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

Evgeny R. Gafarov

Ecole Nationale Superieure des Mines, FAYOL-EMSE, CNRS:UMR6158, LIMOS, F-42023 Saint-Etienne, France,
Institute of Control Sciences of the Russian Academy of Sciences, Profsoyuznaya st. 65, 117997 Moscow, Russia,
email: axel73@mail.ru

Alexandre Dolgui

Ecole Nationale Superieure des Mines, FAYOL-EMSE, CNRS:UMR6158, LIMOS, F-42023 Saint-Etienne, France
email: dolgui@emse.fr

Abstract

In this paper, the single track railway scheduling problem with 2 stations and Q segments of the track is considered. Two subsets of trains N_1' and N_2' are given, where trains from N_1' go from the station 1 to the station 2, and trains from N_2' 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. For this special case with different objective function under different constraints polynomial time solution algorithms are presented.

Keywords: Single track railway scheduling problem, single machine, setup times, equal processing time.

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.

For the problem different optimization criterion on different constraints are considered:

- **optimization criterion**: initial scheduling to minimize total idle-time, rescheduling to minimize delays etc.;
- **additional constraints**: overtake segments, release and due dates, priority and velocity of trains etc.

A literature review on the problem can be found, e.g., in [10].

In this paper, we consider a special **core** subcase with only two stations. The subcase has its own practical significance and appears in private railways, e.g., when a company transports loads between two production points.

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' trains. Trains from the subset N'_1 go from the station 1 to the 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 on Q segments $1, 2, \ldots, Q$. Trains from the set N'_1 traverse segments in an order $1 \to 2 \to \cdots \to Q$ and trains from the set N'_2 in an order $Q \to Q-1 \to \ldots 1$. At most only one train can be on any track segment at a 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 $q,\ q = 1, 2 \ldots, Q$, a traversing time p_q is given, in which a train $j \in N$ traverses the segment, i.e., for each segment $q,\ q = 1, 2 \ldots, Q$, all the trains go with the same speed[2]. 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 job j' from the departure station and $C_{j'}(\Pi)$ is an arrival time to the destination station. Then in a feasible schedule we have:

- $C_{j'} \geq S_{j'} + \sum_{q=1}^{Q} p_q,\ j' \in N'$;
- 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 $d_{j'} \geq 0$, a weight $w_{j'} \geq 0$, 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 $U_{j'}(\Pi) = 1$, otherwise $U_{j'}(\Pi) = 0$. If $C_{j'}(\Pi) \leq d_{j'}$, then train j' is on-time. Moreover, let $T_{j'}(\Pi) = \max\{0, C_{j'}(\Pi) - d_{j'}\}$ be the tardiness of train j' and $C_{max}(\Pi) = \max_{j' \in N'}\{C_{j'}(\Pi)\}$ is the makespan in schedule Π. For the STRSP2 with release dates of minimizing the makespan C_{max}, the objective is to find an optimal schedule Π^* that minimizes the makespan C_{max}. This problem is denoted $STRSP2|r_j|C_{max}$ (similar to the traditional three-field notation $\alpha|\beta|\gamma$ for scheduling problems

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

[2] This assumption is not far away from practice, since most trains travel at a maximal speed allowed.

proposed by Graham et al. [6], where α describes the resource environment, β gives the activity characteristics and further constraints and γ describes the objective function.) In addition, in this paper we deal with the following $STRSP2$ with different objective functions and further constraints:

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

To the best of our knowledge there does not exist any publications for this set of problems. But the following assumptions and notes can be easily made. If $Q = 1$ then the problems under consideration are equivalent to classical single machine scheduling problems [5] and as consequence if speeds of trains are arbitrary for the segment, then some of problems are NP-hard [5]. Note as well that the problems can be also easily reformulated like shop scheduling problems [5] with Q machines. Multi-stage scheduling problems where $Q = 2, 3$, are considered, e.g., in the classical paper [7].

The rest of the paper is organized as follows. In Section 1, a polynomial time reduction of $STRSP2$ to a special case of the single machine equal-processing-time scheduling problem with setup times is suggested. Polynomial time algorithms for the single machine problem with above mentioned objective functions are presented in Sections 2 and 3.

1 Reduction of STRSP2 to the Single Machine Scheduling Problem

Denote $p_{max} = \max_{q=1,2,\ldots,Q}\{p_q\}$ 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 the train $i' \in N_1'$ is the next train which passes the track. Then without violation of feasibility's conditions the train i' can be scheduled as follows: $S_{i'} = \max\{r_{i'}, S_{i'} + p_{max}\}$ and $C_{i'} = S_{i'} + P$, i.e. the train i' departs from the time point $\max\{r_{i'}, S_{j'} + p_{max}\}$ and leaves without idle-times.*

Proof. Let $S^q_{j'},\ S^q_{i'}, q = 1, 2, \ldots, Q$, be the start times to travel 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 $S^q_{i'} \geq S^q_{j'} + p_q, q = 1, 2, \ldots, Q$, i.e. the train i' starts moving on segment q, when the train j' has left it already. We have $S^q_{j'} = S_{j'} + \sum_{l=1}^{q-1} p_l$ and $S^q_{i'} = S_{j'} + \sum_{l=1}^{q-1} p_l = \max\{r_{i'}, S_{j'} + p_{max}\} + \sum_{l=1}^{q-1} p_l \geq S^q_{j'} + p_q$, 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, note 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 $S^q_{i'} = S^q_{j'} + p_q$ and as consequence $|C_{j'} - C_{i'}| \geq p_{max}$ for any j', i' belong to the same subset N'_1 or N'_2.

Lemma 2 *For any j' and i' belong to the same subset N'_1 or N'_2 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 an order in which the trains traverse the track. It is obvious, that 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} belongs to the same subset N'_1 or N'_2, else $(**)$ holds.

According to Lemma 1 this schedule is feasible. Furthermore, for the above mentioned objective functions, which are monotone functions of the completion times of the trains, according to Lemma 2, algorithm (1) from an optimal train sequence constructs an optimal schedule.

Based on these properties, the following reduction to the single machine scheduling problem is proposed.

Single machine scheduling problem Given 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. Job preemption is not allowed. The machine can handle only one job at a time. Processing times of jobs are equal $p,\ \forall j \in N$. For each job $j \in N$, a due date $d_j \geq 0$, a weight $w_j \geq 0$, 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 of the set N from which the corresponding schedule can be uniquely determined by starting each job 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 like 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 the 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 to minimize makespan.

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. 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, $q \in \{1, 2, \ldots, Q\}$ be the index of 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 $(j_1, j_2, \ldots, j_n)$, corresponding train sequence $(j_1', j_2', \ldots, j_n')$, where a job $j_k, k = 1, 2, \ldots, n$, corresponds to a train j_k', and schedules which are determined by starting each job/train as early as possible (for jobs) or by algorithm (1) (for trains) according to the sequences. Then for a job j and a train j' we can construct the following table of correspondence:

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 following correspondence of the objective functions values for the respective schedules:

Table 2: Objective function values' correspondence

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

So, we can conclude that for the functions mentioned in the Table 2 an optimal job sequence $(j_1, j_2, \ldots, j_n)$ corresponds to an optimal train sequence $(j_1', j_2', \ldots, j_n')$, i.e. $STRSP2\vert-\vert-$ problems can be reduced to corresponding $1\vert setup-times, N_1, N_2, p_j = p, -\vert-$ problems in polynomial time. In the resulting single machine problems all jobs $j \in N_1$ starts not earlier than time $r = TAIL_{left}$ and jobs $j \in N_2$ starts not earlier than time $r = TAIL_{right}$. In the following Sections some algorithms are presented for problems where all release dates r equals 0, but these algorithms can be easily adopted for the resulting problems.

A similar reduction can be made for $STRSP2\vert r_j\vert C_{max}$ to $1\vert setup-times, N_1, N_2, p_j = p, r_j\vert C_{max}$, but in such a reduction there is no tight connections between values of C_{max}, i.e., an optimal job sequence $(j_1, j_2, \ldots, j_n)$ can correspond to a not optimal train sequence $(j_1', j_2', \ldots, j_n')$. Although, the modification of solution algorithms for $1\vert setup-times, N_1, N_2, p_j = p, r_j\vert C_{max}$ presented in the next Section can be used for $STRSP2\vert r_j\vert C_{max}$ as well.

So, in the next two Sections solution algorithms not for $STRSP2\vert-\vert-$ problems but for the following $1\vert setup-times, N_1, N_2, p_j = p, -\vert-$ problems are presented:

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

A survey of scheduling problems with setup times can be found, e.g., in [1]. In [9], 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 results in equal-processing-time scheduling are presented in [3, 8]. In [4] authors propose a compact MIP formulation for single machine problems with piecewise linear objective functions, which has been shown to be efficient for academic benchmarks as well as on real-life industrial problems.

Definition 1. We will 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. It is obvious that for any afore mentioned problem there are optimal schedules which are left-shifted.
Definition 2. Let $\Theta = \{t|t = r_j + x_1 \cdot p + x_2 \cdot st_1 + x_3 \cdot st_2,\ j \in \{1, 2, \ldots, n\},\ x_1, x_2, x_3 \in \{0, 1, 2, \ldots, n\},\ x_2 + x_3 \leq x_1\}$.
Notice that there are at most $O(n^4)$ values in Θ.

Lemma 3 *In all left-shifted schedules, job starting times belong to* Θ.

Proof. Let in a feasible left-shifted schedule Π job $j_k,\ 1 < k < n$, 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} belongs 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 $j_{k-1} \in N_2$. Obviously, $S \in \Theta$. Since $S_{j_k} \notin \Theta$, we have $S < S_{j_k}$ and the schedule Π is not left-shifted. □

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

Lemma 4 *For each of the above mentioned problems there is an optimal schedule in which jobs are processed in a 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. The Lemma can be easily proven as follows. If in an optimal schedule Π two jobs which belong to the same subset N_1 or N_2 are processed in violation of corresponding order then we can interchange them in the schedule without increasing of the objective function value. □

Next we present a solution algorithm 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 the Section. Assume that jobs in N_1 and N_2 are ordered according to Lemma 4. In the algorithm one-by-one we consider jobs $i_1, i_2, \ldots, i_{n_2}$. For each job $i_k,\ k = 1, 2, \ldots, n_2$, we have to consider 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 considered (see Lemma 4).

It is easy to establish a time bound for this algorithm. The sets of unscheduled jobs appear 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 $Sequence(k, l, S_{i_{k-1}})$ at the most $O(n^6)$ times. The run time of the function is $O(n)$. So, the running time of the Algorithm 2 is $O(n^7)$.

According to Lemma 4 the Algorithm constructs an optimal job sequence in $O(n^7)$ time.
The function can be easily modified to solve the problem $STRSP2|r_j|C_{max}$. We have to change lines 4 and 9 of the function *Sequence*:

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 of the function:

4. Return pair $(\sum_{x:=l+1}^{n_1} C_{j_x}, \sigma)$, where C_{j_x} – completion time of the job j_x in the partial schedule obtained from sequence σ, where jobs are processed from the time $S_{i_{k-1}} + p + st_1$;

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

1: **if** $k = n_2 + 1$ **then**
2: Schedule jobs $j_{l+1}, j_{l+2}, \ldots, j_{n_1}$ from the time $S_{i_{k-1}}+p+st_1$ according to the Algorithm (1);
3: $\sigma := (j_{l+1}, j_{l+2}, \ldots, j_{n_1})$
4: Return pair $(C_{j_{n_1}}, \sigma)$;
5: **end if**
6: **if** $l = n_1$ **then**
7: Schedule jobs $i_k, i_{k+1}, \ldots, i_{n_2}$ from the time $S_{i_{k-1}} + p$ according to the Algorithm (1);
8: $\sigma := (i_k, i_{k+1}, \ldots, i_{n_2})$
9: Return pair $(C_{i_{n_2}}, \sigma)$;
10: **end if**
11: $S := S_{i_{k-1}}$;
12: $f_{min} := \infty$;
13: $\sigma_{min} := ()$;
14: **for** $pos := l$ to n_1 **do**
15: **if** $pos = l$ **then**
16: $S_{i_k} := \max\{r_{i_k}, S + p\}$;
17: If $l = 0$ then $S := 0$ else $S := S + p + st_2$;
18: $(f, \sigma) := Sequence(k + 1, l, S_{i_k})$;
19: **else**
20: $S_{j_{pos}} := \max\{r_{j_{pos}}, S\}$;
21: $S_{i_k} := \max\{r_{i_k}, S_{j_{pos}} + p + st_1\}$;
22: $(f, \sigma) := Sequence(k + 1, pos, S_{i_k})$;
23: $S := S_{j_{pos}} + p$;
24: **end if**
25: **if** $f_{min} > f$ **then**
26: $f_{min} := f$;
27: $\sigma_{min} := (j_{l+1}, \ldots, j_{pos}, i_k, \sigma)$
28: **end if**
29: **end for**
30: **Return** pair (f_{min}, σ_{min});

Algorithm 2.
$(F, \pi_{opt}) := Sequence(1, 0, -p)$, where π_{opt} is an optimal job sequence and $F = C^*_{max}$ is the minimal makespan;

...

9. Return pair $(\sum_{x:=k}^{n_2} C_{i_x}, \sigma)$, where C_{i_x} – 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**,// 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 $S_{i_{k-1}} + p$;
26. $f_{min} := f + f_{current}$;

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 4.

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

Lemma 5 *The following problems are solvable in $O(n^7)$ or in $O(n^6)$ time by Algorithm 2 and its modifications:*

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

3 Problems with Partially Ordered Subsets

Lemma 6 *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. $d_{j_1} \le d_{j_2} \le \cdots \le d_{j_{n_1}}$ and $d_{i_1} \le d_{i_2} \le \cdots \le d_{i_{n_2}}$.*

Lemma 7 *Assume, that the jobs are ordered according to Lemma 6. 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, $1 \le x \le n_1$ and $y, 1 \le y \le n_2$, where 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 from Lemma 6.*

Both Lemmas 6 and 7 can be proven similarly to Lemma 4.

So, 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 consider at most $(n_1 + 1)\log(n_2 + 1)$ pairs (x, y). For each of the pairs we solve the problem $1|setup - times, N_1, N_2, p_j = p|\sum T_j$ with 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 the Algorithm 2. If $\sum T_j(\pi^*) = 0$ then pair (x, y) is feasible. We can conclude the following:

Lemma 8 *The problem* $1|setup-times, N_1, N_2, p_j = p|\sum U_j$ *is solved in* $O(n^7 \log n)$ *time.*

For the problem $1|setup - times, N_1, N_2, p_j = p|\sum w_j U_j$ a dynamic programming polynomial time algorithm is suggested. The algorithm 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 6. Let $H_n \in N_2$ and $H_n = i_k$. For the H_n a position in a schedule is defined by a pair (t, l), where $t \in \Theta$ is the starting time of 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 in 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 to 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 equal 0. Next, we present a solution Algorithm 3 for this problem.
It is easy to establish a time bound for this algorithm . The sets of unscheduled jobs appear 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 index fives 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 of the function is $O(n^4)$. So, the run time of the Algorithm 3 is $O(n^{15})$.
Note, if $H_h \notin N'$ then assign $h := \max_{x=1,\ldots,h-1}\{H_x \in N'\}$.

Conclusion

In the paper, the single track railway scheduling problem with 2 stations and Q segments is considered. A polynomial time reduction to the single machine scheduling problem with setup-times is presented. The following polynomial time algorithms are proposed:

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

We suppose that running times of Algorithms can be substantivally reduced after more detailed analysis.

Acknowledgement

The authors are grateful to Chris Yukna for his help regarding the English presentation.

References

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

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

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

[4] Ph. Baptiste , R.Sadykov, On scheduling a single machine to minimize a piecewise linear objective function: A compact MIP formulation, Naval Research Logistics 56, Issue 6 (2009) , 487–502.

[5] P. Brucker, Scheduling Algorithms, Springer-Verlag, 2001.

[6] R.L. Graham , E.L. Lawler, J.K. Lenstra , A.H.G. Rinnooy Kan, Optimization and approximation in deterministic machine scheduling: a survey, Ann. Discrete Math. 5 (1979), 287 –326.

[7] S. M. Johnson, Optimal two and three stage production schedules with setup times included, Naval Research Logistics Quarterly 1, Issue 1 (1954), 61 – 68.

[8] S. Kravchenko and F. Werner, Parallel Machine Problems with Equal Processing Times: A Survey, Journal of Scheduling 14, No. 5 (2011), 435 - 444.

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

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

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});

Algorithm 3.
$(F, SCHEDULE_{opt}) := SequenceWU(n, 0, T_{max}, 0, 0, 0, n_2, 0, n_1)$, where $SCHEDULE_{opt}$ is an unfeasible job schedule, which can be transformed to 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;