Algorithms for Special Single Machine Total Tardiness Problems

Alexander A. Lazarev

Institute of Control Sciences of the Russian Academy of Sciences,
Profsoyuznaya st. 65, 117997 Moscow, Russia,
email: 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

July 11, 2008

Abstract

The scheduling problem of minimizing total tardiness on a single machine is known to be NP-hard in the ordinary sense. In this paper, we consider the special case of the problem when the processing times p_j and the due dates d_j of the jobs j, $j \in N = \{1, 2, \ldots, n\}$, are oppositely ordered: $p_1 \geq p_2 \geq \ldots \geq p_n$ and $d_1 \leq d_2 \leq \ldots \leq d_n$. It is shown that already this special case is NP-hard in the ordinary sense, too. The set of jobs N is partitioned into $\Bbbk, 1 \leq \Bbbk \leq n$, subsets $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{\Bbbk}$, $\mathcal{M}_\nu \bigcap \mathcal{M}_\mu = \emptyset$ for $\nu \neq \mu$, $N = \mathcal{M}_1 \bigcup \mathcal{M}_2 \bigcup \ldots \bigcup \mathcal{M}_{\Bbbk}$, such that $\max_{i,j \in \mathcal{M}_\nu} |d_i - d_j| \leq \min_{j \in \mathcal{M}_\nu} p_j$ for each $\nu = 1, 2, \ldots, \Bbbk$. We propose algorithms which solve the problem: in $O(\Bbbk n \sum p_j)$ time if $1 \leq \Bbbk < n$; in $O(n^2)$ time if $\Bbbk = n$; and in $O(n^2)$ time if $\max_{i,j \in N} |d_i - d_j| \leq 1$. The polynomial algorithms do neither require the conditions $p_1 \geq p_2 \geq \ldots \geq p_n$ mentioned above nor integer processing times to construct an optimal schedule. Finally, we apply the idea of the presented algorithm for the case $\Bbbk = 1$ to the even-odd partition problem.

Keywords: scheduling, single machine, total tardiness, even-odd partition, NP-hardness

1 Introduction

In the paper, we consider the problem of minimizing total tardiness for a set $N = \{1, 2, \ldots, n\}$ of n independent jobs on a single machine. Processing of the jobs may start no earlier than time $t_0 \in \mathbb{R}$ when all jobs are assumed to be available. The machine can process at most one job at any time, and preemptions of the processing of a job are forbidden. For any job $j \in N$, a processing time $p_j \in \mathbb{Z}^+$ and a due date $d_j \in \mathbb{R}$ are given. A schedule π is defined as a permutation of the set of jobs N. Let $C_j(\pi)$ be the *completion time* of job j in schedule π. This means that, for example, if $\pi = (j_1, j_2, \ldots, j_n)$, then $C_{j_k}(\pi) = t_0 + \sum_{i=1}^{k} p_{j_i}, k = 1, 2, \ldots, n$. The *total tardiness* problem requires the construction of a schedule π^* that minimizes

$$F(\pi) = \sum_{j=1}^{n} T_j(\pi) = \sum_{j=1}^{n} \max\{0, C_j(\pi) - d_j\},$$

where $T_j(\pi)$ denotes the tardiness of job j in schedule π.

It has been proved that this single machine problem is NP-hard in the ordinary sense by means of a polynomial reduction from the NP-complete even-odd partition problem to special cases of the total tardiness problem (see [1, 2, 3]). The total tardiness problem has been studied by Emmons [4] who proposed the following rule: if for two jobs $i, j \in N$, we have $p_i \leq p_j$ and $d_i \leq d_j$, then there exists an optimal schedule, where job i is processed before job j. Lawler [5] proved a decomposition theorem and proposed a pseudo-polynomial time algorithm that constructs an optimal schedule in $O(n^4 \sum p_j)$ time. This algorithm was also used as a base for developing a fully polynomial approximation scheme with complexity $O(n^7/\epsilon)$ [6]. The decomposition property of the total tardiness problem gives an idea to generate rules which reduce the complexity of the decomposition algorithm. Some decomposition rules have been introduced in [7, 8, 9, 10]. Szwarc et al. [10, 11, 12, 13] have given algorithms based on the known decomposition rules and bounds for the optimal tardiness value. This algorithm has solved test instances with up to $n = 500$ jobs. As far as heuristic algorithms are concerned, Brucker et al. [14] presented an iterated local search algorithm for the total tardiness problem which they tested on problems with up to 1000 jobs. Several other heuristic algorithms have been given even for more general problems including e.g. job weights [15] or sequence-dependent setup times [16]. For a more extensive literature survey, we refer the reader to [3, 17, 18].

After introducing some basic concepts in Section 2, we discuss decomposition properties of the problem in Section 3. Section 4 is the main section and devoted to algorithms for all subcases of oppositely ordered processing times and due dates, i.e. $p_1 \geq p_2 \geq \ldots \geq p_n$ and $d_1 \leq d_2 \leq \ldots \leq d_n$. In Section 5, we present a new polynomially solvable case of the problem when $\max_{i,j \in N} |d_i - d_j| \leq 1$. Finally, in Section 6 we use the idea of Algorithm B-1 presented in Section 4 for the solution of the NP-complete even-odd partition problem.

2 Preliminaries and Notations

We denote by $I = \langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$ an instance with the set of jobs N, the processing times p_j, the due dates d_j, and a given starting time t_0 of the machine. We denote the initial instance I by the pair $\{N, t_0\}$. Let $\Pi(I)$ be the set of all $n!$ possible permutations and $\Pi^*(I)$ be the set of all optimal schedules for instance I.

Furthermore, we introduce a *parametric instance* as follows. Let $d_j(t) = d_j - d_n + t - t_0$ be the parameterized due date for job $j \in N$. Without loss of generality, we assume $d_1 \leq d_2 \leq \ldots \leq d_n$, so that $d_1(t) \leq \ldots \leq d_n(t)$ holds for any real t. For the set $N_k = \{k, k+1, \ldots, n\}$ of jobs which is given for each $k = n, n-1, \ldots, 1$, the parametric instance is denoted by $I_k(t) = \langle \{p_j, d_j(t)\}_{j \in N_k}, 0 \rangle$. Let $\pi_k^*(t)$ and $F_k^*(t)$ be an optimal schedule and the optimal value of the total tardiness function for the instance $I_k(t)$. Let $\{\pi\}$ denote the set of jobs processed in the schedule π. In the following, we use the notation $\pi = (\pi_1, j, \pi_2)$, where π_1 and π_2 are subschedules of π such that π_1 and π_2 contain the preceding and succeeding jobs of j and the sequence of jobs in both subschedules is the same as in π. As a generalization, we may also use two particular jobs to describe a sequence in the form $\pi = (\pi_1, i, \pi_2, j, \pi_3)$. If the processing of a job i precedes the processing of a job j in a schedule π, which implies $C_i(\pi) < C_j(\pi)$, the notation $(i \rightarrow j)_\pi$ is used. In a more general form, the notation $(i \rightarrow j \rightarrow k)_\pi$ is used to describe precedence relations between three jobs in the schedule π.

An instance can be modified by changing the due dates. Let us consider the two instances $I = \langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$ and $I' = \langle \{p'_j, d'_j\}_{j \in N}, t'_0 \rangle$. These instances are called *equivalent* if any optimal schedule for I is also optimal for I' and vice versa. Hence, the sets of optimal schedules for both instances are equal. One can show that, if $p'_j = p_j$, $d'_j = d_j + \tau$, $j \in N$, $t'_0 = t_0 + \tau$,

where τ is an arbitrary real constant, then I and I' are *equivalent*. This follows from

$$\begin{aligned} T_j(\pi) &= \max\left\{0, t_0 + \sum_{i:(i\to j)_\pi} p_i + p_j - d_j\right\} \\ &= \max\left\{0, t_0 + \tau + \sum_{i:(i\to j)_\pi} p_i + p_j - (d_j + \tau)\right\} \end{aligned}$$

for each schedule π and each job $j \in N$. This means that the starting time of each instance can be assumed to be $t_0 = 0$. Moreover, if $p'_j = \alpha p_j$, $d'_j = \alpha d_j$, and $t'_0 = \alpha t_0$ for an arbitrary constant $\alpha > 0$, the instances I and I' are also equivalent.

Without loss of generality, we assume that the due dates belong to the interval $[t_0, t_0 + \sum_{j=1}^{n} p_j]$ due to the following reasons. For any given instance I, let us construct an instance I', where $p'_j = p_j$, $d'_j = \min\{\max\{t_0, d_j\}, t_0 + \sum_{i=1}^{n} p_i\}$ and $t'_0 = t_0$. If we have $d_j > t_0 + \sum_{i=1}^{n} p_i$ for job j, then $d'_j = t_0 + \sum_{i=1}^{n} p_i$, and job j is early in each schedule. Hence, j can be processed on the last position in all optimal schedules for both instances I and I'. If $d_j < t_0$, then job j is tardy in each schedule π, which implies $d_j < C_j(\pi)$ so that $d'_j = \max\{t_0, d_j\} = t_0$. In this case, according to [5], any optimal schedule for I' is also an optimal one for I.

3 Decomposition Property

A decomposition property of the problem has been studied by Lawler [5]. Assume that the set N of jobs is ordered such that $d_1 \le d_2 \le \ldots \le d_n$, if $d_j = d_{j+1}$, then $p_j \le p_{j+1}$. Let j^* denote the job with the largest processing time in N, i.e. $j^* = \arg\max_{j\in N}\{d_j : p_j = \max_{i\in N} p_i\}$, and let $S_k = t_0 + \sum_{j=1}^{k} p_j$, $k = 1, 2, \ldots, n$. Lawler has proved that there exists an optimal schedule π^*, where for some position $k \ge j^*$ in the schedule, we have: $(j \to j^*)_{\pi^*}$ holds for all $j \le k$, $j \ne j^*$, and $(j^* \to j)_{\pi^*}$ holds for all $j > k$. Lawler has also proposed a decomposition-based algorithm. We illustrate the idea of this algorithm in terms of the initial instance $I = \{N, t_0\}$. For each $k \ge j^*$, the algorithm constructs an optimal schedule in which set $N' = \{1, \ldots, k\} \setminus \{j^*\}$ of jobs is scheduled before j^*, and set $N'' = \{k+1, \ldots, n\}$ is scheduled after j^*. Jobs

of N' are to be scheduled optimally with the starting time $t' = t_0$, and the jobs of N'' with the starting time $t'' = S_k$. This means that instance I is decomposed into two subinstances $I' = \{N', t'\}$ and $I'' = \{N'', t''\}$. The best of the constructed schedules for each $k \geq j^*$ is an optimal schedule for the initial instance.

This approach determines an optimal schedule in $O(n^4 \sum p_j)$ time. Later, some decomposition rules have been proposed which allow one to reduce the number of positions k on which j^* is sequenced. These rules have been introduced by Potts and van Wassenhove [7], Lazarev [8], Chang et al. [9], and Szwarc [10].

For an instance $\{N, t\}$, let us define the set $L(N, t)$ of all positions $k \geq j^*$ such that:

- $d_j + p_j \leq S_k$ holds for all $j^* + 1 \leq j \leq k$;
- $S_k < d_{k+1}$,

where additionally $d_{n+1} := +\infty$ is defined. Then the following theorem holds.

Decomposition Theorem [5, 7, 10] *There exists an optimal schedule π^*, where for some $k \in L(N, t)$, $(j \to j^*)_{\pi^*}$ holds for all jobs $j \in \{1, 2, \ldots, k\} \setminus \{j^*\}$ and $(j^* \to j)_{\pi^*}$ holds for all jobs $j \in \{k+1, \ldots, n\}$.*

The decomposition property of the problem suggests the following recursive procedure Sequence(N', t') which constructs an optimal schedule for the set $N' \subseteq N$ of jobs starting at time $t' \geq t_0$. To simplify notation, we present the formal description of the procedure in terms of the initial instance $\{N, t_0\}$.

Sequence (N, t)

1: Let $N = \{j_1, j_2, \ldots, j_n\}$;
2: **if** $N = \emptyset$ **then**
3: π^*:= empty schedule;
4: **else**
5: Find job j^* in N and the set $L = L(N, t)$;
6: **for all** $k \in L$ **do**
7: $N' := \{j_1, \ldots, j_k\} \setminus \{j^*\}$, $t' := t$; $N'' := \{j_{k+1}, \ldots, j_n\}$, $t'' := S_k$;
8: $\pi_k := ($**Sequence** $(N', t'),\ j^*,$ **Sequence** $(N'', t''))$;
9: **end for**
10: π^*:=arg $\min_{k \in L}\{F(\pi_k)\}$;

11: **end if**
12: **return** π^*.

Decomposition Algorithm: $\pi^* :=$ **Sequence** (N, t_0)**.**

This decomposition algorithm can be improved by more recent decomposition rules (see Chang et al. [9]) and bounds for the optimal value (see Szwarc et al. [12, 13]). However, the decomposition algorithm in the above formulation is sufficient for the following investigations. We note that algorithms (analogue to the above decomposition algorithm) with a complexity of $O(n2^{(n-1)/3-1})$ operations using additional rules by Chang et al. for canonical instances have been given (see e.g. Gafarov and Lazarev [2, 3]). Among the canonical instances, there is a class of subinstances, denoted by B-F (when in all $n!$ schedules exactly k jobs are tardy) for which an algorithm with the complexity $O(n^3)$ has been constructed [3, 19].

4 Algorithms for Oppositely Ordered Processing Times and Due Dates

Let us assume that the processing times and due dates are oppositely ordered:

$$\begin{cases} p_1 \geq p_2 \geq \ldots \geq p_n, \\ d_1 \leq d_2 \leq \ldots \leq d_n. \end{cases} \tag{1}$$

In Section 6 (where we deal with the even-odd partition problem), we will show that the above special case of the total tardiness problem is NP-hard in the ordinary sense. Given an arbitrary instance I, let us consider a partition of the set N of jobs into $\Bbbk$ subsets $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\Bbbk$ such that $\max_{i,j \in \mathcal{M}_\nu} |d_i - d_j| \leq \min_{j \in \mathcal{M}_\nu} p_j$ for each $\nu = 1, 2, \ldots, \Bbbk$. Obviously, this partition can be done in polynomial time, namely with $O(n)$ operations. We introduce three algorithms, namely B-1, B-$\Bbbk$ and B-n, to solve all subcases of (1) differing in the number $\Bbbk$ of subsets $\mathcal{M}_\nu$ given by the partition. Algorithm B-1 finds an optimal schedule for the case $\Bbbk = 1$ in $O(n \sum p_j)$ time, Algorithm B-$\Bbbk$ for $1 < \Bbbk < n$ in $O(\Bbbk n \sum p_j)$ time, Algorithm B-n for $\Bbbk = n$ in $O(n^2)$, and Algorithm C-1 (presented in Section 5) for the case when $d_{max} - d_{min} \leq 1$ in $O(n^2)$ time, too (d_{min} denotes the minimal and d_{max} the maximal due date). We note that Algorithms B-n and C-1 do neither require the conditions $p_1 \geq p_2 \geq \ldots \geq p_n$ nor integer processing times to construct an optimal schedule.

For a partition of set N, we use the following notations. The symbol ν is used for indexing the subsets $\mathcal{M} \subseteq N$, and $\gamma(j)$ denotes the index of the subset that contains job $j \in N$, i.e. $j \in \mathcal{M}_{\gamma(j)}$ holds by definition. In what follows, we use the notations α_ν, β_ν to denote the jobs with the smallest and largest numbers of the subset $\mathcal{M}_\nu$, $\nu = 1, 2, \ldots, \Bbbk$, i.e. $\mathcal{M}_\nu = \{\alpha_\nu, \ldots, \beta_\nu\}$ with $\alpha_\nu < \ldots < \beta_\nu$.

Let us consider the following procedure which partitions the set of jobs N into $\Bbbk$ subsets $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\Bbbk$, $\mathcal{M}_\nu \bigcap \mathcal{M}_\mu = \emptyset$ for $\nu \neq \mu$, and $N = \mathcal{M}_1 \bigcup \mathcal{M}_2 \bigcup \ldots \bigcup \mathcal{M}_\Bbbk$, such that $\max_{i,j \in \mathcal{M}_\nu} |d_i - d_j| \leq \min_{j \in \mathcal{M}_\nu} p_j$ for each $\nu = 1, 2, \ldots, \Bbbk$. Such a partition can be done in several ways, we present the following one:

Partitioning Procedure

1: $\Bbbk := 1$, $\alpha_1 := 1$;
2: **for** $j = 2, 3, \ldots, n$ **do**
3: **if** $d_j - d_{\alpha_\Bbbk} > p_j$ **then**
4: $\beta_\Bbbk := j - 1$; $\Bbbk := \Bbbk + 1$; $\alpha_\Bbbk := j$;
5: **end if**
6: **end for**
7: $\beta_\Bbbk := n$; $\mathcal{M}_\nu = \{\alpha_\nu, \alpha_{\nu+1}, \ldots, \beta_\nu\}$, $\nu = 1, 2, \ldots, \Bbbk$.

The procedure runs in $O(n)$ time. As an 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 the two subsets $\mathcal{M}_1 = \{1, 2\}$ and $\mathcal{M}_2 = \{3\}$.

4.1 Properties of an Optimal Schedule

In this subsection, we present some properties of an optimal schedule which are used to develop some algorithms presented in the next subsections.

Lemma 1 *Assume that conditions (1) hold and that* $| d_i - d_j | \leq \min\{p_i, p_j\}$ *and* $k < \min\{i, j\}$ *for jobs* $i, j, k \in N$. *Then there exists an optimal schedule* π^*, *where either* $(k \rightarrow i)_{\pi^*}$ *or* $(j \rightarrow k)_{\pi^*}$.

Consider the subset $Q = \{q, q+1, \ldots, r\}$ of jobs such that $d_r - d_q \leq p_r$ and $k < q$. Then there exists an optimal schedule in which job k is processed before or after all jobs in Q, i.e. one can eliminate all schedules π with $(i \rightarrow k \rightarrow j)_{\pi^*}$ for some $i, j \in Q$ from the consideration.

Lemma 2 *There exists an optimal schedule π^* for case (1), where either $(k \to j)_{\pi^*}$ or $((k+1) \to k)_{\pi^*}$ holds for each pair of jobs k, j with $(j \to (k+1))_{\pi^*}$ and $k < j$.*

If $(j \to k \to (k+1))_\pi$ for some schedule π and jobs k, j with $k > j$, then schedule π can be eliminated from the consideration when searching an optimal schedule.

According to (1), we have $d_1(t) \le d_2(t) \le \ldots \le d_n(t)$ for the parameterized due dates, where $t \in \mathbb{R}$ is arbitrary. The instances $I_k(t)$ and $\langle \{p_j, d_j\}_{j \in N_k}, t_0' \rangle$ are equivalent if $t_0' = d_n - t + t_0$ (see Section 2). In particular, $I_1(t)$ is equivalent to the initial instance I if $t = d_n$. We use these parametric instances in the process of constructing an optimal schedule for the initial instance.

Let $\pi_k^*(t)$ be an optimal schedule for the instance $I_k(t)$. Suppose that optimal schedules $\pi_{k+1}^*(t)$ have been constructed for the parametric instance $I_{k+1}(t)$ at each point t, and suppose that $d_n - d_{k+1} \le p_n$, i.e. $\Bbbk = 1$. Due to Lemma 1, job k need to be considered only on two possible positions in an optimal schedule for the instance $I_k(t)$: before and after all jobs from N_{k+1}. Therefore, $\pi_k^*(t)$ is the best of the two schedules $(k, \pi_{k+1}^*(t - p_k))$ and $(\pi_{k+1}^*(t), k)$. To construct the first schedule, we need to use the schedule $\pi_{k+1}^*(t - p_k)$ since, when job k is sequenced on the first position, we increase the starting time of the jobs of set N_{k+1} by the value p_k. For the parametric instance, such an increase in the starting times can be described by a decrease in the parameterized due dates $d_j(t)$ for all $j \in N_{k+1}$. Moreover, the calculation of the values $F_k^*(t)$ by comparing the schedules $(k, \pi_{k+1}^*(t - p_k))$ and $(\pi_{k+1}^*(t), k)$ does not require $O(n)$ time, but only $O(1)$ time. Clearly, the total tardiness values of these schedules are $\max\{0, p_k - d_k(t)\} + F_{k+1}^*(t - p_k)$ and $F_{k+1}^*(t) + \max\{0, \sum_{j=k}^n p_j - d_k(t)\}$, respectively. This discussion illustrates the basic idea of our approach: concerning the partition of N into the subsets $\mathcal{M}_\nu$, the schedules $\pi_k^*(t)$ are constructed in the order $k = n, n-1, \ldots, 1$ at each integer point t based on the schedules $\pi_j^*(t)$ for $j > k$. Notice that $\pi_n^*(t) = (n)$ and $F_n^*(t) = \max\{0, p_n - t + t_0\}$. Since the initial instance I is equivalent to the instance $I_1(t)$ if $t = d_n$, an optimal schedule for I is given by $\pi_1^*(d_n)$. This allows us to consider only those instances, where d_n is an integer value. If $d_n \notin \mathbb{Z}$, then we construct and solve the instance $I' = \langle \{p_j, d_j'\}_{j \in N}, t_0' \rangle$, where $d_j' = d_j - \Delta$, $t_0' = t_0 - \Delta$ and $\Delta = d_n - \lfloor d_n \rfloor$. In this case, I' is equivalent to I and $d_n' \in \mathbb{Z}$.

If $t \le t_0 + \min_{j\in N} p_j$, then for the parametric instance $\langle \{p_j, d_j(t)\}_{j\in N}, 0\rangle$ we have $d_j(t) \le d_n(t) = t - t_0 \le p_n \le p_j,\ j \in N$. This means that all jobs are tardy in each schedule and the SPT schedule is optimal for this instance. If $t \ge t_0 + 2\sum_{j=1}^{n} p_j$, then we have $t + \sum\limits_{j=1}^{n} p_j \ge t_0 + 2\sum\limits_{j=1}^{n} p_j + d_n - d_1$ due to $d_n - d_1 \le \sum_{j=1}^{n} p_j$. This implies

$$\sum_{j=1}^{n} p_j \le d_1 - d_n + t - t_0 = d_1(t) \le d_j(t),\ j \in N.$$

Therefore, in each schedule all jobs are early, i.e. all schedules are optimal, and the optimal total tardiness value is equal to 0. The above reasons imply that, without loss of optimality, we can eliminate the following points t from the consideration: $t > d_n$ since an optimal schedule for the initial instance is obtained by means of point $t = d_n$; $t < t_0$ since at these points optimal schedules are known (namely the SPT schedules) for all $I_k(t)$ and they can be constructed before the algorithm starts.

Since $t_0 < d_n \le t_0 + \sum_{j=1}^{n} p_j$, it follows that we need to look among all integer points in the interval $[t_0; d_n]$ of a length not more than $\sum_{j=1}^{n} p_j$.

4.2 Algorithm B-1

In this subsection, we consider a subcase of case (1) with

$$\begin{cases} p_1 \ge p_2 \ge \ldots \ge p_n, \\ d_1 \le d_2 \le \ldots \le d_n, \\ d_n - d_1 \le p_n. \end{cases} \tag{2}$$

Hence, $d_1 \le d_2 \le \ldots \le d_n \le d_1 + p_n$ and $\Bbbk = 1$, i.e. $\mathcal{M}_1 = N$.

In this case, we can eliminate schedule π from the consideration, if there exists a job $k \in N$ such that $(i \to k \to j)_\pi$ for all $i, j \in \{k+1, k+2, \ldots, n\}$ due to Lemma 1. Based on this property, we propose the following Algorithm B-1.

Algorithm B-1

1: $\pi_n(t) := (n)$, $F_n^*(t) := \max\{0, p_n - t + t_0\}$;
2: **for** $k = n-1, n-2, \ldots, 1$ **do**
3: $\quad \pi^1 := (k, \pi_{k+1}^*(t - p_k))$; $\pi^2 := (\pi_{k+1}^*(t), k)$;
4: $\quad F(\pi^1) := \max\{0, p_k - d_k(t)\} + F_{k+1}^*(t - p_k)$;

5: $F(\pi^2) := F^*_{k+1}(t) + \max\{0, \sum_{j=k}^{n} p_j - d_k(t)\}$;
6: $F^*_k(t) := \min\{F(\pi^1), F(\pi^2)\}$; $\pi^*_k(t) := \arg\min\{F(\pi^1), F(\pi^2)\}$;
7: **end for**
8: **return** the schedule $\pi^*_1(d_n)$ and its total tardiness value $F^*_1(d_n)$.

Notice that lines 1 and $3-6$ of the algorithm have to be performed for each integer t from the interval $[t_0, t_0 + \sum_{j=1}^{n} p_j]$. From the above discussion, we obtain the following theorem.

Theorem 1 *Algorithm B-1 constructs an optimal schedule for case (2) in* $O(n \sum p_j)$ *time.*

We consider function $F^*_k(t)$ which has the following properties:

- it is continuous and monotonously non-increasing;
- it is a piecewise linear function;
- inequality
$$F^*_k(t-\epsilon) - F^*_k(t) \leq n\epsilon$$
holds for any $t \in \mathbb{R}$ and $\epsilon > 0$;
- there are no more than 2^{n-k} break points.

From lines $4-6$ of Algorithm B-1 we can see that $F^*_k(t)$ results from the two functions $F(\pi^1)$ and $F(\pi^2)$. We can analytically find (and store) the break points of function $F^*_k(t)$. Thus, we do not need the integer conditions for the processing times. The main idea of Algorithm B-1-*modified* is to find and store the break points of function $F^*_k(t)$ in each of the n iterations.

4.3 Algorithm B-$\mathbb{k}$

Assume that the following conditions hold:

$$\begin{cases} d_1 \leq d_2 \leq \ldots \leq d_n, \\ p_1 \geq p_2 \geq \ldots \geq p_n, \\ d_{\beta_1} - d_{\alpha_1} \leq p_{\beta_1}, & \alpha_1 = 1, \\ d_{\beta_2} - d_{\alpha_2} \leq p_{\beta_2}, & \alpha_2 = \beta_1 + 1, \\ \ldots \\ d_{\beta_{\mathbb{k}}} - d_{\alpha_{\mathbb{k}}} \leq p_{\beta_{\mathbb{k}}}, & \beta_{\mathbb{k}} = n. \end{cases} \tag{3}$$

In these inequalities, α_ν and β_ν (which are given by the partitioning procedure) are assigned to subset $\mathcal{M}_\nu = \{\alpha_\nu, \alpha_\nu + 1, \ldots, \beta_\nu\}, \nu = 1, 2, \ldots \Bbbk$. Due to Lemmas 1 and 2, for case (3), we can eliminate each schedule π from the consideration, for which the following condition holds: either there exists a job k such that $(i \to k \to j)_\pi$ for some $i, j \in \mathcal{M}_\nu$, where $\gamma(k) \leq \nu$ and $k < \min\{i, j\}$, or there exists a job k such that $(j \to k \to (k+1))_\pi$ for some $j > k$.

Algorithm B-$\Bbbk$ is an extended version of Algorithm B-1 for the case $\Bbbk > 1$. In contrast to Algorithm B-1, in each step we need to check more than two positions for the current job k in an optimal schedule for $I_k(t)$. Notice that the number of examined positions is less than or equal to $\Bbbk + 1$. To describe the structure of schedule $\pi_k^*(t)$, we use the following notation. Let $G_k(t)$ be an ordered set of the quadruples $\langle \pi_i, \nu_i, P_i, f_i \rangle$, $i = 1, \ldots, g$, $g = |G_k(t)| \leq \Bbbk$, where:

(1) π_i is a subschedule of $\pi_k^*(t)$ such that

   (a) $\pi_k^*(t) = (\pi_1, \pi_2, \ldots, \pi_g)$;

   (b) $k \in \{\pi_1\}$;

   (c) each subset $\mathcal{M}_\nu \subseteq N_k$ is contained only in one subschedule;

   (d) π_i cannot be partitioned into two subschedules such that items (a), (b), (c) hold for the new subschedules;

(2) $\nu_i = \min_{j \in \{\pi_i\}} \{\gamma(j)\}$;

(3) P_i is the total processing time of π_i, i.e. $P_i = \sum_{j \in \{\pi_i\}} p_j$;

(4) f_i is the total tardiness value of π_i.

To construct an optimal schedule, we need to know the positions for job k in schedule $\pi_{k+1}^*(t)$ such that k is not processed between two jobs from the same subset $\mathcal{M}_\nu$ and k is not processed between some job $j > k$ and job $k + 1$. This property of an optimal schedule can be maintained by analyzing schedule $\pi_k^*(t)$ in $O(n)$ time in each step of the algorithm. However, this can be done in $O(\Bbbk)$ time by collecting the information about $\pi_k^*(t)$ from the previous steps of the algorithm.

Let $\sum_q(X)$ denote the sum $\sum_{i=1}^q X_i$ for some indexed values X (we will also use the symbols P and f instead X). Let us now describe the construction of an optimal schedule $\pi_k^*(t)$ for $I_k(t)$. We use the sets $G_{k+1}(t)$

which contain information about the structure of schedule $\pi^*_{k+1}(t)$ at each point t. By Lemmas 1 and 2, the positions for k between the two jobs r, q can be eliminated if $r, q \in \{\pi_i\}$ for some $1 \le i \le g$, and $\pi_i \in \pi^*_{k+1}$. Therefore, we have only $g+1$ positions for job k: before all jobs from N_{k+1}, between each pair of subschedules π_{i-1} and π_i, and after all jobs from N_{k+1}. If the optimal position of job k is between π_{i-1} and π_i, then $\pi^*_k(t) := (\pi_1, \ldots, \pi_{i-1}, k, \pi^*_{\alpha_{\nu_i}}(t - \sum_{i-1}(P) - p_k))$. The schedule $\pi^*_{\alpha_{\nu_i}}(t - \sum_{i-1}(P) - p_k)$ is an optimal schedule for the set of jobs $\{\pi_i\} \bigcup \{\pi_{i+1}\} \bigcup \ldots \bigcup \{\pi_g\}$, since this set is equal to the set $\mathcal{M}_{\alpha_{\nu_i}} \bigcup \ldots \bigcup \mathcal{M}_{\alpha_{\Bbbk}}$, and an optimal schedule for this set has already been constructed. Then the set $G_k(t)$ is constructed in the following way. If k is inserted between π_{i-1} and π_i, then all jobs with smaller indices can be processed before job k only if they are processed before all jobs from N_{k+1}. This follows from Lemma 2. Therefore, we can join the jobs of the subschedules $\pi_1, \ldots, \pi_{i-1}$ into a single subschedule of the set $G_k(t)$:

$$G_k(t) := \Big\{ \big\langle (\pi_1, \ldots, \pi_{i-1}, k), \gamma(k), \textstyle\sum_{i-1}(P) + p_k, \\ \textstyle\sum_{i-1}(f) + \max\{0, \sum_{i-1}(P) + p_k - d_k(t)\} \big\rangle \Big\} \bigcup G_{\alpha_{\nu_i}}(t - \textstyle\sum_{i-1}(P) - p_k).$$

Algorithm *B*-$\Bbbk$

1: $\pi_n(t) := (n)$, $F_n(t) := \max\{0, p_n + t_0 - t\}$, $G_n(t) = \{\langle \pi_n(t), \Bbbk, F_n(t), p_n \rangle\}$;
2: **for** $\nu = \Bbbk, \Bbbk - 1, \ldots, 1$ **do**
3: **for** $k = \beta_\nu, \beta_\nu - 1, \ldots, \alpha_\nu$, $k < n$, **do**
4: **for** $i = 1, 2, \ldots, g+1$ **do**
5: $\pi^i := (\pi_1 \ldots, \pi_{i-1}, k, \pi^*_{\alpha_{\nu_i}}(t - \sum_{i-1}(P) - p_k))$;
6: $F(\pi^i) := \sum_{i-1}(f) + \max\{0, \sum_{i-1}(P) + p_k - d_k(t)\} + \\ \qquad + F_{\alpha_{\nu_i}}(t - \sum_{i-1}(P) - p_k)$;
7: **end for**
8: $i^* := \arg\min_{i=1,\ldots,g+1}\{F(\pi^i)\}$; $\pi^*_k(t) := \pi^{i^*}$; $F_k(t) := F(\pi^{i^*})$;
9: $G_k(t) := \Big\{ \big\langle (\pi_1, \ldots, \pi_{i^*-1}, k), \nu, \sum_{i^*-1}(P) + p_k, \sum_{i^*-1}(f) + \\ \max\{0, \sum_{i^*-1}(P) + p_k - d_k(t)\} \big\rangle \Big\} \bigcup G_{\alpha_{\nu_{i^*}}}(t - \sum_{i^*-1}(P) - p_k)$.
10: **end for**
11: **end for**
12: **return** schedule $\pi^*_1(d_n)$ and its total tardiness value $F_1(d_n)$.

Notice that lines 1 and 4−9 of the algorithm have to be performed for each integer t from the interval $[t_0, t_0 + \sum_{j=1}^n p_j]$. From the previous discussion, the following theorem is obtained.

Theorem 2 *Algorithm B-𝕜 constructs an optimal schedule for case (3) in* $O(\Bbbk n \sum p_j)$ *time.*

4.4 Algorithm *B-n*

Let us now suppose that the following conditions are satisfied:

$$d_j - d_{j-1} > p_j,\ j = 2, 3, \ldots, n. \tag{4}$$

In this subsection, the processing times need not to be integer. An instance of case (1) belongs to this subcase if $\Bbbk = n$. The algorithm for this subcase is a modification of the decomposition algorithm introduced in Section 3 when job $j^* = \arg\max_{j \in N}\{d_j : p_j = \max_{i \in N} p_i\}$ need to be sequenced only on one position in an optimal schedule. We have the following result.

Lemma 3 *There exists an optimal schedule for case (4), where job* j^* *is processed on the first position of set* $L(N, t)$*.*

Sequence B-n (N, t)

1: $S_k := t + p_1 + p_2 + \ldots + p_k$, $k = 1, 2, \ldots, n$;
2: Find j^* and $L(N, t)$; $k^* := \arg\min\{k \in L(N, t)\}$;
3: $N' := \{1, \ldots, k^*\} \setminus \{j^*\}$; $t' := t$; $N'' := \{k^* + 1, \ldots, n\}$; $t'' := S_{k^*}$;
4: $\pi^* :=$ (**Sequence B-n (N', t')**, j^*, **Sequence B-n (N'', t'')**);
5: **return** π^*.

Algorithm *B-n*: π^* := Sequence B-n (N, t_0).

Thus, we obtain the following theorem.

Theorem 3 *Algorithm B-n constructs an optimal schedule for case (4) in* $O(n^2)$ *time.*

5 Algorithm *C*-1

In this section, we present another polynomially solvable special case of the total tardiness problem. Let us suppose that the following conditions are satisfied:

$$\begin{cases} d_1 \le d_2 \le \ldots \le d_n, \\ d_n - d_1 \le 1, \\ t_0 \in \mathbb{Z}. \end{cases} \tag{5}$$

Again, the processing times need not to be integer. In the following algorithmic description, we denote by π_{edd} the EDD sequence composed of the jobs from the set N' in each step of the algorithm.

Algorithm *C*-1

1: $S := t_0 + \sum_{j=1}^{n} p_j$, $N' := N$, $\pi^* := \emptyset$, $\pi_{edd} := (1, 2, \ldots, n)$;
2: **while** $N'' = \{j \in N' : S - p_j \leq z\} = \emptyset$ **and** $N' \neq \emptyset$ **do**
3: $\pi^* := (k, \pi^*)$, where $k := \arg\max_{j \in N'}\{d_j : p_j = \max_{i \in N'} p_i\}$;
4: $S := S - p_k$, $N' := N' \setminus \{k\}$, $\pi_{edd} := \pi_{edd} \setminus \{k\}$; **end while**
5: **if** N' contains only one job, i.e. $N' = \{j\}$, **then** $\pi^* := (j, \pi^*)$, **stop;**
6: **for all** $j \in N'$ such that $S - p_j \leq z + 1$ **do**
7: **for all** $i \in N' \setminus \{j\}$ **do** $\pi_{ij} := (\pi_{edd} \setminus \{i, j\}, i, j)$; **end for**; **end for**;
8: $\pi^* := (\pi, \pi^*)$, where $\pi := \arg\min_{i,j} F(\pi_{ij})$.

We obtain the following result.

Theorem 4 *Algorithm C-1 constructs an optimal schedule for case (5) in* $O(n^2)$ *time.*

6 Algorithm *B*-1 and the Even-Odd Partition Problem

In this section, we consider the even-odd partition problem. First, we introduce a modified even-odd partition problem which is used to prove that the single machine total tardiness problem with oppositely ordered processing times and due dates is NP-hard in the ordinary sense. Then we review some properties of an optimal schedule for the canonical scheduling problem considered before. Moreover, Property *B-1* is defined for a schedule π. It is proved that, if there exists an optimal schedule for an instance I which has this property, then Algorithm B-1 constructs an optimal schedule for this instance (even if the conditions (2) do not hold). At the end of this section, we show that Algorithm B-1 can be used to solve the even-odd partition problem by introducing an adequate modification, called Algorithm B-1-*canonical*.

We note that Du and Leung [1] defined first canonical instances for problem $1 \mid\mid \sum T_j$, and in [2, 3] other class of canonical instances have been defined. In these papers, two classes of *canonical schedules* have been introduced:

- a *DL canonical schedule* for $3n+1$ jobs: one job is in the "center" of the schedule and n groups of 3 jobs are sequenced before or after this job;

- an *LG canonical schedule* for $2n+1$ jobs: one job is in the "center" and $2n$ jobs are sequenced before and after this "center" job (n jobs before and n jobs after this job).

6.1 The Even-Odd Partition Problem and Canonical LG Instances

The even-odd partition (EOP) problem is as follows: Given a set of $2n$ positive integers $B = \{b_1, b_2, \ldots, b_{2n}\}$, $b_i \geq b_{i+1}$, $i = 1, 2, \ldots, 2n-1$. Is there a partition of B into two subsets B_1 and B_2 such that $\sum_{b_i \in B_1} b_i = \sum_{b_i \in B_2} b_i$ and such that for each $i, i = 1, \ldots, n$, subset B_1 (and hence, B_2 too) contains exactly one number of $\{b_{2i-1}, b_{2i}\}$? The EOP problem is a well-known NP-complete problem.

Let $\delta_i = b_{2i-1} - b_{2i},\ i = 1, \ldots, n,\ \delta = \sum_{i=1}^{n} \delta_i$. Now we construct a modified Even-Odd Partition Problem. There is given a set of integers $A = \{a_1, a_2, \ldots, a_{2n}\}$ with

$$\begin{cases} a_{2n} = M + b, \\ a_{2i} = a_{2i+2} + b,\ i = n-1, \ldots, 1, \\ a_{2i-1} = a_{2i} + \delta_i,\ i = n, \ldots, 1, \end{cases} \tag{6}$$

where $b \gg n\delta$ (for example, $b = n^2\delta$) and $M \geq n^3 b$. Obviously, we have $a_i \geq a_{i+1}$ for all $i = 1, 2, \ldots, 2n-1$. Notice that $\delta_i = b_{2i-1} - b_{2i} = a_{2i-1} - a_{2i},\ i = 1, \ldots, n$. The modified problem is equivalent to the original one.

Lemma 4 *[2] The original EOP problem has a solution if and only if the modified EOP problem does.*

By means of the above instance of the EOP, we define a *canonical LG instance* of the total tardiness problem as follows. We have $2n+1$ V-jobs $V_1, V_2, V_3, V_4, \ldots, V_{2i-1}, V_{2i}, \ldots, V_{2n-1}, V_{2n}, V_{2n+1}$, renumbered as $N = \{1, 2, \ldots, 2n, 2n+1\}$ and satisfying the following conditions:

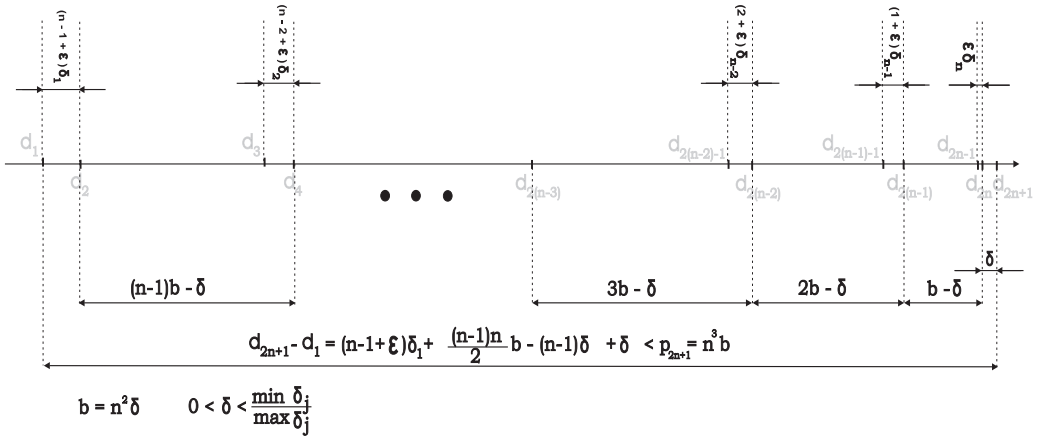

Figure 1: Due date pattern of the canonical LG instance.

$$
\begin{cases}
p_1 > p_2 > \ldots > p_{2n+1}, \\
d_1 < d_2 < \ldots < d_{2n+1}, \\
d_{2n+1} - d_1 < p_{2n+1}, \\
p_{2n+1} = M = n^3 b, \\
p_{2n} = p_{2n+1} + b = a_{2n}, \\
p_{2i} = p_{2i+2} + b = a_{2i}, \ i = n-1, \ldots, 1, \\
p_{2i-1} = p_{2i} + \delta_i = a_{2i-1}, \ i = n, \ldots, 1, \\
d_{2n+1} = \sum_{i=1}^{n} p_{2i} + p_{2n+1} + \frac{1}{2}\delta, \\
d_{2n} = d_{2n+1} - \delta, \\
d_{2i} = d_{2i+2} - (n-i)b + \delta, \ i = n-1, \ldots, 1, \\
d_{2i-1} = d_{2i} - (n-i)\delta_i - \varepsilon\delta_i, \ i = n, \ldots, 1,
\end{cases} \tag{7}
$$

where $b = n^2\delta$, $0 < \varepsilon < \frac{\min_i \delta_i}{\max_i \delta_i}$. The due date pattern of the canonical LG instance with oppositely ordered processing times and due dates is presented in Figure 1.

Let $L = \frac{1}{2}\sum_{i=1}^{2n} p_i$, then we have $d_{2n+1} = L + p_{2n+1}$ since $\frac{1}{2}\sum_{i=1}^{2n} p_i = \sum_{i=1}^{n} p_{2i} + \frac{1}{2}\delta$. Notice that the *canonical DL instances* from paper [1] do not correspond to case (7). From the above discussion, we have obtained the following result.

Theorem 5 *The single machine total tardiness problem with oppositely ordered processing times and due dates is NP-hard in the ordinary sense.*

Next, we summarize two well-known properties from the literature.

Theorem 6 *[2] For case (7), all optimal schedules are canonical LG schedules.*

Theorem 7 *[2] The modified EOP problem has a solution if and only if in an optimal canonical LG schedule, we have* $C_{2n+1}(\pi) = d_{2n+1}$.

If $p_j \in \mathbb{Z}^+,\ j \in N$, then the exact algorithm *B-1* does not only solve case (2), but also the canonical DL instances [1] in $O(n \sum p_j)$ time. If $p_j \notin \mathbb{Z}^+,\ j \in N$, then Algorithm *B*-1-*modified* can also solve the canonical DL instances. So, we can also find a solution for the non-integer even-odd partition and partition problems.

6.2 Property *B-1*

We say that a schedule π has *Property B-1*, if for each job $k \in N$: either $(k \rightarrow j)$ holds for all jobs $j \in \{k+1, \ldots, n\}$, or $(j \rightarrow k)$ holds for all jobs $j \in \{k+1, \ldots, n\}$.

As a consequence, a schedule π does not have Property *B-1*, if there exists a triple of jobs $i, j, k \in N$ such that $(i \rightarrow k \rightarrow j)_\pi$ and $k < \min\{i, j\}$, i.e. some job is processed in π between two jobs with greater numbers.

Let $\Pi_{B-1}(I)$ be the set of schedules for an instance I which have Property *B-1*, and $\Pi^*_{B-1}(I) = \Pi_{B-1}(I) \bigcap \Pi^*(I)$. For a schedule π, let $\chi_j(\pi) = 0$ if $j = n$ or there exists a job $i \in \{j+1, \ldots, n\}$ such that $(i \rightarrow j)_\pi$, and $\chi_j(\pi) = 1$ otherwise. For a schedule $\pi \in \Pi_{B-1}(I)$, we have $\chi_j(\pi) = 0$ if the job j follows after all jobs from the set $\{j+1, \ldots, n\}$ in π and $\chi_j(\pi) = 1$ if a job j precedes all jobs from this set in π. Therefore, we get the following result.

Lemma 5 *If* $\Pi^*_{B-1}(I) \neq \emptyset$*, then Algorithm* B-1 *constructs an optimal schedule for* I.

Hence, if we a priori know that there exists an optimal schedule which has Property *B-1* for some instance I, then Algorithm *B*-1 finds an optimal schedule for I. As a practical matter, Lemma 5 has no importance since, to solve an instance I by Algorithm *B*-1, we need to know whether there exists some optimal schedule with Property *B-1* or not. However, Lemma 5 allows us to use Algorithm *B*-1 to solve the even-odd partition problem since Du and Leung [1] have shown that an instance I corresponding to an instance of the even-odd partition problem has an optimal schedule with Property *B-1*.

6.3 Modification of Algorithm B-1 for the Even-Odd Partition Problem

Since a canonical schedule has Property *B-1*, it follows that Algorithm B-1 finds a solution for the even-odd partition problem. We construct a scheduling problem with $3n+1$ jobs: W_{n+1} and n triples of jobs $V_{2i-1}, V_{2i}, W_i, 1 \leq i \leq n$. We observe that each triple of jobs can be processed only in two ways $(V_{2i-1} \rightarrow W_i \rightarrow V_{2i})_{\pi^*}$ and $(V_{2i} \rightarrow W_i \rightarrow V_{2i-1})_{\pi^*}$ in an optimal canonical schedule π^*. We can use the following modification of Algorithm B-1 for the canonical instances.

Algorithm *B*-1-*canonical*

1: $\pi_n(t) := (W_{n+1})$, $F_n(t) := \max\{0, p_{W_{n+1}} - t\}$;
2: **for** $k = n-1, n-2, \ldots, 1$ **do**
3: $\quad \pi^1 := (V_{2k-1}, W_k, \pi_{k+1}(t - a_{2k-1} - b), V_{2k})$;
4: $\quad \pi^2 := (V_{2k}, W_k, \pi_{k+1}(t - a_{2k} - p_{W_k}), V_{2k+1})$;
5: $\quad F(\pi^1) := \max\{0, a_{2k-1} - d_{V_{2k-1}}(t)\} + \max\{0, a_{2k-1} + b - d_{W_k}(t)\} +$
$\quad F_{k+1}(t - a_{2k-1} - b) + \max\{0, \sum_{j=k}^{n}(a_{2j-1} + a_{2j} + b) - d_{V_{2k}}(t)\}$;
6: $\quad F(\pi^2) := \max\{0, a_{2k} - d_{V_{2k}}(t)\} + \max\{0, a_{2k} + b - d_{W_k}(t)\} +$
$\quad F_{k+1}(t - a_{2k} - b) + \max\{0, \sum_{j=k}^{n}(a_{2j-1} + a_{2j} + b) - d_{V_{2k-1}}(t)\}$;
7: $\quad F_k(t) := \min\{F(\pi^1), F(\pi^2)\}$; $\pi_k(t) := \arg\min\{F(\pi^1), F(\pi^2)\}$;
8: **end for**
9: **return** schedule $\pi_1(d_n)$ and its total tardiness value $F_1(d_n)$.

Thus, we can present the following theorem.

Theorem 8 *Algorithm B-1-canonical constructs an optimal canonical schedule (i.e. it solves instances of the even-odd partition problem) in $O(n\delta)$ time where $\delta = \frac{1}{2}\sum_{i=1}^{n}(a_{2i-1} - a_{2i})$.*

For the canonical instances, we have $p_{min} = p_{2n+1} = n^3 b = n^5\delta$, so $\delta = \frac{p_{min}}{n^5}$, and the complexity of Algorithm B-1-*canonical* is $O(n\delta) = O(\frac{p_{min}}{n^4})$.

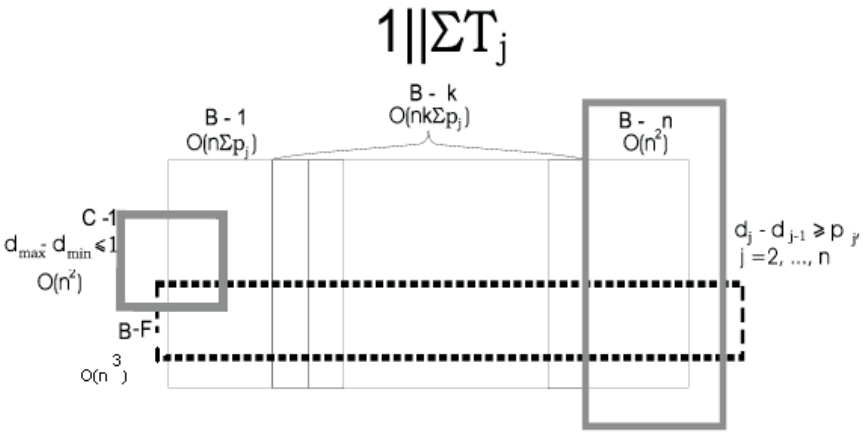

Figure 2: The subcases of the problem.

7 Concluding Remarks

In this paper, we mainly considered the total tardiness problem with oppositely ordered processing times and due dates. We presented pseudo-polynomial and polynomial algorithms for several cases. A simplified representation of the subcases considered is shown in Figure 2.

An overview on the results obtained is given in Table 1. If in the line B-1-*modified* additionally the integer condition $(***)$ is considered, the complexity of Algorithm B-1 is $O(n \sum p_j)$.

Moreover, there is a parallel paper [20] where we deal with the application of Property *B-1* to specific combinatorial problems, namely to the knapsack and partition problems.

Algorithm	Conditions	Complexity
B-$\Bbbk$, $\Bbbk < n$	$p_1 \geq p_2 \geq \ldots \geq p_n \quad (*)$ $d_1 \leq d_2 \leq \ldots \leq d_n \quad (**)$ $p_j \in \mathbb{Z}^+ \qquad (***)$	$O(\Bbbk n \sum p_j)$
B-1-*modified*	$(*), (**)$, $d_n - d_1 \leq p_n$	
B-1-*canonical*	$(*), (**), (***)$ $d_{\max} - d_{\min} \leq p_{\min}$	$O(\frac{p_{\min}}{n^4})$
B-F	$(*), (**)$ fixed number of tardy jobs	$O(n^3)$
B-n	$d_j - d_{j-1} > p_j, \ j = 2, 3, \ldots, n$	$O(n^2)$
C-1	$d_{\max} - d_{\min} \leq 1$	$O(n^2)$

Table 1: Special Cases and Algorithms

Acknowledgements

The authors are grateful to Drs Alexander Kvaratskhelia, Eugene Gafarov, Ruslan Sadykov, Andrei Tchernykh and Professors Peter Brucker, Wlodzimierz Szwarc and Edwin Cheng for a useful discussion of results and constructive comments.

This work partially supported by DAAD (Deutscher Akademischer Austauschdienst: A/08/08679, Ref. 325) and by Russian Funding Support of Scientific School (N-5833.2006.1).

References

[1] J. Du and J. Y.-T. Leung, Minimizing total tardiness on one processor is NP-hard. *Math. Oper. Res.* **15** 483 - 495 (1990).

[2] E.R. Gafarov and A.A. Lazarev, A special case of the single-machine total tardiness problem is NP-Hard. *Journal of Computer and Systems Sciences International*. **45** (3) 450 - 458 (2006).

[3] A.A. Lazarev and E.R. Gafarov, *Theory of scheduling. Minimizing total tardiness for a single machine*, Dorodnicyn Computing Centre of the Russian Academy of Sciences, Moscow (in Russian) (2006).

[4] H. Emmons, One machine sequencing to minimizing certain function of job tardiness. *Oper. Res.* **17** 701-715 (1969).

[5] E.L. Lawler, A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness, *Ann. Discrete Math.* **1** 331-342 (1977).

[6] E.L. Lawler, A fully polynomial approximation scheme for the total tardiness problem. *Oper. Res. Lett.* **1** 207-208 (1982).

[7] C.N. Potts and L.N. van Wassenhove, A decomposition algorithm for the single machine total tardiness problem. *Oper. Res. Lett.* **5** 177-182 (1982).

[8] A.A. Lazarev, Decomposition based algorithm for the total tardiness problem on a single machine. *Investigations on Applied Math.* **17** 71-78 (in Russian) (1990).

[9] S. Chang, Q. Lu, G. Tang, and W. Yu, On decomposition of the total tardiness problem. *Oper. Res. Lett.* **17** 221-229 (1995).

[10] W. Szwarc, *Single machine total tardiness problem revised.* In *Creative and Innovate Approaches to the Science of Management, Quorum Books,* pp. 407-419 (1993).

[11] W. Szwarc and S. Mikhopadhyay, Decomposition of the single machine total tardiness problem. *Oper. Res. Lett.* **19** 243-250 (1996).

[12] W. Szwarc, F. Della Croce and A. Grosso, Solution of the single machine total tardiness problem. *J. Sched.* **2** 55-71 (1999).

[13] W. Szwarc, A. Grosso and F. Della Croce, Algorithmic paradoxes of the single machine total tardiness problem. *J. Sched.* **4** 93-104 (2001).

[14] P. Brucker, J. Hurink and F. Werner, Improving local search heuristics for some scheduling problems - I. *Discrete Appl. Math.* **65** 97-122 (1996).

[15] A. Grosso, F. Della Croce and R. Tadei, An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem. *Oper. Res. Lett.* **32** 68-72 (2004).

[16] P.M. Franca, A. Mendes and P. Moscato, A memetic algorithm for the total tardiness single machine scheduling problem. *European J. Oper. Res.* **132** 224-242 (2001).

[17] C. Koulamas, The total tardiness problem: review and extensions. *Oper. Res.* **42** 1025-1041 (1994).

[18] T. Sen, J.M. Sulek and P. Dileepan, Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey. *Int. J. Production Economics.* **83** 1-12 (2003).

[19] A.A. Lazarev, A.G. Kvaratskhelia and E.R. Gafarov, Algorithms for solving the NP-hard problem of minimizing total tardiness for a single machine. *Doklady Mathematics.* **75** (1) 130-133 (2007).

[20] A.A. Lazarev and F. Werner, A graphical approach for solving NP-hard combinatorial problems. *Working Paper, Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg* (2008).