MISTA 2015

Single Track Train Scheduling

Jonas Harbering · Abhiram Ranade · Marie Schmidt

Abstract In this work we consider the Single Track Train Scheduling Problem. The problem consists in scheduling a set of trains from opposite sides along a single track. The track passes intermediate stations and the trains are only allowed to pass each other at those stations. This problem has a close relation to minimizing the makespan in a job shop scheduling problem with two counter routes and no preemption. We develop a lower bound on the objective value of the train scheduling problem which provides us with an easy solution method in some special cases. The contrast in complexity to the analogous job shop scheduling problem is highlighted. Additionally, we prove the pseudo-polynomial solvability for a more general setting of the train scheduling problem.

1 Introduction

In this paper we consider a scheduling problem which is motivated by a railway application: the scheduling of trains on a single bi-directional track. This problem occurs in passenger transportation in rural areas or when scheduling freight trains, as outlined in [19] and references therein.

In its basic version, the *Single-Track-Train-Scheduling Problem (STTS)* reads as follows: we are given a single track, running from left to right, which has to be passed by P^l trains from the left and P^r trains from the right in the least time possible. The track is divided into several block sections, each block can be occupied by only one train at the same time. However, between the blocks we have stations which have unlimited capacity. Here trains can wait in order to let trains from the opposite direction pass.

Jonas Harbering
Georg-August-University of Göttingen
E-mail: jo.harbering@math.uni-goettingen.de

Abhiram Ranade
Indian Institute of Technology
E-mail: ranade@cse.iitb.ac.in

Marie Schmidt
Erasmus University Rotterdam
E-mail: schmidt2@rsm.nl

The translation to common machine scheduling terminology is quite straightforward: trains correspond to jobs, blocks correspond to machines, the block traversal time corresponds to processing time on a machine, and our objective is to minimize the makespan.

When translating the (STTS) to a machine scheduling problem, we obtain a special case of job-shop scheduling, since we have two subsets of jobs corresponding to trains coming from the left and trains coming from the right, which pass the blocks/machines in reverse order. This is called *job shop scheduling with counter-routes in the machine scheduling context. Furthermore, we have the requirement that a train ride on a block cannot be interrupted, this is referred to as* no preemption *in machine scheduling.* The (STTS) can hence be interpreted as a *job shop scheduling problem with two counter routes and no preemption* (abbreviated as $F^{\pm}||C_{\max}$ following common scheduling notation, see [9]), under the additional assumption that the processing time of a job on a machine is the same for all jobs. This relation is detailed in Section 2.2.

While $F^{\pm}||C_{\max}$ is strongly NP-hard for three machines already [9], the assumption that processing times depend only on the machines (or, to say it in the words of the train scheduling example: that block traversing times are the same for all trains) makes the problem considerably easier. In fact, we are going to show that for any fixed number of blocks, the problem can be solved polynomially in the number of trains and the maximal block length by dynamic programming.

For the case of three blocks and equal train numbers from both sides, as well as for some other special cases, we are even able to prove a closed-form expression for the minimal makespan.

The remainder of the paper is structured as follows. In Section 2 we give a short literature overview on relevant contributions in machine scheduling including some related complexity results and in single-track train scheduling. In Section 3 we formally introduce the problem. In Section 4 we derive a lower bound on the solution value. Using this bound, in Section 5 we discuss special cases where the lower bound can be reached. For the remaining cases we develop a dynamic programming approach in Section 6 and briefly discuss how constraints like limited station capacities and waiting times at stations can be included.

2 Related Problems

2.1 Train Scheduling

The general problem of scheduling or timetabling has many different facets. Since problems arising in train scheduling *on networks* are often of quite different nature, we concentrate on *single-track* train scheduling in this overview.

The problem of scheduling trains on a single track has attracted attention very early already. One of the first to lay a ground for such research is [13]. There, two way traffic systems, similar to the ones considered here, are analyzed. The main difference to our work is that they aim at determining the minimal number of trains that serve a certain train system ensuring periodic schedules. Subsequently, a model allowing for different train speeds and including delays is considered in [22]. There, train systems are analyzed in order to compute the average traveling time per train, depending on the number of trains with different priorities using the same track sequence. Also in [15], delays in train scheduling are considered. The authors aim at a tool which is suitable both for realtime decision support and for timetable evaluation. They also develop a non-linear mixed integer model to minimize the

average weighted travel time which takes into account several different timetabling aspects. Finally, they solve the developed model with a branch-and-bound approach. Similarily, [18] develop a detailed model for scheduling trains on a single track. In their model, the computation of the line capacity is reduced to computing the capacity only on a bottleneck segment. For this bottleneck segment they are able to determine the main relations between input parameters and the capacity of the line. In [25] a mixed integer linear programming model is stated in which trains may start at trains may only pass each other at stations. Different headways are considered for consecutive trains on tracks and at stations. In order to minimize the total travelling time, a branch-and-bound procedure is proposed. Using a similar model to [25], [8] propose a mixed integer linear model with varying departure and traveling times. Additionally, this model takes into account that headways between trains in the same direction are possibly shorter than headways for trains in different directions. They aim at minimizing the total arrival times of all trains at all stations. The model is then solved by a heuristic. Finally, [23] considers a very similar model (to [8,25]) with the objective of minimizing the arrival time of the last train at its destination, i.e. the makespan. Most constraints are adapted from [25] while some problem specific constrains are added in order to decrease computation time. The proposed model is then solved by a heuristic approach. A slightly different problem is discussed in [4]. Here, two trains in opposing directions are scheduled on a two-track segment. Now a part of one of the tracks fails. The question to be answered is how can trains be scheduled on the same track segment in opposing direction without deviating too much from the previous schedule.

See [23] for a broad overview on scheduling on a single bidirectional line and [7,10] for more general train scheduling surveys.

The models in all of the discussed works model real-world train scheduling constraints in varying degree of detail. Our simplified train scheduling problem could, e.g., be considered to be a special case of the models described in [7,8,23,25] and the solution methods described there for more general problems could also be applied to solve our special case. However, to the best of our knowledge, there is so far no complexity analysis for such problems. Hence, the possibility of solving these problems exactly in polynomial time, based, e.g., on combinatorial algorithms or linear programming, has not been ruled out yet. With this work, we aim to lay a ground for filling this gap.

Train scheduling is closely related to machine scheduling. In fact, even more general timetabling problems can be modeled using disjunctive graphs, which are also frequently used to model machine scheduling problems. See [3] for an introduction to this modeling approach and [5] for an application of it to timetabling.

In the next section, we detail how the (**STTS**) can be interpreted in a machine scheduling context and how complexity results from machine scheduling transfer to our problem.

2.2 Relation to Machine Scheduling

The problem (**STTS**) can be interpreted as a machine scheduling problem as follows: Let a set of machines $M_1, \ldots, M_n$ and a set of jobs $J^l_1, \ldots, J^l_{P^l}, J^r_1, \ldots, J^r_{P^r}$ be given. Then the aim is to feasibly schedule the jobs $J^l_1, \ldots, J^l_{P^l}$ on the sequence of machines $M_1 - \cdots - M_n$ and the jobs $J^r_1, \ldots, J^r_{P^r}$ on the sequence $M_n - \cdots - M_1$ while minimizing the makespan. Note that such a sequence of machines is called a *route* in machine scheduling.

The following conditions must hold for a schedule to be feasible: Jobs can only be processed by one machine at a time, machines can only process one job at a time and once the processing of a job on a machine has started, the job can not be interrupted. This problem is called the *job shop problem with two counter routes and no preemption* ($F^{\pm}//C_{max}$). This abbreviation corresponds to the usual three field representation which is used for classifying machine scheduling problems. It means that the problem is a flow shop problem (F) with two counter routes ($\pm$) and the objective is to minimize the makespan (C_{max}). See [9] for an extensive analysis of flow and job shop problems, including this problem.

In our problem, the trains are represented by the jobs and the blocks are represented by the machines. An important difference between (STTS) and $F^{\pm}//C_{max}$ is that in $F^{\pm}//C_{max}$ the processing times of different jobs on a machine can differ, while in (STTS) we assume all trains i to have the same traversal time p_i on each block. Hence, in the above-mentioned classification scheme for scheduling problems, (STTS) corresponds to the problem $F^{\pm}/p_{ij} = p_i/C_{max}$. Hereby, $p_{ij} = p_i$ means that the processing time p_{ij} of job $j \in \{J^l_1, \ldots, J^l_{P^l}, J^r_1, \ldots, J^r_{P^r}\}$ on machine $i \in \{M_1, \ldots, M_n\}$ is independent of the job. This restriction is well-studied in job-shop scheduling ([14,16,20]) but has to the extent of our knowledge not been considered in conjunction with counter routes.

The complexity of $F^{\pm}//C_{max}$ is well researched in the literature: For $n = 2$ the job shop problem with two counter routes and no preemption can be solved in polynomial time. E.g., [17] give an $O\left((P^l + P^r)log(P^l + P^r)\right)$ algorithm for a problem equivalent to $F^{\pm}n//C_{max}$. We conclude that the problem (**STTS**) can be solved in polynomial time for $n = 2$ and that this result would even hold if every train block combination had a different traversal time.

Still, already for $n = 3$ machines, job shop scheduling with two counter routes is NP-hard if processing times on the machines may differ for different jobs. This result immediately follows from the NP-hardness of flow shop scheduling on three machines with only one route [21]. The result even holds for the case that the number of jobs from boths sides are equal, i.e. if $P^l = P^r$, since the special case where all jobs from the right have 0 processing times is again equivalent to flow shop scheduling.

However, this complexity result does not carry over to the (STTS) where the block traversal times are equal for all jobs. Hence, the question of whether (STTS) can be solved in polynomial time or not is still open for the case of three or more blocks.

In [12] an upper bound on the objective value for job shop scheduling with two routes (not necessarily counter routes) on n machines is proven. Also [2] develop an upper bound for the counter routes problem on n machines. In [12] an earlier publication ([1]) on this topic is mentioned, however, we were not able to find this paper. See also the scheduling overview given by [9,24] for a collection of results on the $F^{\pm}//C_{max}$ problem.

3 Model

In this work we investigate the complexity of and approaches for solving the single track train scheduling problem. This problem is described as follows.

Let a linear graph $G = (V, B)$ with stations $V = \{s_1, \ldots, s_{n+1}\}$ and undirected edges, (called *blocks* in the remainder of this paper) $B = \{b_1 = (s_1, s_2), \ldots, b_n = (s_n, s_{n+1})\}$ be given. In order to facilitate the problem description, we imagine that the tracks are going from left to right.

We define station s_1 to be the leftmost station and station s_{n+1} to be the rightmost respectively.

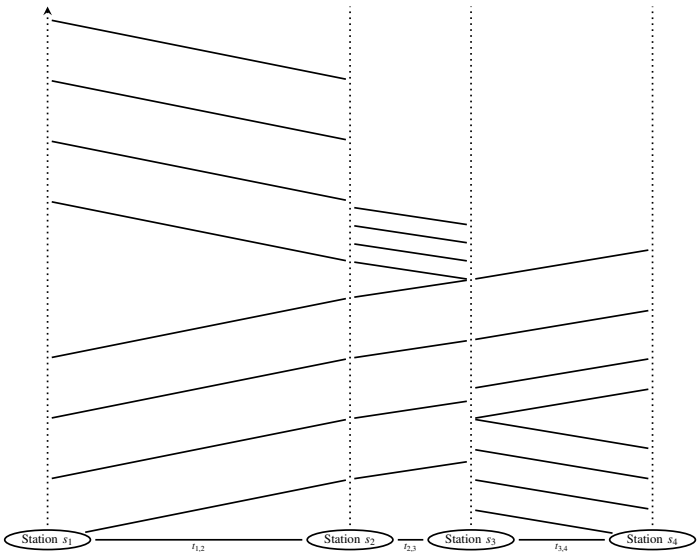

Fig. 1: Example for a space time diagram with $P^l = P^r = 4$ and $n = 4$

The traversal times of the blocks are given as $t_{i,i+1}$ for block $b_i = (s_i, s_{i+1})$ for all $i = 1, \ldots, n$. Waiting times at stations are neglected, i.e., in our model a train can pass a station without stopping. The number of trains traversing the graph from s_1 to s_{n+1} (or from left to right respectively) is specified by P^l and the number of trains traversing G in opposite direction is given as P^r. At any point in time there can at most be one train on each block. The capacity of the stations is not restricted. It means that any number of trains may be stored at any station. The described setting of course oversimplifies reality. However, the intent of this paper is to investigate to which extent the (STTS) is solvable in this very simple setting. Possible extensions and ways to make the model more realistic are discussed in Section 7.

With the above notation we can now formally state the problem of *Single Track Train Scheduling* (**STTS**).

(**STTS**)
Let the graph $G = (V, B)$ with traveling times, station and block capacities be given as above. Then the aim is to minimize the makespan for the traversal of G for P^l trains from left to right and P^r trains from right to left.

In order to illustrate scheduling strategies, we make use of space time diagrams. In such diagrams, the stops of the linear network are denoted on the horizontal axis, the time is given by the vertical axis. Lines in the diagram represent the traversal of trains. Figure 1 depicts such a diagram.

For explanatory purpose we introduce two terms. Suppose a train is at a certain point of the linear network, then we call the remainder of the network, which the train still has to traverse, its *remaining part* of the network. The time-wise distance between two trains (usually on a specific block) is called *headway*.

In the following we discuss how this problem is related to machine scheduling and to what extent complexity results from scheduling theory carry over to (STTS).

4 Lower Bound

In this section we discuss a lower bound on the objective value of the optimal solution for an instance I of (STTS). This value will help us to prove optimality of scheduling strategies for subsequent instances of (STTS).

The lower bound is given by

$$T^{lo}(I) = \max_{i=1,\dots,n} \left\{ \left(P^l + P^r\right) t_{i,i+1} + 2 \min \left\{ \sum_{j=1}^{i-1} t_{j,j+1}, \sum_{j=i+1}^{n} t_{j,j+1} \right\} \right\} \tag{1}$$

Let the block for which the maximum is attained in (1) be called the *bottleneck block*. Later on we will see that in many cases, a good scheduling strategy on the bottleneck block guarantees that the lower bound can be obtained as the makespan.

Lemma 1 *$T^{lo}(I)$ is a lower bound on the objective value of an instance I of (**STTS**).*

Proof Let us first consider one arbitrary block $b_{i,i+1}$. All trains must pass this block which makes up $(P^l + P^r)t_{i,i+1}$ of time. Additionally, the first train that passes the block needs some time to arrive at the block.

The minimum time for a train to arrive at $b_{i,i+1}$ is given by the minimum of the distances of the parts left and right of the block, i.e.

$$\min \left\{ \sum_{j=1}^{i-1} t_{j,j+1}, \sum_{j=i+1}^{n} t_{j,j+1} \right\}$$

Subsequently, all trains pass that block. Still the last train has to reach its destination. The time to reach the destination is again bounded below by the minimum of the distances of the parts left and right of the block, i.e.

$$\min \left\{ \sum_{j=1}^{i-1} t_{j,j+1}, \sum_{j=i+1}^{n} t_{j,j+1} \right\}$$

Summing up all parts we obtain for each block a value which gives a lower bound on the makespan. In particular, the maximal value of those is also a lower bound.

This lower bound will be of help to show that particular cases can be solved in polynomial time.

5 Special Cases

We will now direct to special cases for which we can specify scheduling strategies which allow to reach the lower bound on the makespan.

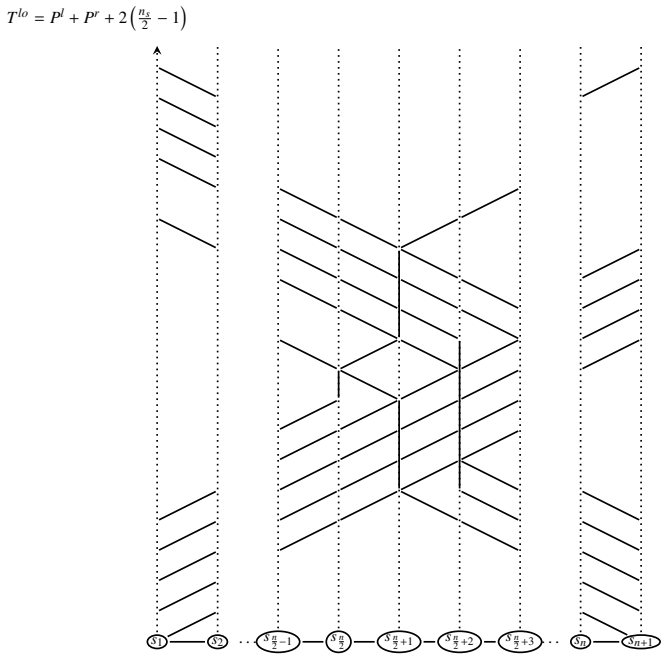

Fig. 2: Strategy for unit processing time and even number of blocks

5.1 Unit processing times and capacity restrictions at stations

Assume that all traversal times on all blocks are equal, i.e. $t_{i,i+1} = 1$ for all $i = 1, \ldots, n$. Under this condition we can find optimal schedules which do not need high station capacity at intermediate stations.

Theorem 1 *For instances of (**STTS**) with unit processing times, the makespan of an optimal schedule is given by $T^{lo}(I)$. This even holds if the capacity at all stations is bounded by three.*

Proof First, assume that no capacity restrictions are enforced. Two different cases will be considered. This is n is even and n is odd.

Consider the case where n is even. The following strategy provides an optimal schedule. See Figure 2 for a sketch of the schedule.

Let all trains from both sides start traversing the linear network at time 0 with a headway of one. If there is a conflict, i.e., if two trains would cross the same block at the same time, always let trains from the left precede those from the right, except for the case that the train from the left is the P^lth - in this case give the trains from the right precedence. After all trains from the left, except the P^lth, have traversed all blocks, all trains from the right traverse the remaining part of the network. After all trains from the right have passed, the P^lth train continues to station s_{n+1}.

The last train from the left traverses the block left of the central station $s_{\frac{n}{2}+1}$ between the first and the second train from the right. Then the train waits at the central station until all trains from the right have passed and finally heads for its destination s_{n+1}. We now have to show that indeed the lower bound $T^{lo}(I)$ is obtained with this strategy.

The first train from the left and from the right reach the central station $s_{\frac{n}{2}+1}$ at the same time $\frac{n}{2}$. After that all trains from the left traverse the central part heading for station $s_n + 1$, in particular the block right of the central station. Between time $\frac{n}{2}$ and time $\frac{n}{2} + P^l - 1$, the trains $1, 2, 3, \ldots, P^l - 1$ from the left cross the block right of the central station one after another and proceed to the right. In the meantime trains from the right have started from station s_{n+1} and traversed as many blocks as possible.

At station $s_{\frac{n}{2}+1}$ there is only one train waiting. At stations $s_{\frac{n}{2}+2}, s_{\frac{n}{2}+3}, \ldots, s_n$ there are two trains waiting as long as there are trains, the remaining trains wait at station s_{n+1}.

At time $\frac{n}{2} + P^l - 2$ the first train from the right starts traversing its remaining part of the linear network. Since then there are no more trains from the right waiting at station $s_{\frac{n}{2}+1}$ and the block $b_{\frac{n}{2}+1,\frac{n}{2}+2}$ is used by the penultimate train from the left for one more time unit, the next train heading for the left has a headway of two to the first train. Hence, in this gap, the last train from the left traverses the block $b_{\frac{n}{2},\frac{n}{2}+1}$ and thus waits at station $s_{\frac{n}{2}+1}$. Finally, all trains from the right traverse their remaining part of the network with a headway of one, including those which were waiting at the station s_{n+1}. Once the last train from the right has arrived at the central station $s_{\frac{n}{2}+1}$ (at time $\frac{n}{2} + P^l + P^r - 1$) both this train and the last train from the left finish their ride to their final station. This ride takes each train $\frac{n}{2}$ time units.

If we now consider block $b_{\frac{n}{2}+1,\frac{n}{2}+2}$ we find the following sequence of trains. Until time $\frac{n}{2} - 1$ there is no train. Then the first train from the right passes this block. After that all trains from the left but the last one traverse this block. Subsequently, all trains from the right but the first pass this block. Finally, the last train passing this block is the last train from the left. Then the block is empty until the last train from the left reaches its destination.

It is easy to see that no conflicts appear on the other blocks.

In order to determine the makespan of this schedule, we consider the block $b_{\frac{n}{2}+1,\frac{n}{2}+2}$. This block is empty until time $\frac{n}{2} - 1$. Afterwards, it is occupied during $P^l + P^r$ time units: Between time $\frac{n}{2} - 1$ and time $\frac{n}{2}$, the first train from the right passes the block. After that all trains from the left but the last one traverse the block until time $\frac{n}{2} + (P^l - 1)$. Subsequently, all trains from the right but the first pass the block until time $\frac{n}{2} + (P^l - 1) + (P^r - 1)$. Finally, the last train passing this block is the last train from the left which arrives at station $\frac{n}{2} + 2$ at time $\frac{n}{2} + P^l + P^r - 1$. Consequently, the last train from the right arrives at the leftmost station s_1 at time $\frac{n}{2} + (P^l - 1) + (P^r - 1) + \frac{n}{2} = P^l + P^r + 2\left(\frac{n}{2} - 1\right) = T^{lo}(I)$ and the last train from the left arrives at the rightmost station s_{n+1} at time $\frac{n}{2} + P^l + P^r + (\frac{n}{2} - 1) = P^l + P^r + 2\left(\frac{n}{2} - 1\right) = T^{lo}(I)$.

To see the second part of the lemma, note that at each station at each point in time there are at most two trains from the right and one from the left. Hence this schedule can be operated with a maximal station capacity of three.

Now assume that n is odd. The strategy in this case is the following. Let all trains from both side traverse the linear network with a headway of one. If there is any conflict between two trains from different directions always let trains from the left precede.

Here we find that the central block $b_{\frac{n+1}{2},\frac{n+1}{2}+1}$ is reached by a train from each side at time $\frac{n-1}{2}$. Subsequently, all trains from the left side traverse the central block followed by all trains from the right side with headway one. Note that this is only possible since at any station from $s_{\frac{n+1}{2}+1}$ to the right there are two trains waiting. Finally, the last train has traversed the central block at time $\frac{n-1}{2} + P^l + P^r$ and it takes another $\frac{n-1}{2}$ time units until it reaches the final station s_1. Hence the makespan equals $P^l + P^r + 2\left(\frac{n-1}{2}\right)$ which equals the lower bound $T^{lo}(I)$.

To see the second part of the theorem, note that at no point in time there are more than two trains waiting at a station and a third one passing in opposing direction. Hence a maximal needed capacity of three is obtained.

If, additionally, the number of trains from left to right and from right to left are equal, even lower station capacity is sufficient to obtain an optimal solution.

Lemma 2 *For instances of (STTS) with unit processing times and $P^l = P^r$, the makespan of an optimal schedule is given by $T^{lo}(I)$. This even holds if the capacity at all stations is bounded by two.*

Proof Denote $P := P^l = P^r$. Again we separate the analysis into the cases where n is even and n is odd. First assume n is even.

The strategy consists in scheduling all trains from the left and the right at the same time with a headway of two. Then, since n is even two passing trains always meet at a station and never face a conflict on the tracks, hence, no train has to wait at intermediate stations. Consequently, the makespan is $n + 2(P-1) = 2P + 2\left(\frac{n}{2} - 1\right) = T^{lo}(I)$.

In case n is odd, let the first train from the left start at time 0 and again let subsequent trains from the left keep a headway of two to the preceding train. Furthermore, let the first train from the right start at time 1 and let subsequent trains from the right keep a headway of two to the preceding train as well. Then, two passing trains always meet at a station and never face a conflict on the tracks, hence, no train has to wait at intermediate stations. Consequently, the last train from the right arrives at time $1+n+2(P-1) = 2P+2\left(\frac{n-1}{2}\right) = T^{lo}$, the last train from the left arrives at time $n + 2(P-1) = 2P + 2\left(\frac{n-1}{2}\right) - 1 < T^{lo}$.

Note that in both cases, no trains have to wait and that at most two trains are passing each other in stations. Hence, the capacity needed equals two.

We conclude this section with a remark on the situation with unit capacity which also holds if block lengths are not equal. In this section, the lower bound T^{lo} cannot be reached. Since trains cannot pass each other, the optimal strategy simply consists of letting the trains from one side pass first, and then the trains from the other side. This leads to the following lemma.

Lemma 3 *If the station capacity is restricted to one, the makespan of an optimal schedule is given by* $2\sum_{i=1}^{n} t_{i,i+1} + (P^l + P^r - 2)\max_{i=1}^{n} t_{i,i+1}$.

5.2 Restricting the Number of Blocks

In the following we consider arbitrary track lengths, but restrict the number of blocks on the track. From the machine scheduling analysis we know that the case $n = 2$ is easily solvable. Hence, we direct to the case where $n = 3$. Take Figure 3 as an illustration of the considered case.

Here, we can find a polynomially solvable case which is given by the following theorem.

Theorem 2 *For instances I of (**STTS**) with three blocks and $P = P^l = P^r$, the makespan of an optimal schedule is given by $T^{lo}(I)$.*

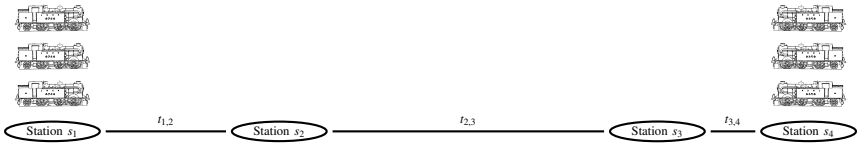

Fig. 3: Example setting for $n = 3$

Proof $T^{lo}(I)$ equals the largest of the three bounds $\overbrace{2Pt_{12}}^{(1)}$, $\overbrace{2Pt_{23} + 2\min\{t_{12}, t_{34}\}}^{(2)}$, $\overbrace{2Pt_{34}}^{(3)}$.

Suppose bound (2) is largest. Then, assume wlog. that $t_{12} \geq t_{34}$. We first construct a schedule with makespan T^{lo} blockwise, then we show its validity. The first half of the schedule is composed as follows. On block b_{12}, we have P trains leaving consecutively (i.e., with headway t_{12}) to the right. Block b_{23} is initially unused until time t_{34}. After that P trains go alternately and consecutively: first to left and then to right until all trains have passed this block at time $t_{12} + 2Pt_{23}$. On block b_{34}, starting at time zero, trains go left consecutively.

The second half of the schedule is described backwards from the time $T^{lo}(I)$. On block b_{12} we have left-going trains arriving consecutively at station s_1 (i.e., they arrive at times T^{lo}, $T^{lo} - t_{12}$, $T^{lo} - 2t_{12}$,...). Block b_{23} is unused for the last duration of t_{34}. Before that we have alternately moving trains, the last rightward, the second last leftward and so on, as described above. On block b_{34} we have right-going trains arriving consecutively at times T^{lo}, $T^{lo} - t_{34}$, $T^{lo} - 2t_{34}$,... at station s_4. Note that on blocks b_{12} and b_{34} there might be unused times around the middle of the schedule.

We now argue that this is a valid schedule, i.e.,

- no two trains are in the same block at any time and
- no train is scheduled to depart from a station before it has arrived there.

We only discuss this for the first half of the schedule; the second half is analogous. The first half is $Pt_{23} + t_{34}$ long. Clearly the P movements scheduled for block b_{23} fit in this duration, after the initial inactive period of length t_{34}. Because bound (2) is largest we know that $Pt_{12}, Pt_{34} \leq Pt_{23} + t_{34}$. Thus the P rightward (leftward) movements on block b_{12} (b_{34}) can also be accommodated, and there are no two trains at the same block at any time.

Next, as described in the schedule above, the ith rightward train must leave block b_{12} at time it_{12} and enter block b_{23} at $t_{34} + (2i - 1)t_{23}$. But since $t_{12} \leq t_{23} + \frac{t_{34}}{P}$, we have that

$$it_{12} \leq it_{23} + i\frac{t_{34}}{P} \leq (2i - 1)t_{23} + t_{34}.$$

Thus, the train is scheduled to enter block b_{23} only after leaving block b_{12}.

Likewise, the ith leftward train is scheduled to leave block b_{34} at it_{34} and to enter block b_{23} at $(2i - 2)t_{23} + t_{34}$. This is possible since it holds that

$$it_{34} = t_{34} + (i - 1)t_{34} \leq t_{34} + (i - 1)t_{12} \leq t_{34} + (i - 1)(t_{23} + \frac{t_{34}}{P}) \leq t_{34} + (2i - 2)t_{23}.$$

The last inequality in this is derived as follows. Note first that if $P = 1$ then $i = 1$ and the proof is direct. Hence consider $P > 1$. We know that $Pt_{34} \leq Pt_{23} + t_{34}$ and thus $t_{23} \geq t_{34}(P - 1)/P$. This results $t_{23} \geq \frac{t_{34}}{P}$ for $P > 1$.

Next suppose bound (1) is the largest. For ease of argument, we increase t_{23} until bound (2) equals bound (1). Then, we construct the same schedule as given above. This schedule can easily be transferred to a schedule for lower values of t_{23}. In the schedule, certain time

intervals are reserved for movements on block b_{23}. Of each such interval we only use a subinterval of length equal to the original value of t_{23}. This clearly does not change the makespan and maintains validity, i.e. trains use distinct time intervals on each block and are still in order in which they appear in each block.

The case of bound (3) being the largest is equivalent to the case of bound (1) being largest.

5.3 Can the lower bound $T^{lo}(I)$ always be achieved?

The following example shows a case for $n = 5$ in which the lower bound $T^{lo}(I)$ can not be achieved.

Example 1 Assume $P^l = P^r = 2$ and $t_{1,2} = t_{5,6} = 10$, $t_{2,3} = t_{4,5} = 3$ and $t_{3,4} = 4$. Then $T^{lo}(I) = (P^l + P^r)t_{3,4} + 2(t_{1,2} + t_{2,3}) = 42$ and the bottleneck is block $b_{3,4}$. Figure 4 shows an optimal scheduling strategy. In this strategy there is an unavoidable gap of 2 time units on the bottleneck block between the first and the second train from the right. Due to this gap, the makespan of this solution is $44 = T^{lo} + 2$.

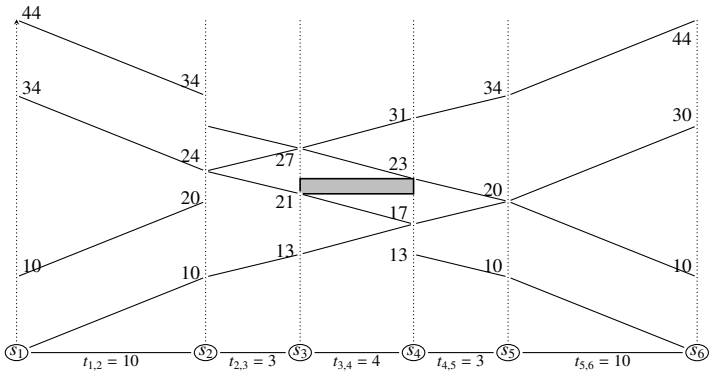

Fig. 4: Sketch for scheduling strategy for Example 1

As a general observation, we conclude that in order to reach the lower bound T^{lo}, we must be able to schedule the trains on the bottleneck block(s) without leaving a gap.

Observation 3 *If there exists a* $k = 1, 2, \ldots, 2P$ *such that at time* $\min\left\{\sum_{j=1}^{i-1} t_{j,j+1}, \sum_{j=i+1}^{n} t_{j,j+1}\right\} + (k-1)t_{jj+1}$ *less than k trains have reached the bottleneck block, there does not exist a schedule which achieves makespan* T^{lo}.

Note that this can occur, even if the longest block is the bottleneck block, as can be seen in the following example.

Example 2

Let $P^l = P^r = 2$ and $(t_{1,2}, t_{2,3}, \ldots, t_{8,9}) = (5, 2, 1, 1, 1, 1, 4)$. Then the bottleneck block is obtained as $b_{1,2}$ with $T^{lo} = 20$. When scheduling, we see that once the second train from left has passed the bottleneck, the first train from the right has not yet reached the bottleneck. Hence the lower bound is not reached. See Figure 5 where the scheduling strategy is depicted.

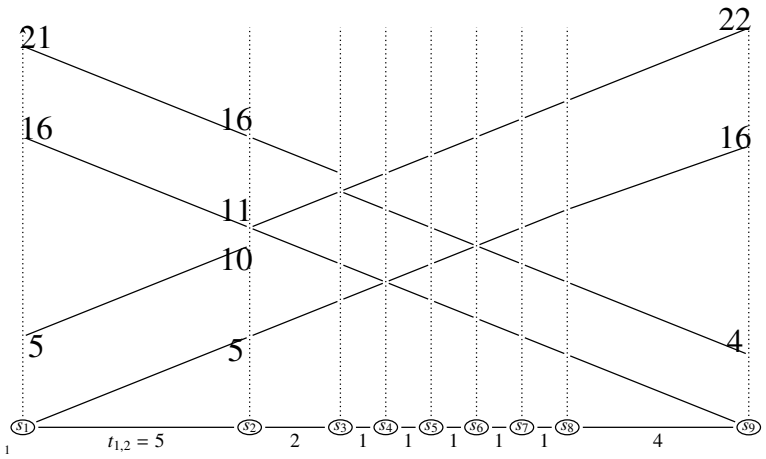

Fig. 5: Lower bound T^{lo} not reached even though longest block is bottleneck

6 Dynamic Programming

In this section we develop a dynamic programming formulation for the problem (STTS) with pseudo-polynomial running time.

Lemma 4 *If block lengths are integer, (STTS) can be solved as a shortest-path problem in an acyclic graph.*

Proof Denote by $L := \sum_{i=1}^{n} t_{i,i+1}$ the total length of the track considered. We divide the tracks into L subblocks c_k of lengths 1 with substations s'_k at every end of a subblock. We denote by I the index set of the substations which are stations and by $I(b_j)$ the index set of the substations corresponding to block b_j, including the stations at both ends of the block. Note that each substation index i belonging to a station s_j (except for $j = 1$ and $j = n + 1$) is contained in two sets $I(b_{j-1})$ and $I(b_j)$.

States: The nodes of the graph in which we are going to formulate the shortest path problem denote the states of the problem. Each state is represented by a tuple X with entries

$$X := (x_0^l/x_0^r, x_1^l/x_1^r, \ldots, x_L^l/x_L^r),$$

also written as

$$X := \frac{x_0^l \,|\, x_1^l \,|\, x_2^l \,|\, \ldots \,|\, x_{L-1}^l \,|\, x_L^l}{x_0^r \,|\, x_1^r \,|\, x_2^r \,|\, \ldots \,|\, x_{L-1}^r \,|\, x_L^r}. \tag{2}$$

Herein, x_i^l represents the number of trains from the left which have passed or reached substation s'_i already, and x_i^r represents the number of trains from the right which have passed or reached substation s'_i already.

For $n \in \mathbf{N}$ we denote $[n] := \{0, 1, 2, \ldots, n\}$.

Feasible states: Since we have P^l trains from the left and P^r trains from the right, $X^l := (x_0^l, x_1^l, \ldots, x_L^l) \in [P^l]^{L+1}$ and $X^r := (x_0^r, x_1^r, \ldots, x_L^r) \in [P^r]^{L+1}$. However, not all vectors $Y \in [P^l]^{L+1} \times [P^r]^{L+1}$ define feasible states of the problem:

- At the beginning, all trains passing from left are on the left of the track, i.e., $x_0^l = P^l$ for all feasible states X, analogously $x_L^r = P^r$ for all feasible states X.

– Since trains pass from left to right, we have $x_i^l \geq x_{i+1}^l$ for all $i = 0, \ldots, L$, analogously $x_i^r \leq x_{i+1}^r$.
– Only one train can be on a block at a time.

$$(\max_{i \in I(b_j)} x_i^l - \min_{i \in I(b_j)} x_i^l) + (\max_{i \in I(b_j)} x_i^r - \min_{i \in I(b_j)} x_i^r) \leq 1 \quad \text{for all } j = 1, \ldots, n$$

Only nodes representing feasible states will be introduced.
Transitions/arcs Two nodes $(X, \hat{X})$ are connected by a directed arc of length 1, if state $\hat{X}$ can be reached from state X in time 1. This is the case, if

– $x_i^k \leq \hat{x}_i^k$ for all $i = 0, \ldots, L$, $k \in \{l, r\}$, i.e., over time, more and more trains pass every substation of the network.
– $\sum_{i \in I(b_j)} (\hat{x}_i^l + \hat{x}_i^r) \leq \sum_{i \in I(b_j)} (x_i^l + x_i^r) + 1$ for all $j = 1, \ldots, n$, i.e., on each block there is at most one train running and it advances at most one substation.
– If for two substations s_i' and s_{i+1}', belonging to the same block b_j, $x_i^l = x_{i+1}^l + 1$, then it follows $\hat{x}_{i+1}^l = x_{i+1}^l + 1$, unless s_i' is a station. Analogously, $x_i^r = x_{i-1}^r + 1$, then $\hat{x}_{i-1}^r = x_{i-1}^r + 1$, unless s_{i+1}' is a station. This models that trains cannot stop between the stations.

These conditions ensure that the graph is acyclic.
Correspondence of solutions to paths There is a one-to-one correspondence between feasible schedules for the (STTS) and paths from node

$$\begin{array}{|c|c|c|c|c|c|} \hline P^l & 0 & 0 & \ldots & 0 & 0 \\ \hline 0 & 0 & 0 & \ldots & 0 & P^r \\ \hline \end{array}$$

to node

$$\begin{array}{|c|c|c|c|c|c|} \hline P^l & P^l & P^l & \ldots & P^l & P^l \\ \hline P^r & P^r & P^r & \ldots & P^r & P^r \\ \hline \end{array}$$

where the length of the path represents the time duration to execute the solution.

Hence, an optimal solution to (STTS) corresponds to a shortest path is the graph.

Lemma 5 *If block lengths are integer, the graph described in Lemma 4 has* $O\left(\binom{P^l+n}{n}\binom{P^r+n}{n} \max\{t_{i,i+1}\}\right)$ *nodes and* $O\left(3^n\binom{P^l+n}{n}\binom{P^r+n}{n} \max\{t_{i,i+1}\}\right)$ *arcs.*

Proof As stated in the proof of Lemma 4, the nodes of the graph correspond to feasible states X (as described in (2)). We now make an attempt on bounding the number of feasible states from above.

We denote by y_j^l the number of trains which have passed the full block b_j from left to right. y_j^r analogously denotes the number of trains which have passed the full block b_j from right to left. I.e, $y_j^l = x_{i_{\text{last}^j}}^l$ and $y_j^r = x_{i_{\text{first}^j}}^l$, where i_{first^j} and i_{last^j} denote the first and the last index in the set $I(b_j)$, respectively.

For every pair (y_j^l / y_j^r) there are $2t_{j,j+1} - 1$ feasible combinations of subindices:

$$\begin{array}{|c|c|c|c|c|} \hline y_j^l = x_{i_{\text{first}^j}}^l & x_{i_{\text{first}^j}+1}^l & x_{i_{\text{first}^j}+2}^l & \ldots & x_{i_{\text{last}^j}-1}^l & y_{j+1}^l = x_{i_{\text{last}^j}}^l \\ \hline y_j^r = x_{i_{\text{first}^j}}^r & x_{i_{\text{first}^j}+1}^r & x_{i_{\text{first}^j}+2}^r & \ldots & x_{i_{\text{last}^j}-1}^r & y_{j+1}^r = x_{i_{\text{last}^j}}^r \\ \hline \end{array},$$

namely

$$\begin{array}{|c|c|c|c|c|c|} \hline y_j^l & y_j^l - 1 & y_j^l - 1 & \ldots & y_j^l - 1 & y_j^l - 1 \\ \hline y_j^r & y_j^r & y_j^r & \ldots & y_j^r & y_j^r \\ \hline \end{array}, \begin{array}{|c|c|c|c|c|c|} \hline y_j^l & y_j^l & y_j^l - 1 & \ldots & y_j^l - 1 & y_j^l - 1 \\ \hline y_j^r & y_j^r & y_j^r & \ldots & y_j^r & y_j^r \\ \hline \end{array}, \ldots, \begin{array}{|c|c|c|c|c|c|} \hline y_j^l & y_j^l & y_j^l & \ldots & y_j^l & y_j^l \\ \hline y_j^r & y_j^r & y_j^r & \ldots & y_j^r & y_j^r \\ \hline \end{array},$$

and

$$\frac{y_j^l}{y_j^r-1}\Big|\frac{y_j^l}{y_j^r-1}\Big|\frac{y_j^l}{y_j^r-1}\Big|\dots\Big|\frac{y_j^l}{y_j^r-1}\Big|\frac{y_j^l}{y_j^r}, \dots, \frac{y_j^l}{y_j^r-1}\Big|\frac{y_j^l}{y_j^r}\Big|\frac{y_j^l}{y_j^r}\Big|\dots\Big|\frac{y_j^l}{y_j^r}\Big|\frac{y_j^l}{y_j^r}, \frac{y_j^l}{y_j^r}\Big|\frac{y_j^l}{y_j^r}\Big|\frac{y_j^l}{y_j^r}\Big|\dots\Big|\frac{y_j^l}{y_j^r}\Big|\frac{y_j^l}{y_j^r},$$

(one case is listed twice here), if $y_j^l, y_j^r > 0$. If one or both values are 0, there are less.

Furthermore, there are

$$\sum_{i_n=0}^{P^l}\sum_{i_{n-1}=0}^{i_n}\sum_{i_{n-2}=0}^{i_{n-1}}\dots\sum_{i_1=0}^{i_2}1\cdot\sum_{i_n=0}^{P^r}\sum_{i_{n-1}=0}^{i_n}\sum_{i_{n-2}=0}^{i_{n-1}}\dots\sum_{i_1=0}^{i_2}1$$

feasible combinations of the y-values. Each nested sum

$$\sum_{i_n=0}^{P^l}\sum_{i_{n-1}=0}^{i_n}\sum_{i_{n-2}=0}^{i_{n-1}}\dots\sum_{i_1=0}^{i_2}1=\binom{P^l+n}{n}$$

then gives a binomial coefficient, see [6] for an explanation. This leads to $O\left(\binom{P^l+n}{n}\binom{P^r+n}{n}\max\{t_{i,i+1}\}\right)$ nodes.

To count the number of arcs, let us consider possible successors $\hat{X}$ of node X in the graph. For each block, there are at most three different possibilities: either a train is moving from left to right, a train is moving from right to left, or no train is moving at all on that block (as formalized when defining the arcs above). Since we have n blocks, each node X has hence at most 3^n successors (and most will have much less). Thus, there are at most $O\left(3^n\binom{P^l+n}{n}\binom{P^r+n}{n}\max\{t_{i,i+1}\}\right)$ arcs in the graph.

Lemma 6 *The (STTS) can be solved in* $O\left(3^n\binom{P^l+n}{n}\binom{P^r+n}{n}\max\{t_{i,i+1}\}\right)$.

Proof Since the described graph is a directed and acyclic graph, we can find a shortest path in linear time in the number of edges, see, e.g., [11].

Of course, for large numbers of blocks n this is impractical and we would probably be better off using an integer programming approach as described, e.g., in [7]. However, from a theoretical point of view, the following conclusion is interesting:

Corollary 1 *For a fixed number of blocks, the (STTS) can be solved in pseudo-polynomial time, i.e., it is polynomial in* P^l, P^r *and* $\max_j t_{j,j+1}$.

Possible Extensions In the basic version of the (STTS) we assumed that stations have an unlimited number of tracks. However, limited station capacity could easily be taken into account in the dynamic programming approach. Note that for each state X of the dynamic program, the number of trains currently halting at station s_j is

$$H_j(X) := \underbrace{x^l_{i^j_{first}} - x^l_{i^j_{first}+1}}_{\text{trains from the left}} + \underbrace{x^r_{i^j_{first}} - x^r_{i^j_{first}-1}}_{\text{trains from the right}}.$$

Hence, track capacity restrictions at stations can be included by creating only the nodes which satisfy $H_j(X) \leq$ capacity of station s_j and the corresponding arcs.

We can also include simple vehicle scheduling constraints in the model. I.e., consider the situation that the trains go back and forth and traverse the track several times during a day. In this case, we would denote by P^l the number of departures from left and by P^r the number of departures from right. Introducing only nodes X which fulfill

$$x_2^l - x_1^r \leq \text{initial number of trains on left}$$

we would make sure that the number of trains which have left the left station never exceeds the number of trains which have arrived there by more than the initial number of trains on the left. It makes sure that there is always a train to leave when there is a scheduled departure. An analogous constraint can be added for the station on the right.

A similar approach could be taken when introducing simple 'balance' constraints. Particularly when transporting passengers, it could be considered unbalanced and unfair, if a large number of trains from the left could pass the full track unhindered while all trains from the right had to wait. To avoid this problem, we could, similarly to the approach described above, exclude all nodes modeling states where, e.g., x_L^l and x_0^r are too different, w.r.t. to the above stated balancedness, or where the difference between x_i^l and x_i^r at some intermediate substation s_i' is too big.

Note that all extensions described so far lead to a smaller network and hence would speed-up the dynamic programming approach.

Other extensions, like minimal waiting times at stations or different train types with different speed profiles could also be included, but *more* states (that is: more nodes) would be needed. However, the general theoretic result of pseudopolynomial solvability for a fixed number of blocks would remain valid also in these cases.

More sophisticated extensions like, e.g., periodicity or routing in stations seem to be out of the scope of this approach.

7 Conclusion

In this paper we studied a basic version of the Single Track Train Scheduling Problem, which can be considered a special case of job shop scheduling with two counter routes and no preemption. Our special case, furthermore, restricts the processing time of a job on a machine to be the same for all jobs.

We were able to prove a lower bound on the minimum makespan of the (STTS) and identify several special cases where this lower bound is tight.

In particular, we found that, although the job shop scheduling problem with two counter routes, no preemption, three machines and equal number of jobs from both sides is strongly NP-hard, that, if we have the additional requirement that the processing times of all jobs on each machine are equal (as it is the case in the (STTS)), we can specify scheduling strategies which reach a makespan equal to the lower bound.

Furthermore, we showed that for any fixed number of blocks (STTS) can be solved in pseudo-polynomial time.

This result can also be generalized for some less basic versions of single track train scheduling which even have the potential to speed-up the algorithm considerably. However, the computation time of our approach increases exponentially in the number of blocks, hence the result is more of a theoretical value and most likely not applicable in real-world train scheduling where the number of blocks is typically large.

In the general cases the lower bound may help to prove optimality or give a good estimate of distance to optimality when applying heuristic solution methods.

To what extent the results in this work can be generalized and the lower bound can be improved is currently under research.

References

1. Babushkin, A., Bashta, A., Belov, I.: Scheduling for the problem with oppositely directed routes. Kibernetika **7**, 130–135 (1974)
2. Babushkin, A., Bashta, A., Belov, I.: Scheduling for problems of counterroutes. Cybernetics and Systems Analysis **13**(4), 611–617 (1977)
3. Balas, E.: Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. Operations research **17**(6), 941–957 (1969)
4. Brucker, P., Heitmann, S., Knust, S.: Scheduling railway traffic at a construction site. Springer (2005)
5. Burdett, R.L., Kozan, E.: A disjunctive graph model and framework for constructing new train schedules. European Journal of Operational Research **200**(1), 85–98 (2010)
6. Butler, S., Karasik, P.: A note on nested sums. Journal of Integer Sequences **13**(2), 3 (2010)
7. Cacchiani, V., Toth, P.: Nominal and robust train timetabling problems. European Journal of Operational Research **219**(3), 727–737 (2012)
8. Castillo, E., Gallego, I., Ureña, J.M., Coronado, J.M.: Timetabling optimization of a single railway track line with sensitivity analysis. TOP **17**(2), 256–287 (2009)
9. Chen, B., Potts, C., Woeginger, G.: A Review of Machine Scheduling: Complexity, Algorithms and Approximability. Handbook of Combinatorial Optimization (1998)
10. Cordeau, J.F., Toth, P., Vigo, D.: A survey of optimization models for train routing and scheduling. Transportation Science **32**(4), 380–404 (1998)
11. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., et al.: Introduction to algorithms, vol. 2. MIT press Cambridge (2001)
12. Dushin, B.: An algorithm for the solution of the two-route johnson problem. Cybernetics and Systems Analysis **24**(3), 336–343 (1988)
13. Frank, O.: Two-way traffic on a single line of railway. Operations Research **14**(5), 801–811 (1966)
14. Gonzalez, T.: Unit execution time shop problems. Mathematics of Operations Research **7**(1), 57–66 (1982)
15. Higgins, A., Kozan, E., Ferreira, L.: Optimal scheduling of trains on a single line track. Transportation Research Part B: Methodological **30**(2), 147–161 (1996)
16. Hromkovič, J., Mömke, T., Steinhöfel, K., Widmayer, P.: Job shop scheduling with unit length tasks: bounds and algorithms. Algorithmic Operations Research **2**(1) (2007)
17. Jackson, J.: An extension of johnson's result on job lot scheduling. Naval Research Logistics Quarterly (1956)
18. Janić, M.: Single track line capacity model. Transportation Planning and Technology **9**(2), 135–151 (1984)
19. Kraay, D., Harker, P.T., Chen, B.: Optimal pacing of trains in freight railroads: Model formulation and solution. Operations Research **39**(1), pp. 82–99 (1991)
20. Lenstra, J.K., Kan, A.R.: Computational complexity of discrete optimization problems. Annals of Discrete Mathematics **4**, 121–140 (1979)
21. M. R. Garey, D.S.J., Sethi, R.: The Complexity of Flowshop and Jobshop Scheduling. Mathematics of Operations Research (1976)
22. Petersen, E.: Over-the-road transit time for a single track railway. Transportation Science **8**(1), 65–74 (1974)
23. Rahman, S.A.A.: Freight train scheduling on a single line network. Ph.D. thesis, The University of New South Wales (2013)
24. Sevast'janov, S.: On some geometric methods in scheduling theory: a survey. Discrete Applied Mathematics **55**(1), 59 – 82 (1994)
25. Zhou, X., Zhong, M.: Single-track train timetabling with guaranteed optimality: Branch-and-bound algorithms with enhanced lower bounds. Transportation Research Part B: Methodological **41**(3), 320–341 (2007)