Draft of paper

Properties of the $1 || \sum T_j$ problem

A.A. Lazarev, E.R. Gafarov

Alexandr.Lazarev@ksu.ru,

http:\\ orsot.narod.ru

Abstract

In this paper we show that the run time of well known algorithms [3, 7, 8] for the problem $1 || \sum T_j$ the run time is more than $O(n2^{(n-1)/3-1})$ for canonical DL instances and great or equal $O(n2^{(n-1)/2})$ for the special case **BF**. For this cases we have constructed two new algorithms.

1 Introduction

Given a set N of n independent jobs that must be processed on a single machine. Preemptions of jobs are not allowed. The single machine can handle only one job at a time. The jobs are available for processing at time 0. For a job j, $j \in N$, a processing time $p_j > 0$ and a due date d_j are given. A schedule π is uniquely determined by a permutation of elements of N. We need to construct an optimal schedule π^* that minimizes the total tardiness value $F(\pi) = \sum_{j=1}^{n} \max\{0, C_j(\pi) - d_j\}$, where $C_j(\pi)$ is the completion time of job j in schedule π. $T_j(\pi) = \max\{0, C_j(\pi) - d_j\}$ is the tardiness of job j in schedule π. The problem $1 || \sum T_j$ is NP-hard in the ordinary sense [1]. A pseudo-polynomial time $O(n^4 \sum p_j)$ dynamic programming algorithm has been proposed by Lawler [2]. The state-of-the-art algorithms of Szwarc et al.[3, 4] handle special instances [5] of the problem for $n \leq 500$.

In well known algorithms [3, 4, 7, 8] following rules of elimination are used: **Elimination Rules 1-4**, **the calculation of parameters** E_j, L_j**, the construction of the modified instance**. We will show that this algorithms have the exponential run time for special cases $1||\sum T_j$ problem.

The paper is organized as following. Section 2 presents some basic properties, definitions and Algorithm A based on elimination rules 1-3.

In section 3 we investigate the run time of algorithms for *canonical DL instances.* We present an alternative algorithm $O(n\delta)$ time.

The special case BF is considered in section 4. For this case we have algorithm $O(n^2)$ time.

Algorithm **B-1 modified** has decided instances when $p_j \notin Z^+$. In section 5 we investigate its complexity time (the number of change of schedule).

2 Elimination rules.

The set N of jobs is considered to be initially ordered $d_1 \le d_2 \le \ldots \le d_n$, 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,
$j^* = \arg\max_{j\in N}\{d_j : p_j = \max_{i\in N} p_i\}$.

We consider a subset of jobs $N' \subseteq N$, let $N' = \{1, 2, \ldots, n'\}$ that must be processed from time $t' \ge t_0$.

Let define the set $L(N', t')$ of all indexes $k \ge j^*$ such that:

(a) $t' + \sum_{j=1}^{k} p_j < d_{k+1}$ (**Elimination Rule 1** [4, 7]) and

(b) $d_j + p_j \le t' + \sum_{j=1}^{k} p_j$, for all $j = \overline{j^*(N') + 1, k}$ (**Elimination Rules 2,3** [4, 7]).

where $d_{n'+1} = +\infty$.

We'll denote $< \{p_j, d_j\}_{j\in N}, t >$ the instance of the problem $1||\sum T_j$ for jobs of set N with parameters $\{p_j, d_j\}_{j\in N}$ from start time t.

Proposition 1 [7] *For all instances $\langle\{p_j, d_j\}_{j\in N}, t_0\rangle$ the set $L(N, t_0)$ isn't empty.*

Proposition 2 [2, 5, 7] *For all instances $\langle\{p_j, d_j\}_{j\in N}, t_0\rangle$ there exist the optimal schedule π^* such that $(j \to j^*)_{\pi^*}$ for all $j \in \{1, 2, \ldots, k\} \setminus \{j^*\}$ and $(j^* \to j)_{\pi^*}$ for all $j \in \{k + 1, \ldots, n\}$ for some $k \in L(N, t_0)$.*

We now describe an algorithm which based on Elimination Rules 1-3.

Procedure ProcL (N, t)

0. There exist the instance $< \{p_j, d_j\}_{j\in N}, t >$ with set of jobs $N = \{j_1, j_2, \ldots, j_n\}$ and start time t, $d_{j_1} \le d_{j_2} \le \ldots \le d_{j_n}$;

1. **IF** $N = \emptyset$ **THEN** π^*:= empty schedule, **GOTO 6.**;

2. Let find $j^* \in N$;

3. We construct the set $L(N,t)$ for job j^*;

4. **FOR ALL** $k \in L(N,t)$ **DO:**

$\pi_k := (\mathbf{ProcL}(N',t'), j^*, \mathbf{ProcL}(N'',t''))$, where $N' := \{j_1,\ldots,j_k\} \setminus \{j^*\}$, $t' := t$, $N'' := \{j_{k+1},\ldots,j_n\}$, $t'' := t + \sum_{i:=1}^{k} p_{j_i}$;

5. $\pi^* := \arg \min_{k \in L(N,t)} \{F(\pi_k,t)\}$;

6. **RETURN** π^*.

Algorithm A

$\pi^* := \mathbf{ProcL}(N,t_0)$.

We realized two versions of algorithm A: in "depth" and in "front".
Let $F(j^*,k)$ be the total tardiness of the modified EDD sequence where job j^* is moved from original position j^* to position k.

Proposition 3 (Elimination Rule 4). [8, 3] *Delete the position* k *from list* $L(N',t')$ *if* $|L(N',t')| > 1$ *and (*$F(j^*,k) > F(j^*,k+1)$ *or* $F(j^*,k) \geq F(j^*,i)$ *for some* $j^* \leq i < k$*).*

Let B_j be the set of jobs that precede job j, and A_j – the set of job that follow job j in an optimal sequence. The sets B_j and A_j may be empty at the same time.

Define $E_j = t' + P(B_j) + p_j$, $L_j = t' + P(N' \setminus A_j)$ as the earliest and latest completion time of job j in this sequence where $P(N') = \sum_{j \in N'} p_j$.

Proposition 4 (Emmons conditions). [9] *There exist an optimal sequence* π^* *where*

1. *i precedes j,* $(i \to j)_{\pi^*}$*, if* $d_i \leq \max(E_j, d_j)$ *and* $p_j \geq p_i$*;*
2. *j precedes i,* $(j \to i)_{\pi^*}$*, if* $d_i + p_i \geq L_j$ *and* $d_i > \max(E_j, d_j)$*,* $p_j \geq p_i$*.*

Proposition 5 [2] *Let $C_j = C_j(\pi^*)$ be the completion time of job j in an optimal sequence π^*. If*

$$\min\{d_j, C_j\} \leq d'_j \leq \max\{d_j, C_j\},$$

then an optimal sequence π' for modified instance $< \{p_j, d'_j\}_{j\in N}, t >$ with due dates $d'_1, d'_2, \ldots, d'_n$ is optimal for original instance $< \{p_j, d_j\}_{j\in N}, t >$ with due dates $d_1, d_2, \ldots, d_n$.

Offer to search solution for modified instance where $p'_j = p_j,\ d'_j = \max\{E_j, d_j\}$ [3].

3 Canonical instances.

In this section we will describe two NP-hard cases of the problem $1||\sum T_j$ – canonical instances DL[1] and LG. For canonical DL instances we research the run time of algorithm A. NP-hardness of canonical instances are showed by reduction from NP- complete *Even-Odd Partition problem* (EOP):

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 $i = 1, \ldots, n$ B_1 (and hence, B_2) contains exactly one number of $\{b_{2i-1}, b_{2i}\}$?

3.1 Canonical LG instances $1||\sum T_j$.

Now we construct the modified Even-Odd Partition Problem (MEOP). There is the following set of integers $A = \{a_1, a_2, \ldots, a_{2n}\}$. Let $\delta_i = b_{2i-1} - b_{2i},\ i = 1, \ldots, n$.

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

where $b \gg 2n\delta,\ M \geq n^3 b,\ \delta = \frac{1}{2}\sum_{i=1}^{n}(b_{2i-1} - b_{2i})$.

Now we present the polynomial reduction from modified **EOP** problem to special subcase **B-1**[7] of the problem $\mathbf{1 \mid\mid \sum \boldsymbol{T_j}}$.

$$
\left\{
\begin{array}{lr}
p_1 > p_2 > \ldots > p_{2n+1}, & (2.1) \\
d_1 < d_2 < \ldots < d_{2n+1}, & (2.2) \\
d_{2n+1} - d_1 < p_{2n+1}, & (2.3) \\
p_{2n+1} = M = n^3 b, & (2.4) \\
p_{2n} = p_{2n+1} + b = a_{2n}, & (2.5) \\
p_{2i} = p_{2i+2} + b = a_{2i},\ i = n-1, \ldots, 1, & (2.6) \\
p_{2i-1} = p_{2i} + \delta_i = a_{2i-1},\ i = n, \ldots, 1, & (2.7) \\
d_{2n+1} = \sum_{i:=1}^{n} p_{2i} + p_{2n+1} + \delta, & (2.8) \\
d_{2n} = d_{2n+1} - 2\delta, & (2.9) \\
d_{2i} = d_{2i+2} - (n-i)b + 2\delta,\ i = n-1, \ldots, 1, & (2.10) \\
d_{2i-1} = d_{2i} - (n-i)\delta_i - \varepsilon\delta_i,\ i = n, \ldots, 1, & (2.11)
\end{array}
\right. \qquad (2)
$$

where $b = 2n^2\delta$, $0 < \varepsilon < \frac{\min_i \delta_i}{\max_i \delta_i}$.

3.2 Canonical DL [1] instances $1||\sum T_j$.

Now we present the other polynomial reduction from modified **EOP** problem to special subcase of the problem $\mathbf{1} \mid\mid \sum \boldsymbol{T_j}$ [1].

Let $a_{2i-1} = b_{2i-1} + (9n^2 + 3n - i + 1)\delta + 5n(b_1 - b_{2n})$ and $a_{2i} = b_{2i} + (9n^2 + 3n - i + 1)\delta + 5n(b_1 - b_{2n})$, $i = 1, \ldots, n$.

We construct the *canonical DL instance* [1] of the problem $\mathbf{1} \mid\mid \sum \boldsymbol{T_j}$ for set of jobs $N = \{V_1, V_2 \ldots, V_{2n}, W_1, W_2, \ldots, W_{n+1}\}$. $|N| = 3n + 1$. Let $b = (4n + 1)\delta$. Define due dates and processing times as follows:

$$p_{V_i} = a_i, \qquad i = 1, 2, \ldots, 2n,$$

$$p_{W_i} = b, \qquad i = 1, 2, \ldots, n+1,$$

$$d_{V_i} = \left\{ \begin{array}{ll} (j-1)b + \delta + (a_2 + a_4 + \ldots + a_{2i}) & i = 2j - 1, \\ d_{V_{2j-1}} + 2(n - j + 1)(a_{2j-1} - a_{2j}) & i = 2j, j = 1, 2, \ldots, n; \end{array} \right.$$

$$d_{W_i} = \left\{ \begin{array}{ll} ib + (a_2 + a_4 + \ldots + a_{2i}) & i = 1, 2, \ldots, n, \\ d_{W_n} + \delta + b & i = n + 1. \end{array} \right.$$

Let $\{V_{i,1}, V_{i,2}\} = \{V_{2i-1}, V_{2i}\}$, $i = 1, \ldots, n$. Define *the canonical DL schedule* as follows

$$\pi = (V_{1,1}, W_1, V_{2,1}, W_2, \ldots, W_{n-1}, V_{n,1}, W_n, W_{n+1}, V_{n,2}, V_{n-1,2}, \ldots, V_{1,2}).$$

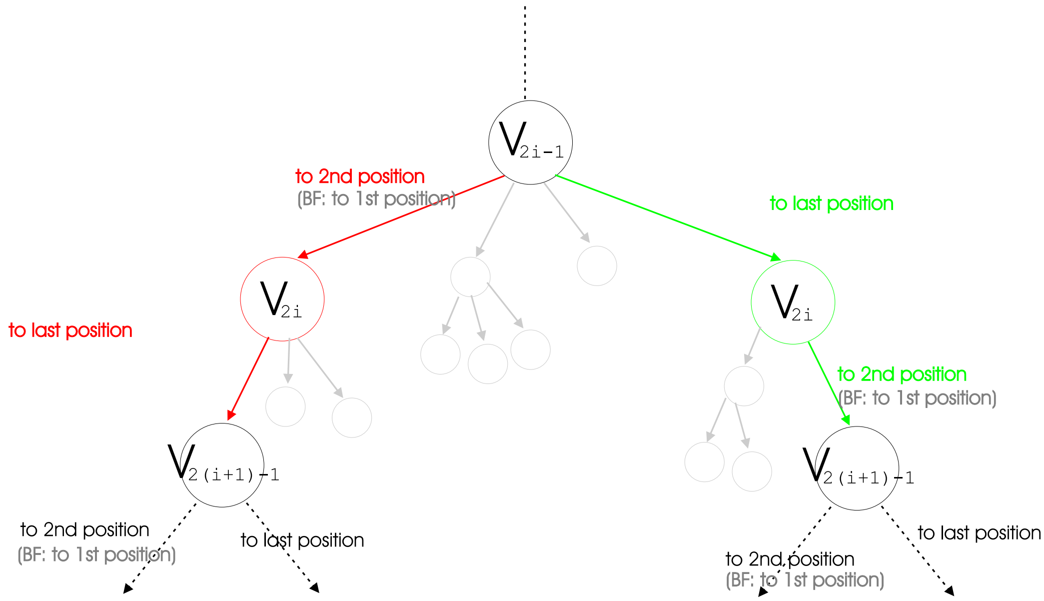

Figure 1: Search tree.

Proposition 6 [1] *For canonical DL instances there exist always an optimal schedule that is a canonical DL schedule.*

Next we show that algorithms based only Elimination Rules 1-4, the calculation of parameters $E_j,\ L_j$, the construction of the modified instance have exponential run time for canonical DL instances.

In Fig.1 there shows the search tree of Algorithm A for canonical DL instances.

Definition *Let canonical DL instances, where*

$$\delta - \sum_{j:=1}^{i-1} \delta_j \geq \delta_i,\ \ 2 \leq i \leq (n-1),$$

be SD (shortly delta) instances.

For the case *SD*

$$\delta_i > \frac{\sum_{j:=1}^{i-1} \delta_j - \delta}{2(n-i+1)},\ \ \ 2 \leq i \leq (n-1),$$

holds, because $\delta - \sum_{j:=1}^{i-1} \delta_j \geq \delta_i > 0$, so $\sum_{j:=1}^{i-1} \delta_j - \delta < 0$.

For example, if

$$\delta_i > 2 \sum_{j:=1}^{i-1} \delta_j, \quad 2 \leq i \leq n,$$

there is the case *SD*.

We consider a set of jobs N where jobs have EDD (early due date) order: $(V_1, V_2, W_1, \ldots, V_{2i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1})$.

Definition. *The skeleton of a tree that contains "double branching" (Fig.1) are called "basis tree".*

Proposition 7 *The search tree contains "double branching" (Fig.1) when Elimination Rules 1–3 are used. There exist a branching when we select a position for each job V_{2i-1}. For job V_{2i} an opposite position is approaching, $i = 1, 2, \ldots, n$.*

Proposition 8 *For the case* SD *Elimination Rule 4 doesn't reduce "double branching" when we select a position for each job V_{2i-1}, $i = 1, 2, \ldots, n$.*

Proposition 9 *Elimination Rule 4 deletes other positions from the position list without the 2-nd and the last current positions.*

That's why Algorithm constructs only canonical DL schedules when Elimination Rule 4 are used.

Proposition 10 *When we use parameters E_j, L_j "double branching" isn't reduced.*

Proposition 11 *Numbers b_i don't influence the position list, only numbers δ_i, $i = 1, 2, \ldots, n$, do.*

3.3 The run time of well known algorithms for the case SD.

Proposition 12 *For the case* SD *algorithms that use only rules:* **Elimination Rules 1-4**, **the calculation of parameters E_j, L_j, the construction of the modified instance** *have the run time more then $O(n2^{(n-1)/3-1})$.*

If a canonical instance doesn't correspondent the case SD then the "basis tree" is not complete. While $\bar{\delta} < \delta + 2(n-i+1)(a_{2i-1} - a_{2i})$ the "double branching" holds.

If $a_{2k-1} - a_{2k} \approx a_{2l-1} - a_{2l},\ \forall k, l = 1, \dots, n$, then the "double branching" holds for $i := 1, \dots, n/2$.

If $\delta_1 \geq \dots \geq \delta_n$ then the run time is smallest.

3.4 Solution algorithms for canonical instances.

If $\delta \notin Z$ let's consider the modified instance where b_i are multiplied to 2. The modified instance are equivalent to the original one.

Define $d_j(t) = d_j - d_{W_{n+1}} + t,\ j \in N$. Let $\pi_l(t)$ and $F_l(t)$ are an optimal schedule and its total tardiness for the instance with set of jobs $N_l = \{V_{2l-1}, V_{2l}, W_l, \dots, V_{2n-1}, V_{2n}, W_n, W_{n+1}\}$ and due dates $d_j(t),\ j = V_{2l-1}, V_{2l}, W_l, \dots, V_{2n-1}, V_{2n}, W_n, W_{n+1},\ l = n+1, \dots, 1$.

Algorithm B-1 canonical

0. $\pi_{n+1}(t) := (W_{n+1}),\ F_{n+1}(t) := \max\{0, b - t\}$
$t \in T_{n+1} := [d_{W_{n+1}} - \sum_{i=1}^{n} a_{2i} - nb - 2\delta, d_{W_{n+1}} - \sum_{i=1}^{n} a_{2i} - nb]$

1. **for** $l = n, n-1, \dots, 1$, for
$t \in T_l := [d_{W_{n+1}} - \sum_{i=1}^{l-1} a_{2i} - (l-1)b - (2\delta - \sum_{i=l-1}^{n} \delta_i), d_{W_{n+1}} - \sum_{i=1}^{l-1} a_{2i} - (l-1)b]$:

$\pi^1 := (V_{2l-1}, W_l, \pi_{l+1}(t - a_{2l-1} - b), V_{2l}),\ \pi^2 := (V_{2l}, W_l, \pi_{l+1}(t - a_{2l} - b), V_{2l-1})$;

$F(\pi^1) := \max\{0, a_{2l-1} - d_{V_{2l-1}}(t)\} + \max\{0, a_{2l-1} + b - d_{W_l}(t)\} + F_{l+1}(t - a_{2l-1} - b) + \max\{0, \sum\limits_{j=l}^{n}(a_{2j-1} + a_{2j} + b) + b - d_{V_{2l}}(t)\}$;

$F(\pi^2) := \max\{0, a_{2l} - d_{V_{2l}}(t)\} + \max\{0, a_{2l} + b - d_{W_l}(t)\} + F_{l+1}(t - a_{2l} - b) + \max\{0, \sum\limits_{j=l}^{n}(a_{2j-1} + a_{2j} + b) + b - d_{V_{2l-1}}(t)\}$;

$F_l(t) := \min\{F(\pi^1), F(\pi^2)\};\ \pi_l(t) := \arg\min\{F(\pi^1), F(\pi^2)\}$.

2. **return:** the optimal schedule $\pi_1(d_{W_{n+1}})$ and its value of the total tardiness $F_1(d_{W_{n+1}})$.

Notice that the step 1 of the algorithm are performed for each integer t from the interval length is 2δ.

Proposition 13 **Algorithm B-1 canonical** *constructs an optimal schedule for canonical Dl instances in $O(n\delta)$ time.*

There exist the exact **Algorithm B-1 modified**. In this algorithm only "points of change of schedule" are considered. Its run time depends on a number of that points.

We investigate the number of that points. Our results are presented in section 5.

4 Special case of the problem $1||\sum T_j$

The following case are considered:

$$\begin{cases} p_1 \geq p_2 \geq \ldots \geq p_n, \\ d_1 \leq d_2 \leq \ldots \leq d_n, \\ d_n - d_1 \leq p_n. \end{cases} \tag{3}$$

This case is called "hard" instances in the paper [6]. The research of known algorithms [3, 7, 8] has shown that for case **B-1** the number of branchings in the search tree is big [7].

Define the case BF as follows:

$$\begin{cases} p_1 \geq p_2 \geq \ldots \geq p_n, \\ d_1 \leq d_2 \leq \ldots \leq d_n, \\ d_n - d_1 \leq p_n, \\ n = 2k, \\ \sum_{i:=1}^{k} p_i < d_j < \sum_{i:=k}^{n} p_i, \ j = 1, 2, \ldots, n, \\ p_1 - p_n \ll p_n, \\ \sum_{i:=k+j+1}^{n} (p_{2j-1} - p_i) > d_{k+j} - d_{2j}, j = 1, ..., (k-1), \\ \sum_{i:=k+j+1}^{n} (p_{2j} - p_i) > d_{k+j} - d_{2j}, j = 1, ..., (k-1). \end{cases} \tag{4}$$

We denote the jobs as $(1, 2, \ldots, n) = (V_1, V_2, \ldots, V_{2j-1}, V_{2j}, \ldots, V_n)$.

Notice that for the case 4 in all $n!$ schedules only k job are tardy.

Proposition 14 *For the case (4) the search tree contains "double branching" (Fig.1) when Elimination Rules 1–4 are used. There exist a branching when we select a position for each job V_{2i-1}, $i = 1, \ldots, (k-1)$. For job V_{2i} an opposite position is approaching.*

Proposition 15 [7] *For all $l \in N$ in the case (3) there exist an optimal schedule $\pi^* = (\pi_1^*, \pi_l, \pi_2^*)$, where $\{\pi_l\} = N_l = \{l, \ldots, n\}$, $\{\pi_1^*, \pi_2^*\} = \{1, \ldots, l-1\}$.*

Proposition 16 *For the case (4) algorithms that use only rules:* **Elimination Rules 1-4**, **the calculation of parameters** E_j, L_j, **the construction of the modified instance** *have the run time great or equal then $O(n2^{n/2})$.*

For the case (3) we have pseudo-polynomial **Algorithm B-1** $O(n \sum p_j)$ and **Algorithm B-1 modified** for $p_j > 0$.

For the case (4) there exist exact **Algorithm BF** run time $O(n^2)$.

5 Computational results

The section describes the search for *the number of change of schedule* that will allow to investigate run time of **Algorithm B-1 modified** for canonical instances. The results are showed in the table after experiment 8.

We consider a set of EOP instances. For each EOP instance we construct the canonical DL instance of the problem $1||\sum T_j$. Then we use **Algorithm B-1 canonical** and count up *the number of points t of change of schedule* for each l.

Experiment 6. We consider all instances for $n = 2, 3, 4, 5$ when

$$\begin{cases} 200 \geq b_1 > b_2 \geq b_3 > b_4 \geq \ldots \geq b_{2n-1} > b_{2n} \geq 1, \\ b_i \in Z^+, \end{cases}$$

holds. That's why the number of instances is great then C_{200}^{2*n}. For $n = 5$ we have :

$$C_{200}^{10} = \frac{200!}{(200-10)!10!} = 22'451'004'309'013'280$$

instances.

The number of points is counted up in each step $l = n, n-1, \ldots, 1$ and then is summarized.

The following results are obtained:

n (number of pairs)	Max number of points
2	1
3	4
4	11
5	21

The follow property holds:

Property 1. *Two EOP instances* $\{(a_1, a_2), (a_3, a_4), \ldots, (a_{2j-1}, a_{2j}), \ldots, (a_{2n-1}, a_{2n})\}$ *and* $\{(a_1, a_2), (a_3, a_4), \ldots, (a_{2j-1} + \triangle, a_{2j} + \triangle), \ldots, (a_{2n-1}, a_{2n})\}$*, where* $\triangle \in Z^+$*,* $a_{2j-1} + \triangle \leq a_{2j-2}$*,* $a_{2j} + \triangle \leq a_{2j+1}$ *are identical. Appropriate canonical instances have equal* points of change of schedule *and identical optimal schedules.*

That's why we denote EOP instance as $(\delta_1, \delta_2, \ldots, \delta_n)$. We can reduce the number of considered instances.

Experiment 8. We consider all EOP instances $(\delta_1, \delta_2, \ldots, \delta_n)$ for $n = 2, 3, \ldots, 7$ when

$$\begin{cases} 50 \geq \sum_{i:=1}^{n} \delta_i, \\ \delta_i \in Z^+, \end{cases}$$

holds.

Denote CS_l – *the number of points of change of schedule* in the step $l = n, n-1, \ldots, 1$ of **Algorithm B-1 canonical**.

We've found "hard" EOP instances that $CS_n = 1, CS_{n-1} = 3,\ CS_{n-2} \geq 7,\ CS_{n-3} \geq 15,\ CS_{n-4} \geq 31, \ldots$

Following properties hold:

Property 2. Let S – a set of EOP instances: $\delta_1 = \sum_{i=2}^{n} \delta_i$. Let U – a set of "hard" instances. Then $S \bigcap U \neq \emptyset$.

Property 3. An instance from set S have "complexity" (1,3,7,15,31,0) (for $n = 6$). We assume that the complexity for some instance can be $(1, 3, 7, \ldots, a_i, 2a_i + 1, \ldots)$.

Property 4. If we'll delete the first pair from set B for "hard" instance then the modified instance may be not "hard".

Property 5. Let the instance $(\delta_1, \delta_2, \ldots, \delta_n)$ have the complexity (1,3,7,15,31) then an instance $(\delta_1 + k * 2, \delta_2, \ldots, \delta_n)$ have this complexity too.

Property 6. "Hard" instances can have the solution or not for original EOP.

Property 7. For some EOP instance we can construct the equivalent EOP instance where $\delta_1 \leq \delta_2 \leq \ldots \leq \delta_n$. But for the modified instance the run time of **Algorithm B-1 canonical** is lower and *the number of points of change of schedule* is less then for original one. For modified instances we have experiment 8.2. Notice, that the case *SD* correspondents to $\delta_1 \leq \delta_2 \leq \ldots \leq \delta_n$.

Computational results:

n	B-1	B-1 canonical $(CS_n, CS_{n-1}, \ldots, CS_1)$	
		$\delta_1, \delta_2, \ldots, \delta_n$	$\delta_1 \leq \delta_2 \leq \ldots \leq \delta_n$
2	1	(1,0)	(1,0)
3	3	(1,3,0)	(1,1,0)
4	8	(1,3,8,0)	(1,3,2,0)
5	15	(1,3,8,18,0)	(1,3,5,5,0)
6	23	(1,3,8,18,32,0)	(1,3,7,7,7,0)
7	38	(1,3,8,18,32,63,0)	(1,3,7,13,19,19,0)
8	44	-	-
9	51	-	-

6 Conclusion.

For cases *BF* and *SD* of the problem $1||\sum T_j$ algorithms that use only rules: **Elimination Rules 1-4**, **the calculation of parameters** E_j, L_j**, the construction of the modified instance** have the exponential run time $O(n2^{(n-1)/3-1})$ or $O(n2^{(n-1)/2})$.
It's difficult to believe that this algorithms will give the solution for $n \geq 100$ in this cases.
We'd better use other scheme of instances generation then scheme [5].
For cases *BF* and *SD* we have exact pseudo-polynomial and polynomial algorithms.

References

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

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

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

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

[5] C.N. Potts and L.N. Van Wassenhove (1982). *A decomposition algorithm for the single machine total tardiness problem*, Oper. Res. Lett., **1** , pp. 177–182.

[6] F. Della Croce, A. Grosso, V. Paschos (2004). *Lower bounds on the approximation ratios of leading heuristics for the single-machine total tardiness problem*, Journal of Scheduling, **7** , pp. 85–91

[7] A. Lazarev, A. Kvaratskhelia, A. Tchernykh (2004). *Solution algorithms for the total tardiness scheduling problem on a single machine*, Workshop Proceedings of the ENC'04 International Conference, pp. 474–480.

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

[9] H. Emmons (1969). *One machine sequencing to minimize certain functions of job tardiness*, Oper. Res., 17, pp. 701–715.