

Solution Algorithms for the Total Tardiness Scheduling Problem on a Single Machine

Alexander Lazarev¹, Alexander Kvaratskhelia, Andrei Tchernykh²

¹ Kazan State University, Russia

Alexandr.Lazarev@mail.ru

² CICESE, Mexico

chernykh@cicese.mx

Abstract. We study the classical NP-hard in the ordinary sense single-machine total tardiness scheduling problem $1 \parallel \sum T_j$. New polynomial and pseudo-polynomial solvable cases of the problem are presented and respective algorithms are proposed with no more than $O(n^2 \sum p_j)$ time.

1 Introduction

We consider the single-machine total tardiness problem. Given a jobset $N = \{1, 2, \dots, n\}$ with n jobs have to be scheduled on a single machine without preemptions. The machine can handle only one job at a time and is available for processing at a start time of machine t_0 , typically $t_0 = 0$. Processing time $p_j > 0$ and due date d_j are given for a job $j \in N$, by default we assume $p_j \in \mathbb{Z}^+$, $d_j, t_0 \in \mathbb{R}$. A schedule π is defined as a permutation of the jobset's N elements $\pi = (j_1, j_2, \dots, j_n)$.

The problem calls for construction an optimal schedule π^* which minimizes function $F(\pi) = \sum_{j=1}^n \max\{0, c_j(\pi) - d_j\}$ where $c_j(\pi)$ is the completion time of job j in a schedule π (let $\pi = (j_1, j_2, \dots, j_n)$ then $c_{j_1}(\pi) = t_0 + p_{j_1}$ and $c_{j_k}(\pi) = c_{j_{k-1}}(\pi) + p_{j_k}$ for $k = 2, 3, \dots, n$).

The problem is known to be NP-hard in the ordinary sense [1]. Lawler proposed pseudo-polynomial algorithm in $O(n^4 \sum p_j)$ time [2] to solve arbitrary instance. Among solution approaches for the problem, the decomposition based algorithms are widely indicated and studied [2–7]. The paper [9] showed that all known constructive heuristics and a number of decomposition heuristics for the total tardiness problem present arbitrarily bad approximation ratios.

In the paper we present four algorithm for solution the problem in the case $p_1 \geq p_2 \geq \dots \geq p_n$ and $d_1 \leq d_2 \leq \dots \leq d_n$.

We use the following notation: $x = \langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$ is an instance of the problem with parameters of jobs p_j, d_j , jobset N and start time t_0 ; $\{\pi\}$ is the set of sequenced in schedule π jobs; $(i \rightarrow j)_{\pi^*}$ means that processing of job i precedes to processing of job j in schedule π (i.e. $c_i(\pi) < c_j(\pi)$); $\Pi^*(N', t')$ is the set of optimal schedules for instance $\langle \{p_j, d_j\}_{j \in N'}, t' \rangle$, $N' \subseteq N$, $t' \geq t_0$.

2 Partitioning procedure

In the paper we proposed polynomial and pseudo-polynomial algorithms for all sub-cases of the case:

$$\begin{cases} p_1 \geq p_2 \geq \dots \geq p_n \\ d_1 \leq d_2 \leq \dots \leq d_n \end{cases} \quad (1)$$

We present the following solution approach for the case (1). The initial jobset N is partitioned into subsets and, depends on the number of subsets, solution algorithm are proposed.

Partitioning procedure

0. $k := 1, \alpha_k := 1;$
1. **for** $j = 2, 3, \dots, n$ **do**:
 - if** $d_j - d_{\alpha_k} > p_j$ **then**
 - $\beta_k := j - 1;$
 - $k := k + 1;$
 - $\alpha_k := j;$
 - end if**
 - end for**
2. $\beta_k := n.$

The procedure partitions the jobset N into $k \leq n$ subsets M_1, M_2, \dots, M_k where $M_i = \{\alpha_i, \alpha_i + 1, \dots, \beta_i\}$, $\alpha_1 = 1, \beta_k = n$, $M_1 \cup M_2 \cup \dots \cup M_k = N$ and $M_i \cap M_j = \emptyset$ if $i \neq j$.

For example, if $N = \{1, 2, 3\}$, $p_1 = 10, p_2 = 10, p_3 = 2$, $d_1 = 7, d_2 = 9, d_3 = 10$ then the procedure constructs subsets $M_1 = \{1, 2\}$ and $M_2 = \{3\}$.

In the following sections we present solution algorithms for the four sub-cases depends on value k :

- pseudo-polynomial algorithm *B-1* for the case $k = 1$;
- pseudo-polynomial algorithm *B-k* for the case $1 < k < n$;
- polynomial algorithm *C-1* for the case $k = 1$ and $d_n - d_1 \leq 1$;
- polynomial algorithm *B-n* for the case $k = n$.

Algorithm *B-k* can be used to solve an arbitrary instance in the case (1) with $O(kn \sum p_j)$ time. If $k = 1$ (i.e. $d_n - d_1 \leq p_n$) then we can use algorithm *B-1* which runs in $O(n \sum p_j)$. If $d_n - d_1 \leq 1$ then algorithm *C-1* constructs an optimal schedule in $O(n^2)$ time. And finally, if $k = n$ then algorithm *B-n* constructs an optimal schedule in $O(n^2)$ time.

3 Pseudo-polynomial algorithms

To describe pseudo-polynomial algorithms *B-1* and *B-k* we need to define values $d_j(t)$:

$$d_j(t) = d_j - d_n + t - t_0, \quad j \in N.$$

Due to (1), for each $t \in \mathbb{R}$ we have $d_1(t) \leq d_2(t) \leq \dots \leq d_n(t)$. Let $\pi_l(t)$ and $F_l(t)$ be an optimal schedule and optimal total tardiness value for the parametric instance $\langle \{p_j, d_j(t)\}_{j \in N_l}, 0 \rangle$ where $N_l = \{l, l+1, \dots, n\}$.

Well known property of the total tardiness problem states that two instances $x_1 = \langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$ and $x_2 = \langle \{p_j, d_j + C\}_{j \in N}, t_0 + C \rangle$, where C is arbitrary constant, have the same sets of optimal schedules, i.e. the instances x_1 and x_2 are equal.

Due to this property, initial instance $x = \langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$ and corresponding parametric instance $\langle \{p_j, d_j(t)\}_{j \in N}, 0 \rangle$ are equal if $t = d_n$.

The main idea of the algorithms *B-1* and *B-k* is the following. Consecutively, for $l = n, n-1, \dots, 1$, schedules $\pi_l(t)$ are constructed and values $F_l(t)$ computed for each point $t \in \mathbb{Z}^+$. Optimal schedule for the initial instance x is $\pi_1(d_n)$.

Notice that if $d_n \notin \mathbb{Z}^+$ we construct and solve instance $\langle \{p_j, d'_j\}_{j \in N}, t'_0 \rangle$, $d'_j = d_j - \Delta$, $t'_0 = t_0 - \Delta$ where $\Delta = d_n - \lfloor d_n \rfloor$, $\lfloor d_n \rfloor$ is the integer part of value d_n . In this case d'_n is integer value and as we showed above schedule $\pi_1(d'_n)$ is optimal schedule for instance $\langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$.

3.1 Case $k = 1$

In the case $k = 1$, parameters of jobs satisfy conditions:

$$\begin{cases} p_1 \geq p_2 \geq \dots \geq p_n \\ d_1 \leq d_2 \leq \dots \leq d_n \\ d_n - d_1 \leq p_n \end{cases} \quad (2)$$

We proved the following main property for the case (2).

Property B-1. *Every schedule $\pi = (\pi_1, i, \pi_2, l, \pi_3, j, \pi_4)$, where $l < \min\{i, j\}$, can be eliminated from consideration without lost of optimality.*

Property B-1 predicates that there exists an optimal schedule π^* where for all $l = n-1, \dots, 1$

- either $(l \rightarrow j)_{\pi^*}$ for all $j \in \{l+1, \dots, n\}$,
- or $(j \rightarrow l)_{\pi^*}$ for all $j \in \{l+1, \dots, n\}$.

Based on this property the Algorithm *B-1* of solution the problem in the case (2) are proposed.

Algorithm B-1. Solution the problem in the case (2).

0. $\pi_n(t) := (n)$, $F_n(t) := \max\{0, p_n + t_0 - t\}$;
1. for $l = n - 1, n - 2, \dots, 1$:
 $\pi^1 := (l, \pi_{l+1}(t - p_l))$, $\pi^2 := (\pi_{l+1}(t), l)$;
 $F(\pi^1) := \max\{0, p_l - d_l(t)\} + F_{l+1}(t - p_l)$;
 $F(\pi^2) := F_{l+1}(t) + \max\{0, \sum_{j=l}^n p_j - d_l(t)\}$;
 $F_l(t) := \min\{F(\pi^1), F(\pi^2)\}$;
 $\pi_l(t) := \arg \min\{F(\pi^1), F(\pi^2)\}$.
end for
2. $\pi^* := \pi_1(d_n)$ and $F(\pi^*) := F_1(d_n)$.

For each $l < n$ and t optimal schedule $\pi_l(t)$ is the better of schedules $(l, \pi_{l+1}(t - p_l))$ and $(\pi_{l+1}(t), l)$. If $t \leq t_0 + p_n$ then all jobs are late and $\pi_l(t)$ is SPT sequence $(n, n - 1, \dots, l)$; if $t \geq t_0 + \sum_{j=1}^n p_j$ then no more than one job is late and $\pi_l(t)$ is EDD sequence $(l, l + 1, \dots, n)$. Due to SPT and EDD sequences can be computed before algorithm runs, we need to process only integer points $t : t_0 + p_n < t < t_0 + \sum_{j=1}^n p_j$. Algorithm B-1 construct schedule $\pi_1(d_n)$ in $O(n \sum p_j)$ time.

3.2 Case $1 < k < n$

In this case, parameters of jobs satisfy conditions

$$\left\{ \begin{array}{ll} d_1 \leq d_2 \leq \dots \leq d_n & \\ p_1 \geq p_2 \geq \dots \geq p_n & \\ d_{\beta_1} - d_{\alpha_1} \leq p_{\beta_1} & \alpha_1 = 1 \\ d_{\beta_2} - d_{\alpha_2} \leq p_{\beta_2} & \alpha_2 = \beta_1 + 1 \\ \dots & \\ d_{\beta_k} - d_{\alpha_k} \leq p_{\beta_k} & \beta_k = n \end{array} \right. \quad (3)$$

Property B-k/1. Every schedule $\pi = (\pi_1, j, \pi_2, l, \pi_3, l+1, \pi_4)$, where $j > (l+1)$, can be eliminated from consideration without lost of optimality.

Property B-k/2. Every schedule $\pi = (\pi_1, i, \pi_2, l, \pi_3, j, \pi_4)$, where $l < \min\{i, j\}$ and i, j belong to the same subset M_ν , can be eliminated from consideration without lost of optimality.

Properties B-k/1 and B-k/2 predicate that there exist an optimal schedule π^* where job l is processed either before all jobs of set $\{l + 1, \dots, n\}$, or after job $(l + 1)$ and is not processed between jobs $i, j : l < \min\{i, j\}$, jobs i, j belong to the same subset M_ν .

Based on these properties, Algorithm B-k of solution the problem in the case (3) are constructed. Due to high complication of formal description of the algorithm, we show the main idea. Algorithm B-k starts as algorithm B-1 by

constructing schedules $\pi_n(t)$. Then for every $l = n - 1, n - 1, \dots, 1$ and t , the algorithm constructs schedules $\pi^1, \pi^2, \dots, \pi^g$, $g \leq k$, g is count of positions for job l predicated by the properties in schedule $\pi_{l+1}(t)$, and select the best one which is schedule $\pi_l(t)$. Algorithm *B-k* constructs schedule $\pi_1(d_n)$ in $O(kn \sum p_j)$ time.

4 Polynomial algorithms

4.1 Case $k = 1$ and $d_n - d_1 \leq 1$

Suppose that parameters of jobs satisfy conditions

$$\begin{cases} d_1 \leq d_2 \leq \dots \leq d_n, \\ d_n - d_1 \leq 1, \\ t_0 \in \mathbb{Z}^+ \end{cases} \quad (4)$$

without restrictions $p_1 \geq p_2 \geq \dots \geq p_n$.

In this case we proved the following properties which allow to construct polynomial solution algorithm. Consider sub-instance with jobset $N' \subseteq N$ and start time t_0 . Let $S = t_0 + \sum_{j \in N'} p_j$ and $z = \lfloor d_n \rfloor$ (z is the integer part of value d_n).

Properties C-1. *There exists an optimal schedule $\pi^* \in \Pi(N', t_0)$ such that $\pi^* = (\pi, j^*)$ and $\pi \in \Pi^*(N' \setminus \{j^*\}, t_0)$ where*

1. *if $S - p_j > z$ for all $j \in N'$ then $j^* = \arg \max_{j \in N'} \{d_j : p_j = \max_{i \in N'} p_i\}$;*
2. *if $S - p_j \leq z$ for some $j \in N'$ then $S - p_{j^*} \leq z + 1$;*
3. *if $S \leq z$ then $j^* = \arg \max_{j \in N'} \{d_j\}$.*

Algorithm C-1. Solution the problem in the case (4).

0. $S := t_0 + \sum_{j=1}^n p_j$, $N' := N$, $\pi^* := \emptyset$, $\pi_D := (1, 2, \dots, n)$;
1. **while** $\{j \in N' : S - p_j \leq z\} = \emptyset$ **and** $N' \neq \emptyset$ **do**:
 $\pi^* := (j^*, \pi^*)$, where $j^* := \arg \max_{j \in N'} \{d_j : p_j = \max_{i \in N'} p_i\}$;
 $S := S - p_{j^*}$, $N' := N' \setminus \{j^*\}$, $\pi_D := \pi_D \setminus \{j^*\}$;
end while
2. **if** $N' = \{j\}$ **then** $\pi^* := (j, \pi^*)$; **stop**.
3. **for all** $j \in N' : S - p_j \leq z + 1$ **do**:
for all $i \in N' \setminus \{j\}$ **do**:
 $\pi_{ij} := (\pi_D \setminus \{i, j\}, i, j)$;
end for
end for
4. $\pi^* := (\pi, \pi^*)$, where $\pi := \arg \min_{i, j} F(\pi_{ij})$.

Algorithm *C-1* starts with jobset N and due to Properties C-1 on each step determines the job which has to be processed after all unsequenced jobs $j \in N'$ and before sequenced jobs $j \in \{\pi^*\}$ in the optimal solution. Algorithm *C-1* constructs an optimal schedule π^* in $O(n^2)$ time.

4.2 Case $k = n$

Consider the case of the problem $1 \parallel \sum T_j$ where

$$d_j - d_{j-1} > p_j, \quad j = 2, 3, \dots, n, \quad (5)$$

without restrictions $p_1 \geq p_2 \geq \dots \geq p_n$ and $p_j \in \mathbb{Z}^+$.

In this case we present polynomial algorithm based on the following property.

Consider sub-instance with jobset $N' \subseteq N$ and start time $t' \geq t_0$, $N' = \{j_1, j_2, \dots, j_m\}$ and $d_{j_1} \leq d_{j_2} \leq \dots \leq d_{j_m}$. Let $j^* = \arg \max_{j \in N'} \{d_j : p_j = \max_{i \in N'} p_i\}$ and $S_\alpha := t' + p_{j_1} + p_{j_2} + \dots + p_{j_\alpha}$, $\alpha = 1, 2, \dots, m$.

Property B-n. *There exists an optimal schedule $\pi^* \in \Pi^*(N', t')$ such that*

$$(j \rightarrow j^*)_{\pi^*} \text{ for all } j \in \{j_1, j_2, \dots, j_{\alpha^*}\} \setminus \{j^*\} \text{ and} \\ (j^* \rightarrow j)_{\pi^*} \text{ for all } j \in \{j_{\alpha^*+1}, \dots, j_m\}.$$

where $\alpha^* = \min\{\alpha \in \{1, \dots, m\} : p_j + d_j \leq S_\alpha < d_{j_{\alpha+1}}, j \in \{j^* + 1, \dots, j_\alpha\}\}$, in addition $d_{j_{m+1}} = +\infty$.

In the notation of decomposition approaches [2–7], Property B-n means that there exists an optimal schedule π^* where the largest processing time job j^* is processed on the first "suitable" position α^* and the sub-instance N', t' is decomposed by the position α^* .

Procedure ProcB-n (N', t')

0. Given set of jobs $N' \subseteq N$, $N' = \{j_1, j_2, \dots, j_m\}$ and start time $t' \geq t_0$, $d_{j_1} \leq d_{j_2} \leq \dots \leq d_{j_m}$;
 1. $S_\alpha := t' + p_{j_1} + p_{j_2} + \dots + p_{j_\alpha}$, $\alpha = 1, 2, \dots, m$;
 $j^* = \arg \max_{j \in N'} \{d_j : p_j = \max_{i \in N'} p_i\}$;
 2. $\alpha^* := \min_{\alpha=1, \dots, m} \{\alpha : d_j + p_j \leq S_\alpha < d_{j_{\alpha+1}}, j \in \{j^* + 1, \dots, j_\alpha\}\}$;
 3. $N_1 := \{j_1, \dots, j_{\alpha^*}\} \setminus \{j^*\}$, $t_1 := t'$;
 $N_2 := \{j_{\alpha^*+1}, \dots, j_m\}$, $t_2 := S_{\alpha^*}$;
 4. Return schedule $\pi^* := (\pi_1, j^*, \pi_2)$ where
 $\pi_1 := \mathbf{ProcB-n}(N_1, t_1)$,
 $\pi_2 := \mathbf{ProcB-n}(N_2, t_2)$.
-

Algorithm B-n. Solution the problem in the case (5).

$$\pi^* := \mathbf{ProcB-n}(N, t_0)$$

Algorithm B-n constructs an optimal schedule π^* in $O(n^2)$ time.

5 Conclusion

In the paper we consider NP-hard in the ordinary sense single-machine total tardiness scheduling problem. The complex research of the case (1) has been made. Four algorithms are proposed to solve all possible sub-cases of the case with no more than $O(n^2 \sum p_j)$ time. In the future we will study the hypothesis that the general case of the problem (without restrictions (1)) can be solved with no more than $O(n^2 \sum p_j)$ time (that is essentially less than in the Lawler algorithm).

References

1. J. Du and J. Y.-T. Leung: Minimizing total tardiness on one processor is NP-hard. *Math. Oper. Res.*, **15** (1990), pp. 483-495.
2. E.L. Lawler: A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness *Ann. Discrete Math.*, **1** (1977), pp. 331-342.
3. C.N. Potts and L.N. Van Wassenhove: A decomposition algorithm for the single machine total tardiness problem, *Oper., Res. Lett.*, **1** (1982), pp. 177-182.
4. S. Chang, Q. Lu, G. Tang, and W. Yu: On decomposition of the total tardiness problem, *Oper. Res. Lett.*, **17** (1995), pp. 221-229.
5. W. Szwarc, F. Della Croce and A. Grosso: Solution of the single machine total tardiness problem. *J. Sched.*, **2** (1999), pp.55-71.
6. W. Szwarc, A. Grosso and F. Della Croce: Algorithmic paradoxes of the single machine total tardiness problem. *J. Sched.*, **4** (2001), pp. 93-104.
7. A.A. Lazarev: Efficient algorithms of decisions some problems of theory of scheduling for single machine with deadline terms of service jobs. PhD Thesis, 1989, (in Russian).
8. H. Emmons: One machine sequencing to minimizing certain function of job tardiness. *Oper. Res.*, **17** (1969), pp. 701-715.
9. F. Della Croce, A. Grosso, V. Paschos: Lower bounds on the approximation ratios of leading heuristics for the single-machine total tardiness problem. *J. Sched.* **7** (2004), pp. 85-91.