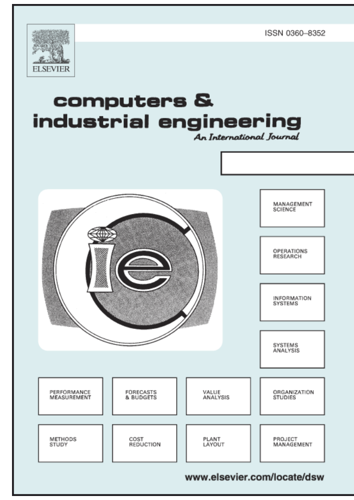

Two-Station Single-Track Railway Scheduling Problem With Trains of Equal Speed

Evgeny R. Gafarov, Alexandre Dolgui, Alexander A. Lazarev

PII: S0360-8352(15)00123-0
DOI: http://dx.doi.org/10.1016/j.cie.2015.03.014
Reference: CAIE 3989

To appear in: *Computers & Industrial Engineering*

Received Date: 16 July 2014
Revised Date: 28 January 2015
Accepted Date: 23 March 2015

Two-Station Single-Track Railway Scheduling Problem With Trains of Equal Speed

Abstract

In this paper, the single-track railway scheduling problem with two stations and several segments of the track is considered. Two subsets of trains are given, where trains from the first subset go from the first station to the second station, and trains from the second subset go in the opposite direction. The speed of trains over each segment is the same. A polynomial time reduction from the problem under consideration to a special case of the single-machine equal-processing-time scheduling problem with setup times is presented. Different polynomial time algorithms are developed for special cases with divers objective functions under various constraints. Moreover, several theoretical results which can be ranked in a series of similar investigations of NP-hardness of equal-processing-time single-machine scheduling problems without precedence relations are obtained.

Keywords: Single machine scheduling, Setup times, Transportation, Train scheduling, Computational complexity, Polynomial time algorithms
2000 MSC: 90B35

1. Introduction

A single-track network can be seen as an embryonic portion for any type of railway network topology. Furthermore, almost all national railway networks have sections where there is a single-track between some stations. For some countries (USA, Australia) a significant part of the network is single-track. For multi-track networks such a single-track segment can be considered as a bottleneck, in which the traffic capacity is restricted. Often, single-track problems are considered in the case of maintenance of one track of a double-track line. For example, the French railway company SNCF develops such models to produce a new transport schedule in the event of an incident on one of our double-track line sections [22].

For this problem, different optimization criteria and different constraints are considered:

- **optimization criteria**: initial scheduling to minimize total idle-time, rescheduling to minimize delays, etc.;

- **additional constraints**: overtaking segments, release and due dates, priority and velocity of trains, etc.

In this paper, we consider a case with only two stations. This case appears, for example, in private railways when a company transports loads between two production centers [14]. It represents also an elementary section of a larger railway network. There are segments on the track and only one train can travel on a segment at one time. Segments of a single-track and restriction that "at most only one train can be on any track segment at any the time" were already used in [25].

The Single Track Railway Scheduling Problem with two stations (STRSP2) is formulated as follows. Given a single-track railway between two stations and a set $N' = N'_1 \bigcup N'_2,\ N'_1 \bigcap N'_2 = \emptyset$ of $n' = |N'|$ trains. Trains from the subset N'_1 go from station 1 to station 2, and trains from the subset N'_2 go in the opposite direction. $|N'_1| = n'_1$ and $|N'_2| = n'_2,\ n'_1 + n'_2 = n'$. The track is divided into Q segments $1, 2, \dots, Q$. Trains from the set N'_1 traverse segments in an order $1 \to 2 \to \dots \to Q$ and trains from the set N'_2 in the opposite order $Q \to Q-1 \to \dots \to 1$. At most one train can be on any track segment at one time[1]. If a train $j' \in N'_1$ is on a track segment, then no train $i' \in N'_2$ can be on the track and vice versa. For each segment $q,\ q = 1, 2 \dots, Q$, a traversing time p_q is given, in which a train $j' \in N'$ traverses the segment, i.e., for each segment $q,\ q = 1, 2 \dots, Q$, all the trains go with the same speed[2].

All notations used hereafter are typical for scheduling theory. To represent tardy trains, Boolean variables U which are equal to 0 if a job is on-time or 1, otherwise, are used.

Let $S_{j'}(\Pi)$ and $C_{j'}(\Pi),\ j' \in N'$, be the start and completion times of the train j' in a schedule Π, i.e., $S_{j'}(\Pi)$ is a departure time of the train j' from

[1]A segment is circumscribed by two signals: one signal from each side, which will control whether a train can or cannot proceed on that segment. This is a safety precaution.

[2]This assumption is not far removed from practice, since most trains travel at a maximal authorized speed. Nevertheless, a safety precaution should be satisfied for unforseen situations, i.e. at most only one train can be on any track segment at one time.

the departure station and $C_{j'}(\Pi)$ is an arrival time at the destination station. Then in a feasible schedule we have:

- $C_{j'} \geq S_{j'} + \sum_{q=1}^{Q} p_q, \ \forall j' \in N'$;
- for any $i' \in N_1'$ and for any $j' \in N_2'$ we have $C_{i'} \leq S_{j'}$ or $C_{j'} \leq S_{i'}$.

In addition, a due date $d_{j'}$, a weight $w_{j'} > 0$, a release date $r_{j'} \geq 0$ (the earliest possible starting time, i.e., $S_{j'} \geq r_{j'}$) for each train $j' \in N'$ can be given. If $C_{j'}(\Pi) > d_{j'}$, then train j' is tardy and we have $U_{j'}(\Pi) = 1$. If $C_{j'}(\Pi) \leq d_{j'}$, then train j' is on-time and $U_{j'}(\Pi) = 0$. Moreover, let $T_{j'}(\Pi) = \max\{0, C_{j'}(\Pi) - d_{j'}\}$ be the tardiness of train j' and $C_{max}(\Pi) = \max_{j' \in N'}\{C_{j'}(\Pi)\}$ be the makespan for schedule Π. For the STRSP2 with release dates, the objective is to find an optimal schedule Π^* that minimizes the makespan C_{max} taking into account release dates. This problem is denoted $STRSP2|r_j|C_{max}$ by using the traditional three-field notation $\alpha|\beta|\gamma$ for scheduling problems proposed by Graham et al. [13], where α describes the resource environment, β gives the activity characteristics and further constraints and γ describes the objective function.

In this paper, we deal with some extensions of STRSP2 with different objective functions and further constraints. We minimize:

- number of late trains $STRSP2||\sum U_j$;
- weighted number of late trains $STRSP2||\sum w_j U_j$;
- total completion time $STRSP2|r_j|\sum C_j$ when release dates are given;
- weighted total completion time $STRSP2||\sum w_j C_j$;
- total tardiness $STRSP2||\sum T_j$.

A literature review on similar problems is given in Section 2. In Section 3, a polynomial time reduction of $STRSP2$ to a special case of the single-machine equal-processing-time scheduling problem with setup times is suggested. Then, polynomial time algorithms for the single-machine problems with above mentioned objective functions are presented in Sections 4 and 5.

This paper was motivated by some train scheduling problems on single path railways, but after the proof of reduction of these problems to equivalent single-machine scheduling problems, the contributions of this paper are on both train scheduling and theoretical single-machine scheduling issues.

2. Related literature

Work on single-track railway scheduling problems (STRSP) goes back to 1970's, with the publication [24]. A recent literature review on the single-track railway scheduling problem can be found, e.g., in [21]. A short survey on STRSP with several stations where trains are able to pass one other is presented in [25]. A STRSP with two types of trains – express and local – to minimize the period of a cyclic railway timetable is considered in [15].

Similar problems arise on a river canal (inland waterways) with a chain of shipping locks. If each shipping lock can operate only one ship at a time and the width of the canal is not enough to take more than one ship, then we will have STRSP2. A lockmaster's problem is considered in [6], where a single lock is given with a constant lockage time which can handle several ships from the same subset at a time. Arrival times of the ships are defined and the goal is to minimize total ship waiting time. In [6] a dynamic programming algorithm is proposed that solves this lockmaster's problem and some generalizations in $O(n^5)$ time. This lockmaster's problem can be presented as a single batching machine scheduling problem $1|p-batch; b=n; r_i; p_i=p|\sum w_j F_j$ [6]. Although this problem seems to be similar to STRSP2, there are obvious differences between the problems. In the lockmaster's problem there is a single segment (lock) which can process several trains (ships) at a time.

A similar problem which arises on German railways is considered in [5]; it deals with rescheduling trains in the case where one track of a railway section consisting of two tracks in opposing directions is closed due to construction activities and the sub-sequences of trains for both directions outside the construction site are fixed. In that NP-hard problem the maximal lateness should be minimized when single segment and safety distances between pairs of trains are given. Unlike that problem, in the problem considered in this paper st_1 and st_2 can be different.

Another similar problem of bidirectional scheduling is considered in [8]. Note that our paper was submitted to the journal before we found this new work. Thus, our results were obtained independently from [8]. The principal differences between the results presented in this paper and the results presented in [8] are as follows:

- different problem formulations – in [8], trains can wait in between segments, i.e. the condition "for any $i' \in N_1'$ and for any $j' \in N_2'$ we have $C_{i'} \le S_{j'}$ or $C_{j'} \le S_{i'}$", is not satisfied;

- different additional conditions – in [8], weights $w_{j'}$ and due dates $d_{j'}$ of jobs (trains) are not taken into account;

- different objective functions – unlike [8], we consider also objective functions $\sum w_j C_j$, $\sum w_j U_j$, and $\sum T_j$;

- different ideas on what the solution algorithms presented in both papers are based, e.g., in this paper we employ for our developments a set of possible starting times of trains, and so on.

Other surveys of optimization models for train routing and scheduling can be found in [7, 23].

To the best of our knowledge there are no other publications for the set of STRSP2 problems considered in this paper.

This paper is an extended, completed and developped version of our previous work (see our first preprint [10] and conference proceedings [11, 12]).

3. Reduction of STRSP2 to a Single Machine Scheduling Problem

The idea to reduce STRSP to standard scheduling problems is not new. If the number of segments $Q = 1$, then STRSP2 problems under consideration are equivalent to standard single-machine scheduling problems [4] and as a consequence, if speeds of trains are arbitrary on the segment, then most of these problems are NP-hard [4], e.g., single-machine total tardiness or weighted number of late jobs problems. Note as well that STRSP2 problems can be also easily reformulated as shop scheduling problems [4] with Q machines. A reduction of STRSP with several stations to a job-shop scheduling problem is presented in [25], as well as, a shifting bottleneck algorithm to get a close to optimal schedule.

Here we present a reduction of STRSP2 with an arbitrary Q to a single-machine scheduling problem with setup times between two groups of jobs.

Denote $p_{max} = \max_{q=1,2,\dots,Q}\{p_q\}$ and $P = \sum_{q=1}^{Q} p_q$.

Lemma 1. *Assume that for a train $j' \in N_1'$ we have $C_{j'} = S_{j'} + P$ and train $i' \in N_1'$ is the next train which passes along the track. Then, without violation of feasibility conditions, the train i' can be scheduled as follows: $S_{i'} = \max\{r_{i'}, S_{j'} + p_{max}\}$ and $C_{i'} = S_{i'} + P$, i.e., the train i' departs at time point $\max\{r_{i'}, S_{j'} + p_{max}\}$ and goes without incurring any idle-time.*

Proof. Let $S_{j'}^q,\ S_{i'}^q, q = 1, 2, \ldots, Q$, be the start times to begin traveling upon segment q by the trains j' and i', respectively. To prove the feasibility of the schedule under consideration, we have only to show that $S_{i'}^q \geq S_{j'}^q + p_q, q = 1, 2, \ldots, Q$, i.e., the train i' only starts moving on segment q, when the train j' has already left it. We have $S_{j'}^q = S_{j'} + \sum_{l=1}^{q-1} p_l$ and $S_{i'}^q = S_{i'} + \sum_{l=1}^{q-1} p_l = \max\{r_{i'}, S_{j'} + p_{max}\} + \sum_{l=1}^{q-1} p_l \geq S_{j'}^q + p_q$, i.e., the Lemma is true. □

In fact, Lemma 1 defines the rhythm of departures of trains from the same subset if they follow one-by-one. In addition, notice that $\max\{r_{i'}, S_{j'} + p_{max}\}$ is the earliest possible departure time for the train i', since for the track $q,\ p_q = p_{max}$, we have $S_{i'}^q = S_{j'}^q + p_q$ and as consequence $|C_{j'} - C_{i'}| \geq p_{max}$ for any j', i' which belong to the same subset N_1' or N_2'. So, we can conclude the following:

Corollary 1. *For any j' and i' belong to the same subset N_1' or N_2', in any feasible schedule, we have $|S_{j'} - S_{i'}| \geq p_{max}$ and $|C_{j'} - C_{i'}| \geq p_{max}$.*

Let a sequence of trains $(j_1', j_2', \ldots, j_n')$ be the order in which the trains travel along the track. Evidently a feasible schedule corresponds to one and only one train sequence. Thus, an optimal schedule corresponds to just one *optimal* train sequence. According to the train sequence $(j_1', j_2', \ldots, j_n')$ a schedule can be computed as follows:

$$\begin{cases} S_{j_1'} = r_{j_1'}, & C_{j_1'} = S_{j_1'} + P, \\ S_{j_k'} = \max\{r_{j_k'}, S_{j_{k-1}'} + p_{max}\}, & C_{j_k'} = S_{j_k'} + P, \quad k = 2, \ldots, n', (*) \\ S_{j_k'} = \max\{r_{j_k'}, S_{j_{k-1}'} + P\}, & C_{j_k'} = S_{j_k'} + P, \quad k = 2, \ldots, n', (**). \end{cases} \tag{1}$$

$(*)$ holds if both j_k' and j_{k-1}' belong to the same subset N_1' or N_2', otherwise $(**)$ holds.

According to Lemma 1 this schedule is feasible. Furthermore, for the above mentioned objective functions, which are the monotone functions of the completion times of the trains, according to Lemma 1, by using rule (1) from an optimal train sequence we can construct an optimal schedule.

Based on these properties, the following reduction to a single-machine scheduling problem is proposed:

Single machine scheduling problem. A set $N = N_1 \bigcup N_2,\ N_1 \bigcap N_2 = \emptyset$ of n independent jobs that must be processed on a single-machine is given. Job preemption is not allowed. The machine can handle only one job at a time. Processing times p_j of jobs are equal to p, $\forall j \in N$. For each job $j \in N$, a due date d_j, a weight $w_j > 0$, and a release date $r_j \geq 0$ (i.e., the earliest possible starting time) can be given. A feasible solution is described by a permutation $\pi = (j_1, j_2, \ldots, j_n)$ of the jobs from the set N. The corresponding schedule can be uniquely determined by starting each job of this permutation as early as possible. Let $S_{j_k}(\pi)$, $C_{j_k}(\pi) = S_{j_k}(\pi) + p$ be the start and completion times of job j_k in the schedule resulting from the sequence π. If $j_k \in N_1$ and $j_{k+1} \in N_2$, then between jobs the machine has to be idle during a setup time $st = st_1$. If $j_k \in N_2$ and $j_{k+1} \in N_1$, then between jobs the machine has to be idle during a setup time $st = st_2$. There is no setup time between processing of jobs from the same subset, i.e., $st = 0$. In a feasible schedule $S_{j_{k+1}} = \max\{r_{j_{k+1}}, C_{j_k} + st\}$ holds. Objective functions are the same as for $STRSP2$. If $C_j(\pi) > d_j$, then job j is tardy and we have $U_j(\pi) = 1$, otherwise $U_j(\pi) = 0$. 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 and $C_{max}(\pi) = \max_{j \in N}\{C_j(\pi)\}$ is the makespan. We note these scheduling problems according to the traditional three-field notation $\alpha|\beta|\gamma$, e.g., $1|setup{-}times, N_1, N_2, p_j = p, r_j|C_{max}$ for the single-machine scheduling problem with equal-processing-times, setup times and release dates with the objective function minimizing makespan.

A survey of scheduling problems with setup times can be found, e.g., in [1]. In [19], a single-machine problem with jobs grouped in classes is considered, where setup times are only required when processing switches from jobs of one class to jobs of another class. Another single-machine problem with setup-times and jobs families is considered in [2]. Some algorithms for single-machine problems with batch setup times are presented in [20] where a number of batches and processing times are arbitrary, $r_j = 0$, $\forall j \in N$, the objective functions are maximal lateness and a weighted number of late jobs. The usefulness of the algorithms for the problems under consideration is discussed in the next Section. In [26] single-machine problems to minimize total weighted flow time or maximal lateness are considered. A single-machine scheduling with family jobs to minimize weighted number of tardy jobs is investigated in [18]. For the single-machine scheduling problem with due dates and batch setup times to minimize the weighted number of tardy jobs, a pseudo-polynomial algorithm was presented in [9].

Some results of equal-processing-time scheduling are presented in [17]. In the series of papers of Ph.Baptiste (see e.g. [3]) classical single-machine problems with equal-processing-time without precedence relations are considered.

It was shown that problems $1|p_j = p, r_j|\sum T_j$ and $1|p_j = p, r_j|\sum w_j U_j$ as well as some others are polynomially solvable. One can suppose that all such problems are polynomially solvable. However, the complexity status of $1|p_j = p, r_j|\sum w_j T_j$ is still open.

The problems $STRSP2|-|-$ for the previously mentioned objective functions can be reduced to $1|setup - times, N_1, N_2, p_j = p, -|-$ problems as follows. The subset of trains N_1' corresponds to the subset of jobs N_1, $|N_1| = |N_1'|$, and subset N_2' of trains to the subset N_2, $|N_2| = |N_2'|$, of jobs. Let q^*, $q^* \in \{1, 2, \dots, Q\}$ be the index of a segment for which $p_{q^*} = p_{max}$. Denote $TAIL_{left} = \sum_{l=1}^{q^*-1} p_l$, $TAIL_{right} = \sum_{l=q^*+1}^{Q} p_l$. Then, assume $p = p_{max}$, $st_1 = 2 \cdot TAIL_{right}$, $st_2 = 2 \cdot TAIL_{left}$. If $j \in N_1$, then release date $r_j = r_{j'} + TAIL_{left}$, else $r_j = r_{j'} + TAIL_{right}$. If $j \in N_1$, then due date $d_j = d_{j'} - TAIL_{right}$, else $d_j = d_{j'} - TAIL_{left}$. Weights are the same.

Let us consider a job sequence $(j_1, j_2, \dots, j_n)$ and a corresponding train sequence $(j_1', j_2', \dots, j_{n'}')$, where job $j_k, k = 1, 2, \dots, n,\ n = n'$, corresponds to train j_k', and schedules are determined by starting each job/train as early as possible (for jobs) or by rule (1) (for trains) according to these sequences. Then, for a job j and a train j' we can construct the following table of correspondence (see Tables 1 and 2).

Table 1: Parameters' correspondence

train/job	release date	due date	start time	completion time
$j \in N_1$	$r_j = r_{j'} + TAIL_{left}$	$d_j = d_{j'} - TAIL_{right}$	$S_{j'} + TAIL_{left}$	$C_{j'} - TAIL_{right}$
$j' \in N_1'$	$r_{j'}$	$d_{j'}$	$S_{j'}$	$C_{j'}$
$j \in N_2$	$r_j = r_{j'} + TAIL_{right}$	$d_j = d_{j'} - TAIL_{left}$	$S_{j'} + TAIL_{right}$	$C_{j'} - TAIL_{left}$
$j' \in N_2'$	$r_{j'}$	$d_{j'}$	$S_{j'}$	$C_{j'}$

In addition, we have the correspondence presented in Table 2 between the values of objective functions for the respective schedules.

So, we can conclude that for the functions mentioned in Table 2, an optimal job sequence $(j_1, j_2, \dots, j_n)$ corresponds to an optimal train sequence $(j_1', j_2', \dots, j_{n'}')$, i.e., $STRSP2|-|-$ problems can be reduced to corresponding $1|setup - times, N_1, N_2, p_j = p, -|-$ problems in a polynomial time. In the resulting single-machine problems, all jobs $j \in N_1$ start not earlier than

Table 2: Objective function values' correspondence

Objective function value $STRSP2\|-\|-$	Objective function value $1\|setup-times, N_1, N_2, p_j = p, -\|-$
$\sum w_{j'}U_{j'}$	$\sum w_{j'}U_{j'}$
$\sum T_{j'}$	$\sum T_{j'}$
$\sum w_{j'}C_{j'}$	$\sum w_{j'}C_{j'} + \sum_{j' \in N_1'} w_{j'} \cdot TAIL_{right} + \sum_{j' \in N_2'} w_{j'} \cdot TAIL_{left}$

$r = TAIL_{left}$ and jobs $j \in N_2$ start not earlier than $r = TAIL_{right}$. In the following Sections, some algorithms are presented for problems where all release dates r are equal to 0. These algorithms can be easily adopted for the initial scheduling problems with initial release dates.

A similar reduction can be made for

$STRSP2|r_j|C_{max}$ to $1|setup-times, N_1, N_2, p_j = p, r_j|C_{max}$,

but in such a reduction there is no tight connection between values of C_{max}, i.e., an optimal job sequence $(j_1, j_2, \dots, j_n)$ can correspond to a non optimal train sequence $(j_1', j_2', \dots, j_n')$.

The modification of solution algorithms for

$1|setup-times, N_1, N_2, p_j = p, r_j|C_{max}$,

presented in the next Section, can be used for $STRSP2|r_j|C_{max}$ as well.

Thus, in the next two Sections, solution algorithms for the following $1|setup-times, N_1, N_2, p_j = p, -|-$ problems are presented:

- $1|setup-times, N_1, N_2, p_j = p, r_j|C_{max}$;
- $1|setup-times, N_1, N_2, p_j = p, r_j|\sum C_j$;
- $1|setup-times, N_1, N_2, p_j = p|\sum w_jC_j$;
- $1|setup-times, N_1, N_2, p_j = p|\sum T_j$;
- $1|setup-times, N_1, N_2, p_j = p|\sum U_j$;
- $1|setup-times, N_1, N_2, p_j = p|\sum w_jU_j$.

Definition 1. We call schedules for $1|setup - times, N_1, N_2, p_j = p, -|-$ problems *left-shifted*, if they are determined by starting each job as early as possible. Obviously, for any aforementioned problem there are optimal schedules which are left-shifted.

Definition 2. Let $\Theta = \{t | t = r_j + x_1 \cdot p + x_2 \cdot st_1 + x_3 \cdot st_2,\ j \in \{1, 2, \ldots, n\},\ x_1,\ x_2,\ x_3 \in \{0, 1, 2, \ldots, n\},\ x_2 + x_3 \leq x_1\}$.That means Θ - is a set of possible starting times of jobs *in* a left-shifted schedule.
Notice that there are at most $O(n^4)$ values in set Θ, since there are no more than n different values for each $r_j,\ x_1,\ x_2,\ x_3$.

Lemma 2. *In all left-shifted schedules, job starting times belong to* Θ.

Proof. Let in a feasible left-shifted schedule which corresponds to a job sequence $(j_1, j_2, \ldots, j_n)$, job $j_k,\ k \geq 1$, be the earliest job for which $S_{j_k} \notin \Theta$, i.e., a starting time of its predecessor $S_{j_{k-1}} \in \Theta$. The earliest possible starting time S of the job j_k is defined as follows:

$$\begin{cases} S = \max\{r_{j_k}, S_{j_{k-1}} + p\}, & (*) \\ S = \max\{r_{j_k}, S_{j_{k-1}} + p + st_1\}, & (**) \\ S = \max\{r_{j_k}, S_{j_{k-1}} + p + st_2\}. & (***) \end{cases}$$

(*) holds if both j_k and j_{k-1} belong to the same subset N_1 or N_2, (**) holds if $j_k \in N_2$ and $j_{k-1} \in N_1$ and (***) holds if $j_k \in N_1$ and $j_{k-1} \in N_2$. Certainly, $S \in \Theta$. Since $S_{j_k} \notin \Theta$, we have $S < S_{j_k}$ and the schedule Π is not left-shifted.
$\square$

4. Algorithms for the Problems with Ordered Subsets N_1 and N_2

In this Section, solution algorithms for the following problems are presented:

- $1|setup - times, N_1, N_2, p_j = p, r_j|C_{max}$;
- $1|setup - times, N_1, N_2, p_j = p, r_j|\sum C_j$;
- $1|setup - times, N_1, N_2, p_j = p|\sum w_j C_j$;
- $1|setup - times, N_1, N_2, p_j = p|\sum T_j$.

All the algorithms are based on the same properties of optimal solutions and use the same search procedure.

Denote the subsets $N_1 = \{j_1, j_2, \ldots, j_{n_1}\}$ and $N_2 = \{i_1, i_2, \ldots, i_{n_2}\}$.

Lemma 3. *For each of the above mentioned problems there is an optimal schedule in which jobs are processed in the following special order:*

- *for the problems $1|setup-times, N_1, N_2, p_j = p, r_j|C_{max}$ and $1|setup-times, N_1, N_2, p_j = p, r_j|\sum C_j$ jobs are ordered according to non-decreasing release dates, i.e., $r_{j_1} \leq r_{j_2} \leq \cdots \leq r_{j_{n_1}}$ and $r_{i_1} \leq r_{i_2} \leq \cdots \leq r_{i_{n_2}}$;*
- *for the problem $1|setup-times, N_1, N_2, p_j = p|\sum w_j C_j$ jobs in each subset are ordered according to non-increasing weights, i.e., $w_{j_1} \geq w_{j_2} \geq \cdots \geq w_{j_{n_1}}$ and $w_{i_1} \geq w_{i_2} \geq \cdots \geq w_{i_{n_2}}$;*
- *for the problem $1|setup-times, N_1, N_2, p_j = p|\sum T_j$ jobs in each subset are ordered according to non-decreasing due dates, i.e., $d_{j_1} \leq d_{j_2} \leq \cdots \leq d_{j_{n_1}}$ and $d_{i_1} \leq d_{i_2} \leq \cdots \leq d_{i_{n_2}}$.*

Proof. If in an optimal schedule Π two jobs which belong to the same subset N_1 or N_2 are processed in violation of their corresponding order, then we can interchange them in the schedule without increasing the objective function value. □

Next, we present Algorithm 1 for the $1|setup-times, N_1, N_2, p_j = p, r_j|C_{max}$ problem and explain how it can be used for other problems considered in this Section. Assume that jobs in N_1 and N_2 are ordered according to Lemma 3. In this algorithm, we consider jobs $i_1, i_2, \ldots, i_{n_2}$ one-by-one. For each job i_k, $k = 1, 2, \ldots, n_2$, we have to take into account all positions l, $l = 0, 1, 2, \ldots, n_1$, where position l means that job j_l precedes job i_k in a constructed schedule and i_k precedes job j_{l+1}. If for the job i_k a position l is chosen, then for the job i_{k+1} only positions $l, l+1, \ldots, n_1$ have to be evaluated (see Lemma 3).

It is easy to establish a time bound for this algorithm. The sets of unscheduled jobs appearing in the arguments of the recursive procedure are of the form:

$$\{j_{l+1}, j_{l+2}, \ldots, j_{n_1}, i_k, i_{k+1}, \ldots, i_{n_2}\},$$

i.e., they are completely characterized by the index pairs (k, l). The arguments $S_{i_{k-1}}$ belong to the set Θ. Thus, we need to call function

Algorithm 1. $(F, \pi_{opt}) := Sequence(1, 0, -p)$, where π_{opt} is an optimal job sequence and $F = C^*_{max}$ is the minimal makespan.

Function $Sequence(k, l, S_{i_{k-1}})$

if $k = n_2 + 1$ **then**
    Schedule jobs $j_{l+1}, j_{l+2}, \ldots, j_{n_1}$ from the time $S_{i_{k-1}} + p + st_2$ according to rule (1);
    $\sigma := (j_{l+1}, j_{l+2}, \ldots, j_{n_1})$
    Return pair $(C_{j_{n_1}}, \sigma)$;
end if
if $l = n_1$ **then**
    Schedule jobs $i_k, i_{k+1}, \ldots, i_{n_2}$ from the time $S_{i_{k-1}} + p$ according to rule (1);
    $\sigma := (i_k, i_{k+1}, \ldots, i_{n_2})$
    Return pair $(C_{i_{n_2}}, \sigma)$;
end if
$S := S_{i_{k-1}}$;
$f_{min} := \infty$;
$\sigma_{min} := ()$;
for $pos := l$ to n_1 **do**
    if $pos = l$ **then**
        $S_{i_k} := \max\{r_{i_k}, S + p\}$;
        If $l = 0$ then $S := 0$ else $S := S + p + st_2$;
        $(f, \sigma) := Sequence(k + 1, l, S_{i_k})$;
    else
        $S_{j_{pos}} := \max\{r_{j_{pos}}, S\}$;
        $S_{i_k} := \max\{r_{i_k}, S_{j_{pos}} + p + st_1\}$;
        $(f, \sigma) := Sequence(k + 1, pos, S_{i_k})$;
        $S := S_{j_{pos}} + p$;
    end if
    if $f_{min} > f$ **then**
        $f_{min} := f$;
        $\sigma_{min} := (j_{l+1}, \ldots, j_{pos}, i_k, \sigma)$
    end if
end for
Return pair (f_{min}, σ_{min});

$Sequence(k, l, S_{i_{k-1}})$ at the most $O(n \cdot n \cdot n^4) = O(n^6)$ times. The run time of the function is $O(n)$. So, the running time of Algorithm 1 is $O(n^7)$.

According to Lemma 3, Algorithm 1 constructs an optimal job sequence in $O(n^7)$ time.

Notice that in [20] the algorithm presented does not take into account release dates, although it also deals with ordered subsets. In the algorithm, on each step $j = 1, 2 \ldots, n$ an unscheduled job is chosen to be inserted at the end of a partial job sequence to minimize its cost. For problem $1|setup - times, N_1, N_2, p_j = p, r_j|\sum C_j$, it does not lead to an optimal solution, e.g., for instance $n_1 = 3,\ n_2 = 1,\ st_1 = 1,\ st_2 = 3,\ r_{j_1} = 0,\ r_{j_2} = r_{j_3} = 4,\ r_{i_1} = 3$. For problem $1|setup - times, N_1, N_2, p_j = p|\sum w_j C_j$, it also does not lead to an optimal solution, e.g., for instance $n_1 = 2,\ n_2 = 1,\ st_1 = 0,\ st_2 = 100,\ w_{j_1} = 1,\ w_{j_2} = 3,\ w_{i_1} = 2$.
The function $Sequence(k, l, S_{i_{k-1}})$ can be easily modified to solve the problem $STRSP2|r_j|C_{max}$. We have to change lines 4 and 9 as follows:

4. Return pair $(C_{j_{n_1}} + TAIL_{right}, \sigma)$;

9. Return pair $(C_{i_{n_2}} + TAIL_{left}, \sigma)$;

To solve the problem $1|setup - times, N_1, N_2, p_j = p, r_j|\sum C_j$ we have to change the following lines:

4. Return pair $(\sum_{x:=l+1}^{n_1} C_{j_x}, \sigma)$, where C_{j_x} is the completion time of the job j_x in the partial schedule obtained from sequence σ, where jobs are processed from the time $S_{i_{k-1}} + p + st_2$;

9. Return pair $(\sum_{x:=k}^{n_2} C_{i_x}, \sigma)$, where C_{i_x} is the completion time of the job i_x in the partial schedule obtained from sequence σ, where jobs are processed from the time $S_{i_{k-1}} + p$;

25. **If** $f_{min} > f + f_{current}$ **then**,

26. $f_{min} := f + f_{current}$;

where $f_{current}$ is the total completion time of jobs in a partial sequence $(j_{l+1}, \ldots, j_{pos}, i_k)$ scheduled from time $S_{i_{k-1}} + p$. Remember that for the problem $1|setup - times, N_1, N_2, p_j = p, r_j|\sum C_j$, jobs in N_1 and N_2 have to be ordered according to Lemma 3.

Analogously, the procedure *Sequence* can be changed to solve problems $1|setup-times, N_1, N_2, p_j = p|\sum w_jC_j$ and $1|setup-times, N_1, N_2, p_j = p|\sum T_j$. Note that for these two problems $|\Theta| = O(n^3)$, since all the release dates are equal to 0, i.e., the running time of the modified algorithms for these problems is equal to $O(n^6)$.

Let's consider the following numerical instance of the problem (see Fig.1). Here we have $p = 2,\ st_1 = 1,\ st_2 = 2, n_1 = 4, r_{j_1} = 0,\ r_{j_2} = 5,\ r_{j_3} = 8,\ r_{j_4} = 14,\ n_2 = 2,\ r_{i_1} = 1,\ r_{i_2} = 6$. In Algorithm 1, all schedules corresponding to Lemma 3 are considered, an optimal of them is found. In Fig.1, three of them are presented. Schedule 2 is an optimal schedule for the problems $1|setup-times, N_1, N_2, p_j = p, r_j|C_{max}$ and $1|setup-times, N_1, N_2, p_j = p, r_j|\sum C_j$, where $C_{max} = 16$ and $\sum C_j = 57$.

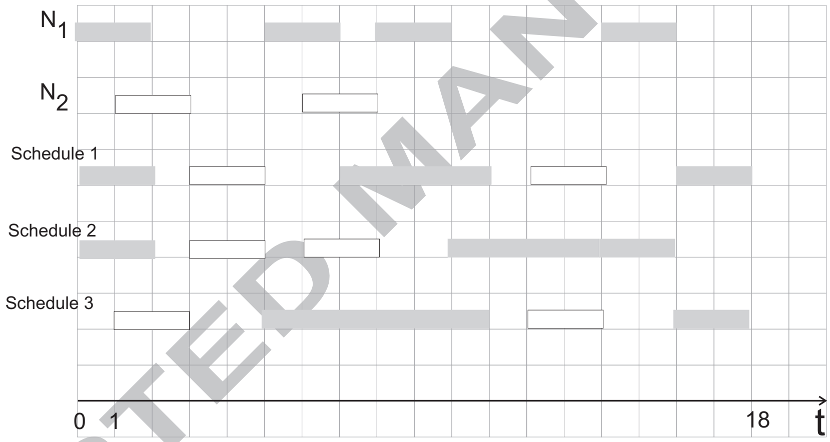

Figure 1: A numerical instance

Theorem 1. *The following problems are solvable in a polynomial time, in $O(n^7)$ time for:*

- $1|setup-times, N_1, N_2, p_j = p, r_j|C_{max}$ *and* $STRSP2|r_j|C_{max}$,
- $1|setup-times, N_1, N_2, p_j = p, r_j|\sum C_j$ *and* $STRSP2|r_j|\sum C_j$,

and in $O(n^6)$ time for:

- $1|setup-times, N_1, N_2, p_j = p|\sum w_jC_j$ *and* $STRSP2||\sum w_jC_j$,

- $1|setup - times, N_1, N_2, p_j = p|\sum T_j$ *and* $STRSP2||\sum T_j$,

by Algorithm 1 and its modifications.

Proof. Algorithm 1 constructs a schedule with the minimal objective function value among schedules which correspond to Lemma 3, i.e. an optimal schedule. The running time of the algorithm is presented above in the text. □

5. Problems with Partially Ordered Subsets

Lemma 4. *For the problem* $1|setup - times, N_1, N_2, p_j = p|\sum w_j U_j$, *there is an optimal left-shifted schedule, where on-time jobs from the same subset* N_1 *or* N_2 *are ordered according to non-decreasing due dates, i.e.,* $d_{j_1} \leq d_{j_2} \leq \cdots \leq d_{j_{n_1}}$ *and* $d_{i_1} \leq d_{i_2} \leq \cdots \leq d_{i_{n_2}}$.

Lemma 5. *Assume that jobs are ordered according to Lemma 4. For the problem* $1|setup-times, N_1, N_2, p_j = p|\sum U_j$, *there is an optimal left-shifted schedule and such indexes* x, $1 \leq x \leq n_1$, *and* $y, 1 \leq y \leq n_2$, *that only jobs* $j_x, j_{x+1}, \ldots, j_{n_1}, i_y, i_{y+1}, \ldots, i_{n_2}$ *are on-time and processed according to the order given by Lemma 4.*

Both Lemmas 4 and 5 can be proven similarly to Lemma 3.

Theorem 2. *The problem* $1|setup-times, N_1, N_2, p_j = p|\sum U_j$ *can be solved in* $O(n^7 \log n)$ *time.*

Proof. According to Lemmas 4 and 5 for the problem $1|setup - times, N_1, N_2, p_j = p|\sum U_j$, we have to choose indexes x and y, such that $x + y \to \max$ and jobs $j_x, j_{x+1}, \ldots, j_{n_1}, i_y, i_{y+1}, \ldots, i_{n_2}$ can all be processed on-time at the beginning of a schedule. Thus, we have to take into account at most $(n_1 + 1)\log(n_2 + 1)$ pairs (x, y). For each of these pairs, we solve the problem $1|setup - times, N_1, N_2, p_j = p|\sum T_j$ with the set of jobs $\{j_x, j_{x+1}, \ldots, j_{n_1}, i_y, i_{y+1}, \ldots, i_{n_2}\}$ by a modification of Algorithm 1. If $\sum T_j(\pi^*) = 0$, then pair (x, y) is feasible. □

For the problem $1|setup - times, N_1, N_2, p_j = p|\sum w_j U_j$, a dynamic programming polynomial time algorithm can be suggested. This algorithm is based on the following assumptions. Note jobs in $N = \{H_1, H_2, \ldots, H_n\}$, where $w_{H_1} \leq w_{H_2} \leq \cdots \leq w_{H_n}$. If $w_{H_k} = w_{H_{k+1}}$, then $d_{H_k} \leq d_{H_{k+1}}$. Jobs from N_1 and N_2 are noted and ordered according to Lemma 4. Let $H_n \in N_2$ and $H_n = i_k$. For H_n, a position in a schedule is defined by a pair (t, l), where $t \in \Theta$ is the starting time for the job, the index $l = 0, 1, \ldots, n_1$ means that on-time jobs from the subset $\{j_1, j_2, \ldots, j_l\}$ precede the job H_n in a schedule and on-time jobs from the subset $\{j_{l+1}, j_{l+2}, \ldots, j_{n_1}\}$ are scheduled after H_n. A position $(-, n_1 + 1)$ means that the job H_n is late and processed at the end of schedule from time $T \in \Theta$.

Then, for each position (t, l) among $O(n^4)$ possible, we can decompose the initial problem into two independent subproblems:

- with a set of jobs $N_{left} = \{j_1, j_2, \ldots, j_l, i_1, i_2, \ldots, i_{k-1}\}$ which have to be processed in interval $[0, t)$;
- with a set of jobs $N_{right} = \{j_{l+1}, j_{l+2}, \ldots, j_{n_1}, i_{k+1}, i_{k+2}, \ldots, i_{n_2}\}$ which have to be processed in interval $[t + p, T)$.

Denote $T_{max} = \max\{t | t \in \Theta\}$. Note that for the $1|setup-times, N_1, N_2, p_j = p| \sum w_j U_j$ problem, $|\Theta| = O(n^3)$, since all release dates of jobs are equal to 0.

Next, we present Algorithm 2 to solve this problem. It is easy to establish a time bound for this algorithm. The sets of unscheduled jobs appearing in the arguments of the recursive procedure are in the form of

$$N' = \{j_{l_1}, j_{l_1+1}, \ldots, j_{l_2}, i_{k_1}, i_{k_1+1}, \ldots, i_{k_2}\},$$

$$N' \bigcap \{H_{h+1}, H_{h+2}, \ldots, H_n\} = \emptyset,$$

i.e., they are completely delineated by the five indices h, k_1, k_2, l_1, l_2. The arguments t_1, t_2 belong to the set Θ.
Thus, we need to call function $SequenceWU(h, t_1, t_2, I_1, I_2, k_1, k_2, l_1, l_2)$ at most $O(n^{5+3+3})$ times. The run time for this function is $O(n^4)$. So, the running time of Algorithm 2 is $O(n^{15})$. Note that if $H_h \notin N'$, then assign $h := \max_{x=1,\ldots,h-1}\{H_x \in N'\}$.

6. Conclusion

In this paper, the single-track railway scheduling problem with 2 stations and Q segments was considered. A polynomial time reduction to the single-machine scheduling problems with setup-times was presented. The reduction was made owing to an observation that trains from the same subset can begin traveling every p_{max} time unit if they follow one-by-one. For some objective functions and additional restrictions, these problems were solved polynomially. In Table 3 polynomial time cases are presented.

Table 3: Complexity of algorithms

Railway problem	Corresponding single-machine problem	Running time of algorithms
$STRSP2\|r_j\|C_{max}$	$1\|setup-times, N_1, N_2, p_j = p, r_j\|C_{max}$	$O(n^7)$
$STRSP2\|r_j\|\sum C_j$	$1\|setup-times, N_1, N_2, p_j = p, r_j\|\sum C_j$	$O(n^7)$
$STRSP2\|\|\sum w_jC_j$	$1\|setup-times, N_1, N_2, p_j = p\|\sum w_jC_j$	$O(n^6)$
$STRSP2\|\|\sum T_j$	$1\|setup-times, N_1, N_2, p_j = p\|\sum T_j$	$O(n^6)$
$STRSP2\|\|\sum U_j$	$1\|setup-times, N_1, N_2, p_j = p\|\sum U_j$	$O(n^7 \log n)$
$STRSP2\|\|\sum w_jU_j$	$1\|setup-times, N_1, N_2, p_j = p\|\sum w_jU_j$	$O(n^{15})$

Motivated by problems of single path trains scheduling, this paper was finally focused on solving equivalent single machine-scheduling problems and proofs of their complexity.

As a perspective for our research, we have sought how to reduce the complexity of algorithms proposed. Another question arises for single-machine equal-processing-time scheduling problems without setup-times and precedence relations: "Are there problems with equal processing time of jobs, which are NP-hard?"

References

[1] Allahverdi, A., Ng, C.T., Cheng, T.C.E., Kovalyov, M.Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187, 985-1032.

[2] Baker, K. R. (1999). Heuristic procedures for scheduling job families with setups and due dates. *Naval Research Logistics*, 46(8), 978-991.

[3] Baptiste, Ph., Brucker, P.,Knust, S., Timkovsky, V.G. (2004). Ten notes on equal-processing-time scheduling. *4OR: Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2, 111-127.

[4] Brucker, P. (2001). Scheduling Algorithms. Springer-Verlag.

[5] Brucker, P., Heitmann, S., Knust, S. (2002). Scheduling railway traffic at a construction site. *OR Spectrum*, 24(1), 19-30.

[6] Coene, S., Spieksmay F.C.R., Berghe, G. V. (2011). The Lockmaster's Problem, *ATMOS 2011*, Saarbruecken.

[7] Cordeau, J.-F., Toth, O., Vigo, D. (2000). A survey of optimization models for train routing and scheduling, *Transportation Science*, 32(4), 380-396.

[8] Disser, Y., Klimm, M., Lubbecke, E. (2014). Bidirectional Scheduling on a Path, *Matheon Preprint #1060, Research Center Matheon,Technische Universitat Berlin*, 32 pages.

[9] Erel, E., Ghosh, J.B. (2007). Batch scheduling to minimize the weighted number of tardy jobs. *Computers & Industrial Engineering*, 53(3), 394-400.

[10] Gafarov, E.R., Dolgui., A. (2012). Two-Station Single Track Railway Scheduling Problem With Equal Speed of Trains. *Research Report RR-12-09, LIMOS, CNRS UMR 6158*, April 7th, 2012, 14 pages.

[11] Gafarov, E., Dolgui, A., Lazarev A. (2012). Two station single track railway scheduling problem with equal speed of trains, *21st International Symposium on Mathematical Programming*, August 19-24, 2012, Berlin, Germany.

[12] Gafarov, E., Lazarev A., Dolgui, A. (2012). Solution algorithms for the two-station single track railway scheduling problem. In: *Proceedings of the 6th Multidisiplinary International Scheduling Conference: Theory and Applications (MISTA 2013)*, Ghent, Belgium, 27-29 August 2013, Graham Kendall, Greet Vanden Berghe, Barry McCollum (Eds.), p. 636-640.

[13] Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. (1979). Optimization and approximation in deterministic machine scheduling: a survey. *Annals of Discrete Mathematics*, 5, 287-326.

[14] Hajiaghaei-Keshteli, M., Aminnayeri, M., Fatemi Ghomi S.M. (2014). Integrated Scheduling of Production and Rail Transportation, *Computers & Industrial Engineering*, DOI: 10.1016/j.cie.2014.05.026.

[15] Heydar, M., Petering, M.E.H., Bergmann, D.R. (2013) Mixed integer programming for minimizing the period of a cyclic railway timetable for a single-track with two train types, *Computers & Industrial Engineering*, 66, 171-185.

[16] Higgins, A., Kozan, E., Ferreira, L. (1996). Optimal Scheduling of Trains on a Single Line Track. *Transportation Research*, 30, 147-161.

[17] Kravchenko, S., Werner, F. (2011). Parallel Machine Problems with Equal Processing Times: A Survey. *Journal of Scheduling*, 14(5), 435-444.

[18] Li, S.-S., Chen, R.-X. (2014). Single-machine parallel-batching scheduling with family jobs to minimize weighted number of tardy jobs. *Computers & Industrial Engineering*, 73, 5-10.

[19] Mason, A. J., Anderson, E.J. (1991) Minimizing flow time on a single-machine with job classes and setup times. *Naval Research Logistics*, 38(3), 333-350.

[20] Monma, C.L., Potts, C.N. (1989) On the Complexity of Scheduling with Batch Setup Times. *Operations Research*, 37(5), 798-804

[21] de Oliveira, E.S. (2001) Solving Single Track Railway Scheduling Problem Using Constraint Programming. The University of Leeds. School of Computing. PhD Thesis.

[22] Sourd, F. (2009) A new tool for reducing delays, *Avancees*, SNCF, 1(October), http://www.avancees.eu/01/index.htm.

[23] Sourd, F. (2011) Application of scheduling theory to solve real-life railway problems, In: *Proceedings of Conference on Optimization and Practices in Industry (COPI'11)*, Paris, France.

[24] Szpigel, B. (1972). Train Scheduling on a Single Track Railway. In: *Proceedings of IFORS Conference on Operational Research'72*, M. Roos (Editor), 343-352.

[25] Sotskov, Y.N., Gholami, O. (2012). Shifting bottleneck algorithm for train scheduling on a single-track railway. In: *Proceedings of 14th IFAC Symposium on Information Control Problems in Manufacturing*, May 23-25, 2012, Bucharest, Romania, T. Borangiu, I. Dumitrache, A. Dolgui, F. Filip (Editors), Elsevier Science, IFAC-PapersOnline.net (ISSN 1474-6670), 87-92.

[26] Webster, S., Baker, K.R. (1995). Scheduling Groups of Jobs on a Single Machine. *Operations Research*, 43(4), 692-703.

Algorithm 2.
$(F, SCHEDULE_{opt}) := SequenceWU(n, 0, T_{max}, 0, 0, 0, n_2, 0, n_1)$, where $SCHEDULE_{opt}$ is an unfeasible job schedule, which can be transformed to an optimal one by rescheduling jobs started at time T_{max}, and $F = \sum w_j(1 - U_j)$ is the maximal weighted number of on-time jobs.
Function $SequenceWU(h, t_1, t_2, I_1, I_2, k_1, k_2, l_1, l_2)$

$f_{max} := -\infty$;//weighted number of on-time jobs;
$\sigma_{max} := \{\}$;//a set of pairs (h, S_h), i.e., a schedule.
if $H_h \in N_1$ **then**
    $I = 1$; $pos_1 := k_1$; $pos_2 := k_2$;
    if $I_1 = 1$ **then** $t_{min} := t_1$;
    if $I_1 = 2$ **then** $t_{min} := t_1 + st_2$;
    if $I_1 = 0$ **then** $t_{min} := 0$;
    if $I_2 = 1$ **then** $t_{max} := t_2 - p$;
    if $I_2 = 2$ **then** $t_{max} := t_2 - p - st_1$;
    if $I_2 = 0$ **then** $t_{max} := T_{max}$;
else
    $I = 2$; $pos_1 := l_1$; $pos_2 := l_2$;
    if $I_1 = 1$ **then** $t_{min} := t_1 + st_1$;
    if $I_1 = 2$ **then** $t_{min} := t_1$;
    if $I_1 = 0$ **then** $t_{min} := 0$;
    if $I_2 = 1$ **then** $t_{max} := t_2 - p - st_2$;
    if $I_2 = 2$ **then** $t_{max} := t_2 - p$;
    if $I_2 = 0$ **then** $t_{max} := T_{max}$;
end if
for $pos := pos_1$ to pos_2 **do**
    for each $t \in \Theta,\ t_{min} \le t \le t_{max},\ t + p \le d_{H_h}$ **do**
        if $H_h \in N_1$ **then**
            Let $j_l = H_h$;
            $(\sigma_1, f_1) := SequenceWU(h-1, t_1, t+p, I_1, I, k_1, pos, l_1, l-1)$;
            $(\sigma_2, f_2) := SequenceWU(h-1, t+p, t_2, I, I_2, pos, k_2, l+1, l_2)$;
        else
            Let $i_k = H_h$;
            $(\sigma_1, f_1) := SequenceWU(h-1, t_1, t+p, I_1, I, k_1, k-1, l_1, pos)$;
            $(\sigma_2, f_2) := SequenceWU(h-1, t+p, t_2, I, I_2, k+1, k_2, pos, l_2)$;
        end if
        if $f_1 + f_2 + w_{H_h} > f_{max}$ **then**
            $f_{max} := f_1 + f_2 + w_{H_h}$;
            $\sigma_{max} := \sigma_1 \bigcup \sigma_2 \bigcup \{(h, t)\}$;
        end if
    end for
end for
//In addition, we consider the case, when the job H_h is late.
$(\sigma_1, f_1) := SequenceWU(h-1, t_1, t_2, I_1, I_2, k_1, k_2, l_1, l_2)$;
if $f_1 > f_{max}$ **then**
    $f_{max} := f_1$;
    $\sigma_{max} := \sigma_1 \bigcup \{(h, T_{max})\}$;
end if
Return pair (f_{max}, σ_{max});

Two-Station Single-Track Railway Scheduling Problem With Trains of Equal Speed

Evgeny R. Gafarov[a], Alexandre Dolgui[b], Alexander A. Lazarev[c]

[a]**Corresponding author** *Institute of Control Sciences of the Russian Academy of Sciences, Profsoyuznaya st. 65, 117997 Moscow, Russia, email: axel73@mail.ru*
[b]*Ecole Nationale Superieure des Mines, CNRS UMR6158, LIMOS, F-42023 Saint-Etienne, France, email: dolgui@emse.fr*
[c]*Institute of Control Sciences of the Russian Academy of Sciences, Profsoyuznaya st. 65, 117997 Moscow, Russia, email: jobmath@mail.ru*

Abstract

In this paper, the single-track railway scheduling problem with two stations and several segments of the track is considered. Two subsets of trains are given, where trains from the first subset go from the first station to the second station, and trains from the second subset go in the opposite direction. The speed of trains over each segment is the same. A polynomial time reduction from the problem under consideration to a special case of the single-machine equal-processing-time scheduling problem with setup times is presented. Different polynomial time algorithms are developed for special cases with divers objective functions under various constraints. Moreover, several theoretical results which can be ranked in a series of similar investigations of NP-hardness of equal-processing-time single-machine scheduling problems without precedence relations are obtained.

Keywords: Single machine scheduling, Setup times, Transportation, Train scheduling, Computational complexity, Polynomial time algorithms
2000 MSC: 90B35

Highlights

A single track railway scheduling for 2 stations and several segments is considered.
Often this situation concerns the case of maintenance of one track of a double track line.
A reduction to the single machine scheduling problems with setup-times is presented.
Polynomial time solution algorithms were developed.
Can serve as a basis to develop efficient algorithms for decision support systems.