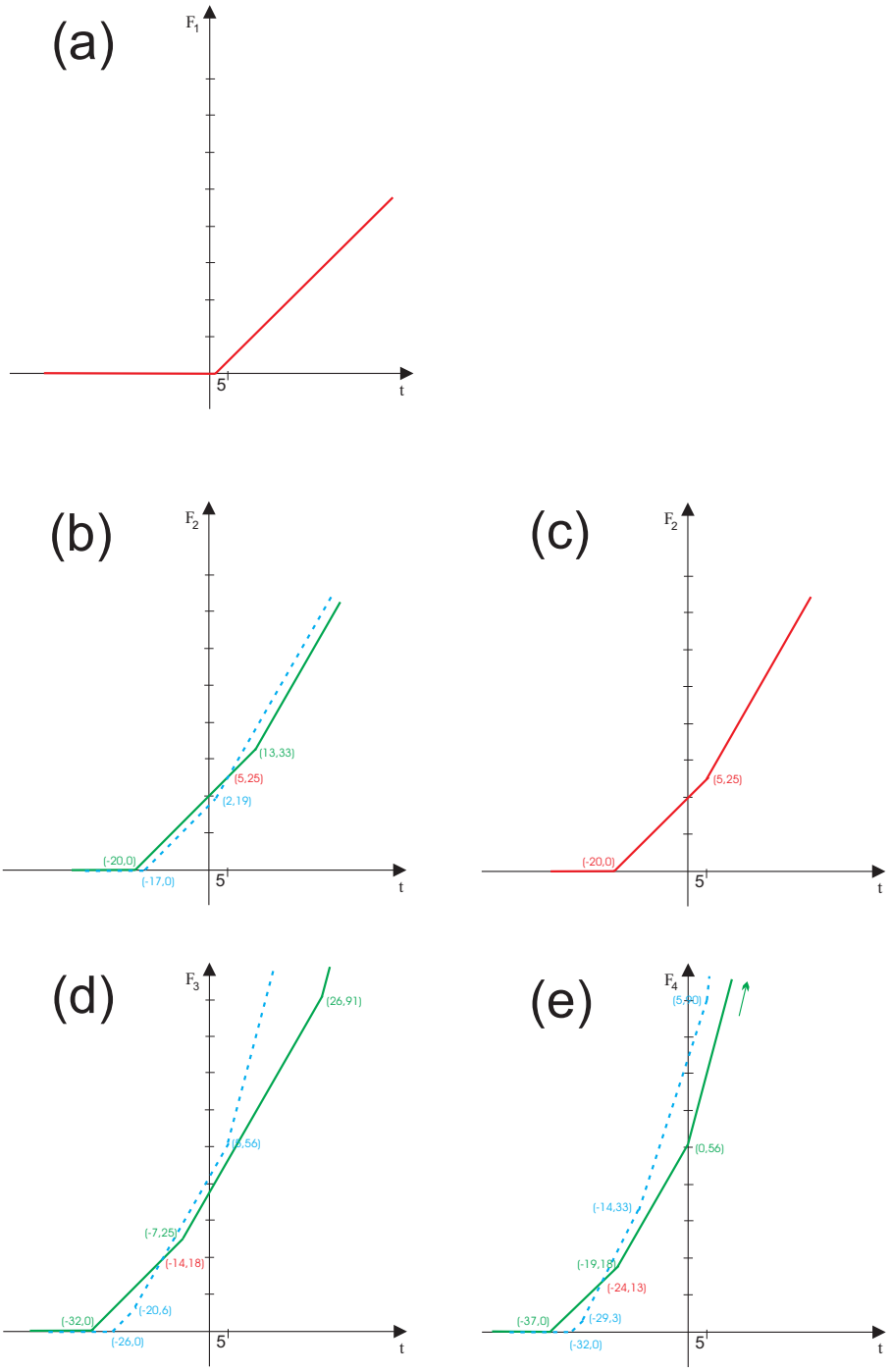

Figure 1: Example

Table 9: Function $F_1(t)$

t	$(-\infty, 2]$	$(2, +\infty)$
$F_1(t)$	0	0
u	0	1
π_1	(1)	(1)

The graph of function $F_1(t)$ is displayed in Fig. 1 (a).

Let $l = 2$**.** We obtain the following results given in Table 10.

We determine the new break point $t' = d_2 - p_2 = 13$, and we get Table 10 for function $F(\pi^1, t)$.

Table 10: Function $F(\pi^1, t)$

t	$(-\infty, -20]$	$(-20, 13]$	$(13, +\infty)$
$F(\pi^1, t)$	0	0	33
u	0	1	2
π^1	(2,1)	(2,1)	(2,1)

We determine the new break point $t' = d_2 - (p_1 + p_2) = -17$, and we obtain Table 11 for function $F(\pi^2, t)$.

Table 11: Function $F(\pi^2, t)$

t	$(-\infty, -17]$	$(-17, 2]$	$(2, +\infty)$
$F(\pi^2, t)$	0	0	19
u	0	1	2
π^2	(1,2)	(1,2)	(1,2)

Now we consider each of intervals $(-\infty, -20]$, $(-20, -17]$, $(-17, 2]$, $(2, 13]$ and $(13, +\infty)$ and check whether there are intersection points of functions $F(\pi^1, t)$ and $F(\pi^2, t)$ in these intervals. Here, for $t \in (2, 13]$, we obtain from $F(\pi^1, t) = (t + 20) \cdot 1 + 0 = F(\pi^2, t) = (t - 2) \cdot 2 + 19$ the intersection point $(5, 25)$. Thus, combining the results from the two tables for $F(\pi^1, t)$ and $F(\pi^2, t)$, we obtain Table 12 for $F_2(t)$. Note that for $t \leq 5$, the partial sequence (2,1) belonging to $F(\pi^1, t)$ is better while for $t > 5$, the partial sequence $(1, 2)$ belonging to $F(\pi^2, t)$ is better.

Table 12: Function $F_2(t)$

t	$(-\infty, -20]$	$(-20, 5]$	$(5, +\infty)$
$F_2(t)$	0	0	25
u	0	1	2
π_2	(2,1)	(2,1)	(1,2)

The graph of function $F_2(t)$ is displayed in Fig. 1 (c) obtained from Fig. 1 (b). In an analogous way, we consider the remaining jobs $l = 3, 4$.

Let $l = 3$. We obtain the following results.

We have the new break point $t' = d_3 - p_3 = 26$ and obtain function $F(\pi^1, t)$ given in Table 13.

Table 13: Function $F(\pi^1, t)$ for $l = 3$

t	$(-\infty, -32]$	$(-32, -7]$	$(-7, 26]$	$(26, +\infty)$
$F(\pi^1, t)$	0	0	25	91
u	0	1	2	3
π^1	(3,2,1)	(3,2,1)	(3,1,2)	(3,1,2)

We have the new break point $t' = d_3 - (p_1 + p_2 + p_3) = -26$ and obtain function $F(\pi^2, t)$ given in Table 14.

Table 14: Function $F(\pi^2, t)$ for $l = 3$

t	$(-\infty, -26]$	$(-26, -20]$	$(-20, 5]$	$(5, +\infty)$
$F(\pi^2, t)$	0	0	6	56
u	0	1	2	3
π^2	(2,1,3)	(2,1,3)	(2,1,3)	(1,2,3)

The intersection point of the graphs of functions $F(\pi^1, t)$ and $F(\pi^2, t)$ is $(-14, 18)$. Thus, we obtain Table 15 for function $F_3(t)$. The graph of function $F_3(t)$ is displayed in Fig. 1 (d).

Table 15: Function $F_3(t)$

t	$(-\infty, -32]$	$(-32, -14]$	$(-14, 5]$	$(5, +\infty)$
$F_3(t)$	0	0	18	56
u	0	1	2	3
π_3	(3,2,1)	(3,2,1)	(2,1,3)	(1,2,3)

Let $l = 4$. We obtain the following results:

We have the new break point $t' = d_4 - p_4 = 35$ and obtain function $F(\pi^1, t)$ given in Table 16.

Table 16: Function $F(\pi^1, t)$ for $l = 4$

t	$(-\infty, -37]$	$(-37, -19]$	$(-19, 0]$	$(0, 35]$	$(35, +\infty)$
$F(\pi^1, t)$	0	0	18	56	161
u	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)

We have the new break point $t' = d_4 - (p_1 + p_2 + p_3 + p_4) = -29$ and obtain function $F(\pi^2, t)$ given in Table 17.

Table 17: Function $F(\pi^2, t)$ for $l = 4$

t	$(-\infty, -32]$	$(-32, -29]$	$(-29, -14]$	$(-14, 5]$	$(5, +\infty)$
$F(\pi^2, t)$	0	0	3	33	90
u	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)

The intersection point of the graphs of functions $F(\pi^1, t)$ and $F(\pi^2, t)$ is $(-24, 13)$. Thus, we obtain Table 18 for function $F_4(t)$. The graph of function $F_4(t)$ is displayed in Fig. 1 (e).

Table 18: Function $F_4(t)$

t	$(-\infty, -37]$	$(-37, -24]$	$(-24, -14]$	$(-14, 5]$	$(5, +\infty)$
$F_4(t)$	0	0	13	33	90
u	0	1	2	3	4
π_4	(4,3,2,1)	(4,3,2,1)	(3,2,1,4)	(2,1,3,4)	(1,2,3,4)

The optimal objective function value is $F_4(0) = 33 + 14 \cdot 3 = 75$, and the resulting optimal schedule is $\pi_4(0) = (2, 1, 3, 4)$.

5 Concluding Remarks

In this paper, we used a graphical approach to improve a known pseudo-polynomial algorithm for problem $1||\max \sum T_j$ 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 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 made (e.g. in the problem under consideration, a job may be completed on-time or not or for a knapsack problem, an item can be put into the knapsack or not). However, for the knapsack problem, the graphical algorithm mostly reduces substantially the number of points to be considered but it does not reduce the time complexity of the algorithm. 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 has not only a practical but also a theoretical importance.

References

[1] *Du J., Leung J. Y.-T., Minimizing Total Tardiness on One Processor is NP-hard*, Math. Oper. Res., Vol. 15, 1990, 483 - 495.

[2] *Lazarev A.A., Gafarov E.R., Special Case of the Single-Machine Total Tardiness Problem is NP-hard.*, Journal of Computer and Systems Sciences International, Vol. 45, No. 3, 2006, 450 – 458.

[3] *Lawler E.L. , A Pseudopolynomial Algorithm for Sequencing Jobs to Minimize Total Tardiness*, Ann. Discrete Math., Vol. 1, 1977, 331 - 342.

[4] *Szwarc, W., Della Croce, F., Grosso, A., Solution of the Single Machine Total Tardiness Problem*, Journal of Scheduling, Vol. 2, 1999, 55 – 71.

[5] *Potts C.N., Van Wassenhove L.N., A Decomposition Algorithm for the Single Machine Total Tardiness Problem*, Oper. Res. Lett., Vol. 1, 1982, 363 - 377.

[6] *Lazarev, A.A., Werner, F., Algorithms for Special Cases of the Single Machine Total Tardiness Problem and an Application to the Even-Odd Partition Problem,* Mathematical and Computer Modelling, Vol. 49, N 9-10, 2009, 2061 – 2072.

[7] *Gafarov E.R., Lazarev A.A., Werner F., Algorithms for Maximizing the Number of Tardy Jobs or Total Tardiness on a Single Machine*, to appear in Automation and Remote Control, Vol. 9, 2010.

[8] *Gafarov E.R., Lazarev A.A., Werner F., Classical Combinatorial and Single Machine Scheduling Problems with Opposite Optimality Criteria*, Preprint 11/10, FMA, Otto-von-Guericke-Universität Magdeburg, 2010.

[9] *Lazarev, A.A., Werner, F., A Graphical Realization of the Dynamic Programming Method for Solving NP-hard Combinatorial Problems*, Computers and Mathematics with Applications, Vol. 58, 2009, 619 - 631.

[10] *Lazarev A.A., Gafarov E.R., Scheduling Theory. Total Tardiness Problem.* Moscow. Computing Center of the Russian Academy of Sciences, ISBN 5-201-09849-5, 128 pp. (in Russian)