Complexity, Bounds and Dynamic Programming Algorithms for Single Track Train Scheduling

Jonas Harbering (jo.harbering@math.uni-goettingen.de)
Georg-August-University of Göttingen

Abhiram Ranade (ranade@cse.iitb.ac.in)
Indian Institute of Technology

Marie Schmidt (schmidt2@rsm.nl)
Erasmus University Rotterdam

Oliver Sinnen (o.sinnen@auckland.ac.nz)
University of Auckland

October 24, 2017

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. Traversal times of the trains on the blocks between the stations only depend on the block lengths but not on the train. This problem is a special case of minimizing the makespan in job shop scheduling 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. Additionally, we prove that for a fixed number of blocks the problem can be solved in pseudo-polynomial time.

Keywords: *Machine Scheduling, Train Scheduling, Complexity Analysis, Counter Routes*

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 Kraay et al. (1991) 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 n^l trains from the left and n^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. See Figure 1 for an example of the problem (STTS).

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

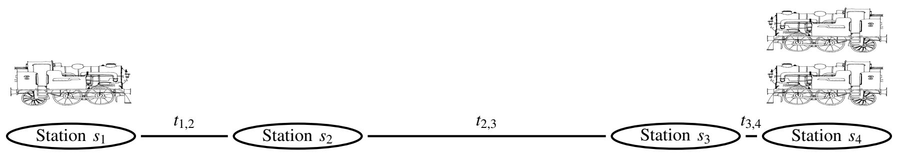

Figure 1: Example setting for a track with 3 block sections, $n^l = 1$ trains from left to right and $n^r = 2$ trains from right to left. Note that the tracks have capacity one and the stations have unlimited capacity.

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 Chen et al. (1998)), 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 2.3.

While $F^{\pm}||C_{\max}$ is strongly NP-hard for three machines already Chen et al. (1998), the assumption that processing times depend only on the machines and not on the jobs (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 single-track train scheduling and in machine scheduling. In Section 3 we formally introduce the problem. In Section 4 we prove complexity results. To this end, we first develop an upper and a lower bound on the makespan in Section 4.1, in Section 4.2 we discuss some special cases in which a schedule with makespan equal to the lower bound can be found, give an example where the minimal makespan is higher than the lower bound, and provide a pseudo-polynomial algorithm for the case of three blocks. In Section 4.3 we show that for any fixed block length the problem is solvable in pseudo-polynomial time by dynamic programming. Finally, a conclusion and discussion is given in Section 5.

2 Related Work

2.1 Literature on Train Scheduling

The general problem of scheduling or timetabling has many different facets. Since the challenges arising in train scheduling *on networks* are quite different from *single-track* train scheduling, we concentrate on the latter in this overview and refer the reader to Cacchiani and Toth (2012); Cordeau et al. (1998) for an overview on timetabling in networks.

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 Frank (1966). There, two way traffic systems, similar to the ones considered here, are analyzed. The main difference to our work is that Frank (1966) aims 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 Petersen (1974). 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 Higgins et al. (1996), 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. For the same model, Higgins and Ferreira (1997) present and compare many different heuristic algorithms. Similarily, Janić (1984) 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 Zhou and Zhong (2007) a mixed integer linear programming model is stated in which 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 as in Zhou and Zhong (2007), Castillo et al. (2009) 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, Rahman (2013) considers a very similar model (to Castillo et al. (2009); Zhou and Zhong (2007)) with the objective of minimizing the arrival time of the last train at its destination, i.e., the makespan. Most constraints are adapted from Zhou and Zhong (2007) 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 Brucker et al. (2005). Here, two trains in opposing directions are scheduled on a two-track segment. Then, 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. In Sotskov and Gholami (2012) a single track train scheduling problem is transferred to a corresponding machine scheduling problem with due date related objective. For this problem the classical shifting bottleneck algorithm from machine scheduling with makespan objective is adapted. In other publications based on single track scheduling, the control signalling system is studied in Bersani et al. (2015), the passenger demand receives a focus by modeling fuzzy demand in Yang et al. (2009), vehicle schedules on a single line receive including passenger demand consideration receive a focus in Mesa et al. (2014) and stochastic demand Yang et al. (2010) and train rescheduling algorithms under disruptions with a focus on learning procedures are discussed in Šemrov et al. (2016).
See Rahman (2013) for a broad overview on scheduling on a single bidirectional line and Cacchiani and Toth (2012); Cordeau et al. (1998) 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 Borndörfer et al. (2014); Cacchiani and Toth (2012); Castillo et al. (2009); Rahman (2013); Zhou and Zhong (2007) 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 Balas (1969) for an introduction to this modeling approach and Chen et al. (1998) for an application of it to timetabling.

Apart from the relation to other train scheduling as discussed above, the problem addressed in this paper is also related to ship canal scheduling, which look at in the next section.

2.2 Literature on Ship Canal Scheduling

A different scheduling application with similar characteristics is the scheduling of ships in a canal. In Lübbecke (2015); Lübbecke et al. (2014) results on bidirectional ship canal scheduling with an application for the Kiel canal are presented. Some more theoretical results are given in Disser et al. (2015). Ship canals can usually be traveled in both directions while there are sidings at fixed points which allow two ships to pass each other. In comparison to the problem (STTS), many more details are considered in the mentioned papers on ship canal scheduling. E.g., ships which go in the same direction are allowed smaller headway than ships in opposing direction. Additionally, if two ships are small enough, they can even pass each other with neither of them halting in a siding. For each pair of two ships a compatibility relationship is defined, stating whether those two ships can pass each other.

Since in our problem (STTS) the compatibility relation states that no two trains can pass each other on any track segment, which is a simplification to the problem from Lübbecke (2015), the NP-hardness result from Lübbecke (2015) cannot be carried over. Lübbecke et al. (2014) show how to successfully apply solutions for the bidirectional traffic scheduling problem to practically relevant situations.

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.

2.3 Relation to Machine Scheduling

The problem (STTS) can be interpreted as a machine scheduling problem as follows: Let a set of machines $M_1, \dots, M_m$ and a set of jobs $J^l_1, \dots, J^l_{n^l}, J^r_1, \dots, J^r_{n^r}$ be given. The objective is to feasibly schedule the jobs $J^l_1, \dots, J^l_{n^l}$ on the sequence of machines $M_1 - \dots - M_m$ and the jobs $J^r_1, \dots, J^r_{n^r}$ on the sequence $M_m - \dots - M_1$ 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 cannot be interrupted. This problem is called the *job shop scheduling 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 scheduling problem (F) with two counter routes ($\pm$) and the objective is to minimize the makespan (C_{max}). See Chen et al. (1998) for an extensive analysis of flow and job shop scheduling problems, including $F^{\pm}//C_{max}$. In (STTS), 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 $j \in \{J^l_1, \dots, J^l_{n^l}, J^r_1, \dots, J^r_{n^r}\}$ on machine $i \in \{M_1, \dots, M_m\}$ is independent of the job. This restriction is well-studied in job-shop scheduling Gonzalez (1982); Hromkovič et al. (2007); Lenstra and Kan (1979) 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 $m = 2$ the job shop scheduling problem with two counter routes and no preemption can be solved in polynomial time.

E.g., Jackson (1956) give an $O\Big((n^l + n^r)log(n^l + n^r)\Big)$ algorithm for a problem equivalent to $F^{\pm}n//C_{max}$. We conclude that the problem (STTS) can be solved in polynomial time for $m = 2$ and that this result would even hold if every train block combination had a different traversal time.

Still, already for $m = 3$ machines, job shop scheduling with two counter routes is NP-hard without the constraint $p_{ij} = p_i$, that is, 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 Garey et al. (1976). The result even holds for the case that the number of jobs from both sides are equal, i.e., if $n^l = n^r$, since the special case where all jobs from the right have 0 processing times is again equivalent to flow shop scheduling.

However, this complexity proof does not carry over to the (STTS) where the block traversal times are equal for all jobs, that is, $p_{ij} = p_i$. 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 Dushin (1988) an upper bound on the objective value for job shop scheduling with two routes (not necessarily counter routes) on m machines is proven. Also Babushkin et al. (1977) develop an upper bound for the counter routes problem on m machines. In Dushin (1988), Babushkin et al. (1974) is mentioned as an earlier publication on this subject however, we were not able to locate this paper. See also the scheduling overviews given by Chen et al. (1998) and Sevast'janov (1994) 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. Table 1 summarizes the important notations for reference.

Let a linear graph (i.e., a graph which can be displayed as a line with intermediate nodes) $G = (V, B)$ with stations $V = \{s_1, \dots, s_{m+1}\}$ and undirected edges (called *blocks* in the remainder of this paper) $B = \{b_1 = (s_1, s_2), \dots, b_m = (s_m, s_{m+1})\}$ be given. In order to facilitate the problem description, we assume that the blocks are indexed from left to right. That is, we define station s_1 to be the leftmost station and station s_{m+1} to be the rightmost respectively.

The traversal times of the blocks are given as t_i for block $b_i = (s_i, s_{i+1})$ for all $i = 1, \dots, m$. For simplicity of notation we denote by $t_{j,k}$ the joint traversal time for blocks $b_j, \dots, b_k$, i.e., $t_{j,k} = \sum_{i=j}^{i=k} t_i$. 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_{m+1} (or from left to right respectively) is specified by n^l and the number of trains traversing G in opposite direction is given as n^r. Although in most practical applications, trains going to the same direction could be on the same block as long as they maintain a safety distance between them, in this paper we make the simplifying assumption that at any point in time there can at most be one train on each block. Furthermore we assume that the capacity of the stations is not restricted, that is, an unlimited number of trains may be stored at any station. We make the described simplifying assumptions in order to study the simplest version of the (STTS) which is still interesting and challenging. Possible extensions and ways to make the model more realistic are discussed in Section 5.

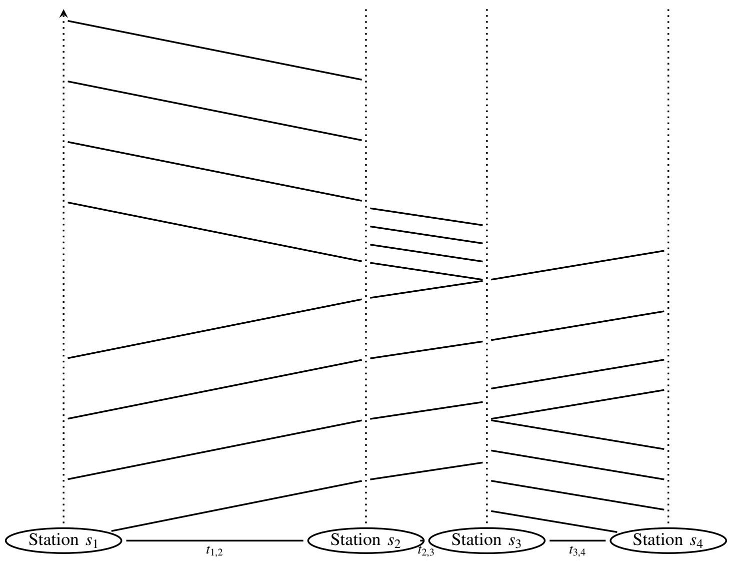

Figure 2: Example for a space time diagram with $n^l = n^r = 4$ and $m = 3$

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. Let $n^l > 0$ denote the number of trains from left to right and $n^r > 0$ the number of trains from right to left. The objective is to minimize the makespan for the traversal of G for all trains.

In the following we denote an instance I of the problem (STTS) as a tuple $I = ((t_1, \dots, t_m), n^l, n^r)$ for which the station and block capacities are given as above. Note that the graph G can be constructed from the times $(t_1, \dots, t_m)$.

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 trajectories of trains. Figure 2 gives an example of 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, the *remaining part* of the network. The time-wise distance between two trains (usually on a specific block) is called *headway*.

Before we study bounds, complexity of special cases and efficient solution techniques in the next section, we make a formal mathematical definition of the optimization problem by formulating it as an ILP. In the following we present an ILP-formulation which is similar to common ILP formulations in machine scheduling, see e.g., Manne (1960).

This ILP-formulation relies on the idea of introducing variables which express the precedence of different jobs on each machine, or, in our train representation, different trains on each block. Let $\mathcal{J} = \left\{1, \dots, n^l, n^l + 1, \dots, n^l + n^r\right\}$ be the set of trains. The variables determining the sequence in which they are processed are specified as

$$y_{k,j}^b = \begin{cases} 1 & \text{, if train } j \text{ follows directly after train } k \text{ on block } b \\ 0 & \text{, otherwise.} \end{cases} \text{, and}$$

$$y_{j,n^l+n^r+1} = \begin{cases} 1 & \text{, if } j \text{ is the last train} \\ 0 & \text{, otherwise.} \end{cases}$$

Furthermore, we have to determine the points in time at which a train i starts to traverse a block b (referred to as *starting times* in the remainder of this paper). The variable S_i^b states the starting time of train i on block b. With this, the IP-formulation can be stated formally.

$$\textbf{(STT-MS)} \qquad \min \max_{\substack{i=1,\dots,n \\ j\in\mathcal{J}}} \left(S_j^{b_i} + t_i\right) \tag{1}$$

$$\text{s.t.} \qquad S_j^{b_{i-1}} + t_{i-1} \leq S_j^{b_i} \quad \forall\, j \in \left\{1,\dots,P^l\right\}, i = 2,\dots,n-1 \tag{2}$$

$$S_j^{b_i} + t_i \leq S_j^{b_{i-1}} \quad \forall\, j \in \left\{P^l+1,\dots,P^l+P^r\right\}, i = 2,\dots,n-1 \tag{3}$$

$$S_k^{b_i} + \left(t_i - M\left(1 - y_{k,j}^{b_i}\right)\right) \leq S_j^{b_i} \qquad \forall\, k, j \in \mathcal{J},\ i = 1,\dots,n \tag{4}$$

$$\sum_{j=1}^{P^l+P^r} y_{k,j}^{b_i} = 1 \qquad \forall\, k \in \mathcal{J}\ i = 1,\dots,n \tag{5}$$

$$\sum_{j=1}^{P^l+P^r} y_{j,P^l+P^r+1}^{b_i} = 1 \qquad \forall\, i = 1,\dots,n \tag{6}$$

$$S_j^{b_i} \in \mathbb{N} \qquad \forall\, j \in \mathcal{J},\ i = 1,\dots,n \tag{7}$$

$$y_{k,j}^{b_i} \in \{0,1\} \qquad \forall\, k, j \in \mathcal{J},\ i = 1,\dots,n \tag{8}$$

Constraints (2,3) ensure that a train does not start traversing a block before it has completed traversing the preceding block. Constraints (4) enforce that a block can only be traversed by one train at a time. The requirement that each train has exactly one successor is encoded in constraint (5). Constraints (6) ensure that only one train is marked as the last train on each block b_i. Finally the objective (1) minimizes the time the last train completes the traversal of all blocks.

To find a suitable value for M in constraints (4), we note that $S_k^{b_i} + t_i$ is bounded by any bound on the makespan, for example $T^{up}(I)$ as given in Lemma 4.2.

In order to overcome this drawback of big-M constraints we develop specific bounds and complexity results and a dynamic programming algorithm in the following.

Note that it is also possible to give an ILP-formulation where variables specify precedence constraints, in contrast to the *immediate* precedence constraints used in this formulation. However, we omit this formulation here since it is very similar.

4 Theoretical Results

After the definition of the problem and the model in the previous section and its formalization as an ILP, we now develop complexity results for (STTS). First, we prove bounds which help to subsequently prove complexity results for special cases of (STTS). Finally, we develop a dynamic programming algorithm.

4.1 Bounds

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

Note that in the formulation of (STTS) we assumed that there are trains from both sides. For completeness, let us briefly discuss what happens when trains traverse only from one side ($n^l = 0$ or $n^r = 0$). Then, the optimal objective value is given by

$$t_{1,m} + (n^l + n^r - 1) \max_{i=1,\dots,m} t_i. \tag{9}$$

This result follows from Carlier (1982). Thus, for the remainder of this paper, we only consider the bidirectional case, hence require $n^l > 0$ and $n^r > 0$.

In case of bidirectional traffic the problem solutions are not as easy as with unidirectional traffic, hence we state the following lower bound.

Lemma 4.1. *$T^{lo}(I)$ is a lower bound on the objective value of an instance I of (STTS).*

$$T^{lo}(I) = \max_{i=1,\dots,m} \left\{ \left(n^l + n^r\right) t_i + 2 \min \left\{ t_{1,i-1}, t_{i+1,m} \right\} \right\} \tag{10}$$

In words, equation (10) means that the time needed for all trains to traverse the block can never be lower than the time needed to traverse any block b_i plus the time the first train needs to reach the block b_i plus the time the last train needs to reach its destination after departing from block b_i. Since this has to hold for all blocks, and we wish to have the highest possible lower bound we take the maximum over all such blocks b_i, $i = 1, \dots, m$.

Let the block for which the maximum is attained in (10) be called the *bottleneck block*. Throughout this section we see that in many cases, a good scheduling strategy on the bottleneck block guarantees that the lower bound can be obtained as the makespan.

Note that alternatively to the intuitive explanation given above, Lemma 4.1 can also be deduced from a result in Carlier (1982), by transferring the (STTS) to an equivalent unidirectional scheduling problem on three blocks with two different types of trains, which differ in the traversal times of the first and last block.

This lower bound is of help to show that particular cases can be solved in polynomial time. For the proof of the complexity of one instance of problem (STTS) in Theorem 4.8 we also need the upper bound on the objective value.

$$T^{up}(I) = (n^l + n^r) \max_{i=1,\dots,m} t_i + 2 \left(t_{1,m} - \max_{i=1,\dots,m} t_i \right) \tag{11}$$

We now show the correctness of $T^{up}(I)$.

Lemma 4.2. *For an instance I of (STTS) $T^{up}(I)$ is an upper bound.*

Proof. We proof this lemma by constructing a feasible schedule with makespan $T^{up}(I)$. Our schedule is as follows: first all trains from the left traverse the network with a headway of $\max_{i=1,\dots,m} t_i$, afterwards, all trains from the right traverse the network, also with a headway of $\max_{i=1,\dots,m} t_i$. To compute the makespan

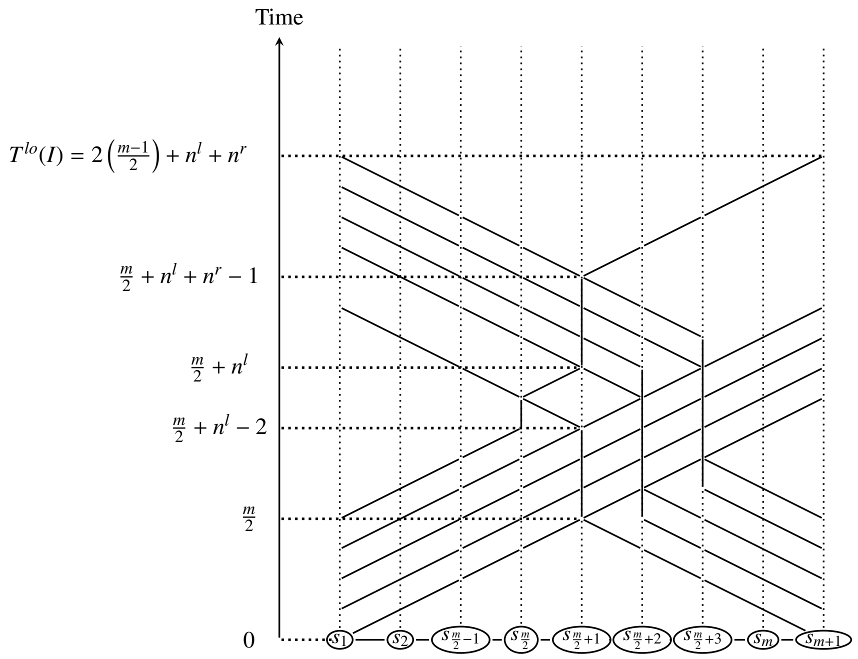

Figure 3: Strategy for unit processing time and even number of blocks with $m = 8$, $n^l = n^r = 5$

of this schedule, we observe that the first train from the right reaches its destination after $t_{1,m}$ units of time. Since all other trains from the left follow with headways of $\max_{i=1,\dots,m} t_i$, the last train from the left arrives at its destination at time $t_{1,m} + (n^l - 1) \cdot \max_{i=1,\dots,m} t_i$. Using an analogous argument, the trains from the right need time $t_{1,m} + (n^r - 1) \cdot \max_{i=1,\dots,m} t_i$ to traverse the network. Adding up these two values gives us a makespan of $T^{up}(I)$. $\square$

4.2 Special Cases

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

4.2.1 Unit Processing Times and Capacity Restrictions at Stations

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

Theorem 4.3. *For instances of (STTS) with unit traversal 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. Note that in this case, since $t_i = 1$ for all i, we have $T^{lo}(I) = \left(n^l + n^r\right) + m - 1$.

First, assume that no capacity restrictions are enforced. Two different cases are considered: m is even and m is odd.

Consider the case where m is even. The following strategy provides an optimal schedule. See Figure 3 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 n^lth - in this case give the trains from the right precedence. After all trains from the left, except the n^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 n^lth train continues to station s_{m+1}.

The makespan of this strategy will be the arrival time of the last train from the left at its destination. We will thus follow its course through the network in the remainder of this proof. The last train from the left traverses the block left of the central station $s_{\frac{m}{2}+1}$ 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 block $s_{\frac{m}{2}+1}$ and finally heads for its destination s_{m+1}.

To see when the last train from the left leaves from station $s_{\frac{m}{2}+1}$, we need to determine when the last train from the right has passed block $s_{\frac{m}{2}+1}$: The first train from the left and from the right reach the central station $s_{\frac{m}{2}+1}$ at the same time $\frac{m}{2}$. After that, all trains from the left head for station s_{m+1}. In particular, they traverse $b_{\frac{m}{2}+1}$, the block immediately right of the central station. At time $\frac{m}{2} + n^l - 1$, all trains from the left except for the last one have crossed $b_{\frac{m}{2}+1}$.

Let us now consider the positions of the trains from the right: It is easy to see that at time $\frac{m}{2} + n^l - 2$, the trains from the right are at the following positions: At station $s_{\frac{m}{2}+1}$ there is one train waiting. If the number of trains from the right is at least $m-1$, at stations $s_{\frac{m}{2}+2}, s_{\frac{m}{2}+3}, \ldots, s_m$ there are two trains waiting the remaining trains wait at station s_{m+1}. If the number of trains from the right is lower than $m-1$, then there are no trains at the rightmost stations.

At time $\frac{m}{2} + n^l - 2$ 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 $s_{\frac{m}{2}+1}$ and the block $b_{\frac{m}{2}+1}$ 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 $b_{\frac{m}{2}}$ and thus waits at station $s_{\frac{m}{2}+1}$. 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_{m+1}. Once the last train from the right has arrived at the central station $s_{\frac{m}{2}+1}$ (at time $\frac{m}{2} + n^l + n^r - 1$) both this train and the last train from the left finish their ride to their final station. This ride takes each train $\frac{m}{2}$ time units. Hence, the makespan of this schedule is $\frac{m}{2} + n^l + n^r - 1 + \frac{m}{2} = T^{lo}(I)$.

To make sure that we can indeed schedule the trains as descried above without capacity conflicts, we make the following considerations: On block $b_{\frac{m}{2}+1}$ we find the following sequence of trains. Until time $\frac{m}{2} - 1$ 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.

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 m is odd. To achieve a makespan which equals the lower bound from Lemma 4.1, we use the following strategy: Let all trains from both sides 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 $b_{\frac{m+1}{2}}$ is reached by a train from each side at time $\frac{m-1}{2}$. 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 $s_{\frac{m+1}{2}+1}$ to the right there are two trains waiting. Finally, the last train has traversed the central block at time $\frac{m-1}{2} + n^l + n^r$ and it takes another $\frac{m-1}{2}$ time units until it reaches the final station s_1. Hence the makespan equals $n^l + n^r + 2\left(\frac{m-1}{2}\right)$ 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 4.4. *For instances of (STTS) with unit processing times and $n^l = n^r$, 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 $n := n^l = n^r$. Again we separate the analysis into the cases where m is even and m is odd. First assume m is even.

To achieve a makespan which equals the lower bound from Lemma 4.1, we use the following strategy: Every two time units, we let a train from the left depart at station s_1, and a train from the right depart at station s_{m+1}. Then, since m 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 $m + 2(n-1) = 2n + 2\left(\frac{m}{2} - 1\right) = T^{lo}(I)$.

In case m 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 $1 + m + 2(n-1) = 2n + 2\left(\frac{m-1}{2}\right) = T^{lo}(I)$, the last train from the left arrives at time $m + 2(n-1) = 2n + 2\left(\frac{m-1}{2}\right) - 1 < T^{lo}(I)$.

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 station capacity one which also holds if block lengths are not equal. In this case, since trains cannot pass each other, an optimal strategy simply consists of letting the trains from one side pass first, and then the trains from the other side.x This leads to the following lemma.

Lemma 4.5. *If the station capacity is restricted to one, the makespan of an optimal schedule is given by $2t_{1,m} + (n^l + n^r - 2)\max_{i=1}^{m} t_i$.*

Note that for this case, the makespan is higher than $T^{lo}(I)$.

4.2.2 Restricting the Number of Blocks

In the following we consider arbitrary block lengths, but restrict the number of blocks on the track. From the machine scheduling analysis we know that the case $m = 2$ is easily solvable. Hence, we direct to the case where $m = 3$. Take Figure 1 as an illustration of the considered case.

To prove our results in Theorem 4.7 and Theorem 4.8, we need the notion of a *sorted* schedule. We will call a schedule with makespan T on a three-block train line *sorted*, if the trains from the left start traversing block b_1 at times 0, t_1, $2t_1$, …, $(n^l - 1)t_1$ and start traversing block b_3 at times $T - n^l t_3$, $T - (n^l - 1)t_3$, $T-(n^l-2)t_3, \dots, T-t_3$, and the trains from the right start traversing block b_3 at times $0, t_3, 2t_3, \dots, (n^r-1)t_3$ and start traversing block b_1 at times $T - n^r t_1$, $T - (n^r - 1)t_1$, $T - (n^r - 2)t_1, \dots, T - t_1$.

In the proof of Theorem 4.8 we make use of the following lemma.

Lemma 4.6. *Consider an instance I of (STTS). If there exists a feasible schedule with makespan T for I, then there is also a feasible sorted schedule with makespan T.*

Proof. Assume that we have a feasible schedule with makespan T. We now show that if the schedule is not sorted, we can iteratively interchange the order of trains on block b_1 and b_3 while preserving feasibility and without increasing the makespan. The following argument is symmetrically applicable to both blocks b_1 and b_3 and thus we only show it on block b_1. Assume that the schedule S is not sorted, in particular that there is a train from the right scheduled before a train from the left on block b_1 If we now interchange the order of these two trains on block b_1, we obtain a new schedule S'. This resulting schedule is still feasible since the train from the left only arrives earlier at station s_2 and the train from the right only departs later from station s_2. Also T is neither increased nor decreased, i.e., there is no change in the objective value. By iteratively applying these switches on block b_1, and similar ones on block b_3, we obtain a schedule which is feasible and has the same makespan.

If we now move the departures of the trains from the left on b_1 forward in time and the departures of the trains from the right on b_1 backwards in time, and apply an analogous schedule update on b_3, we obtain a feasible schedule with the same makespan which is sorted. □

We first consider the special case $n^l = n^r$. In Theorem 4.7 we show that in this case, we can always find a schedule with makespan $T^{lo}(I)$.

Theorem 4.7. *For instances I of (STTS) with three blocks and $n = n^l = n^r$, the makespan of an optimal schedule is given by $T^{lo}(I)$.*

Proof. For $i = 3$, $T^{lo}(I)$ equals the largest of the three lower bounds $\overbrace{2nt_1}^{(1)}, \overbrace{2nt_2 + 2\min\{t_1, t_3\}}^{(2)}, \overbrace{2nt_3}^{(3)}$.

Suppose bound (2) is largest. Then, assume wlog. that $t_1 \geq t_3$. We now construct a sorted schedule with makespan $T^{lo}(I)$, then we show that it is feasible. Since the schedule is sorted, we only need to describe departures on block b_2 to describe the schedule. Block b_2 is initially unused until time t_3. After that n trains go alternately and consecutively starting with a train from right to left until n trains have passed this block at time $t_3 + nt_2$. The second half (timewise) of the schedule is described backwards in time from the time $T^{lo}(I)$. Block b_2 is unused from $T^{lo}(I) - t_3$ until $T^{lo}(I)$. Before that time we have alternately moving trains, the last with destination s_4, the second last with destination s_1 and so on. We now argue that this is a feasible 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 (timewise) half of the schedule; the second half is analogous. Following the argumentation of above, the time duration of the first half of the schedule is $t_3 + nt_2$. Clearly the n movements scheduled for block b_2 fit in this duration, after the initial inactive period of length t_3. Because bound (2) is largest we know that $nt_1, nt_3 \leq nt_2 + t_3$. Thus the n rightward (leftward) movements on block b_1 (b_3) 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_1 at time it_1 and enter block b_2 at $t_3 + (2i-1)t_2$. Since $t_1 \leq t_2 + \frac{t_3}{n}$, we have that

$$it_1 \leq it_2 + i\frac{t_3}{n} \leq (2i-1)t_2 + t_3.$$

Thus, the train is scheduled to enter block b_2 only after leaving block b_1.

Likewise, the ith leftward train is scheduled to leave block b_3 at it_3 and to enter block b_2 at $(2i-2)t_2 + t_3$. This is possible since it holds that

$$it_3 = t_3 + (i-1)t_3 \leq t_3 + (i-1)t_1 \leq t_3 + (i-1)(t_2 + \frac{t_3}{n}) \leq t_3 + (2i-2)t_2.$$

The last inequality in this is derived as follows. Note first that if $n = 1$ then $i = 1$ and the proof is immediate. Hence consider $n > 1$. We know that $nt_3 \leq nt_2 + t_3$ and thus $t_2 \geq t_3(n-1)/n$. This gives us $t_2 \geq \frac{t_3}{n}$ for $n > 1$.

Next suppose bound (1) is the largest. For ease of argument, we increase t_2 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_2. In the schedule, certain time intervals are reserved for movements on block b_2. Of each such interval we only use a subinterval of length equal to the original value of t_2. 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. □

If $n^l \neq n^r$, it is not always possible to construct a schedule whose makespan is the lower bound from Lemma 9, even if we have only three blocks (see Example 4.10).

In the following we describe how an optimal schedule can be found when we have three blocks but $n^l \neq n^r$. First note that the upper bound $T^{up}(I)$, given in (11), reduces to the following in case of $m = 3$.

$$T^{up}(I) = \left(n^l + n^r\right) \max_{i=1,\ldots,3} t_i + 2\left(t_{1,3} - \max_{i=1,\ldots,3} t_i\right) \tag{12}$$

Theorem 4.8. *For any instance I of (STTS) with three blocks, i.e., $m = 3$, we can find a schedule with minimal makespan in time*

$$O\left(\left(n^l + n^r\right)^2 \log\left(n^l + n^r\right) \log\left(t_{1,3} - \max_{i=1,\ldots,3} t_i\right)\right) \tag{13}$$

Proof. We first show that with a given makespan T, for which a feasible schedule exists, we are able to construct a solution to I with makespan T in polynomial time. In particular, this is true if T is the optimal makespan $T = T_{opt}$.

According to Lemma 4.6, we can restrict our search to sorted schedules. We now translate the problem of scheduling the trains on the central block b_2 one-to-one to a single machine scheduling problem with jobs having release and due dates (Simons, 1978). Each train corresponds to one job i. Its release time $r_{k,i}$ is the arrival time at station s_2 for trains from the left ($k = l$) and at station s_3 for trains from the right ($k = r$). The deadline $d_{k,i}$ is the departure time from station s_3 for trains from the left ($k = l$) (according to the above created schedule on b_3) and the departure time from s_2 for trains from the right ($k = r$). Release times and due dates are matched in order, i.e., train 1 from left has release time $r_{l,1} = 0$ and due date $d_{l,1} = T - n^l t_3$. Train 2 from the left has release time $r_{l,2} = t_1$ and due date $d_{l,2} = T - (n^l - 1)t_3$ and so on.

By applying the algorithm given in Simons (1978), we find a schedule which is feasible with respect to the release times and due dates specified, or conclude that no such schedule exists in time $O\left(\left(n^l + n^r\right)^2 \log\left(n^l + n^r\right)\right)$ (Simons, 1978).

Hence, if the algorithm of Simons (1978) does not find a feasible schedule on the central block, then there is no solution to the (STTS) with makespan T or lower. If feasible schedule is found, we can translate it easily into a solution to the (STTS) with makespan T.

To find the optimal schedule for I, we conduct a binary search on the makespan T in the interval given by the lower bound (10) and upper bound (12), $T \in [T^{lo}(I), T^{up}(I)]$. The complexity of the binary search steps is determined by the difference $T^{up}(I) - T^{lo}(I)$.

$$\begin{aligned} T^{up}(I) - T^{lo}(I) &= \left(n^l + n^r\right) \max_{i=1,\dots,3} t_i + 2\left(t_{1,3} - \max_{i=1,\dots,3} t_i\right) \\ &\quad - \max_{i=1,\dots,3} \left\{\left(n^l + n^r\right) t_i + 2\min\left\{t_{1,i-1}, t_{i+1,3}\right\}\right\} \\ &\leq 2\left(t_{1,3} - \max_{i=1,\dots,3} t_i\right) \end{aligned}$$

Hence, the number of steps in the binary search is $O\left(\log\left(t_{1,3} - \max_{i=1,\dots,3} t_i\right)\right)$ and the resulting total complexity is

$$O\left(\left(n^l + n^r\right)^2 \log\left(n^l + n^r\right) \log\left(t_{1,3} - \max_{i=1,\dots,3} t_i\right)\right)$$

□

4.2.3 Can the Lower Bound $T^{lo}(I)$ Always be Achieved?

The following example shows a case for $m = 5$ and $n^l = n^r$ in which the lower bound $T^{lo}(I)$ cannot be achieved.

Example 4.9. Assume $n^l = n^r = 2$ and $t_1 = t_5 = 10$, $t_2 = t_4 = 3$ and $t_3 = 4$. Then $T^{lo}(I) = (n^l + n^r)t_3 + 2(t_1 + t_2) = 42$ and the bottleneck is block b_3. 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 $44 = T^{lo} + 2$.

The next example shows that in case of $m = 3$ there also exist cases in which the lower bound $T^{lo}(I)$ cannot be reached.

Example 4.10. Let $n^l = 2$ and $n^r = 5$, $m = 3$, $t_{1,2} = 7$, $t_{2,3} = 6$, and $t_{3,4} = 8$. Then $T^{lo}(I) = (n^l + n^r)t_{2,3} + 2\min\{t_{1,2}, t_{3,4}\} = 7 \cdot 6 + 2 \cdot 7 = 56$ and b_2 is the bottleneck block. Figure 5 shows an optimal scheduling strategy. In this schedule the last train from the right on the central block has to wait for its traversal of the first block because the first block is still occupied by the penultimate train from the right. This is where a one additional time unit which is not covered by the lower bound is added to the schedule.

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.

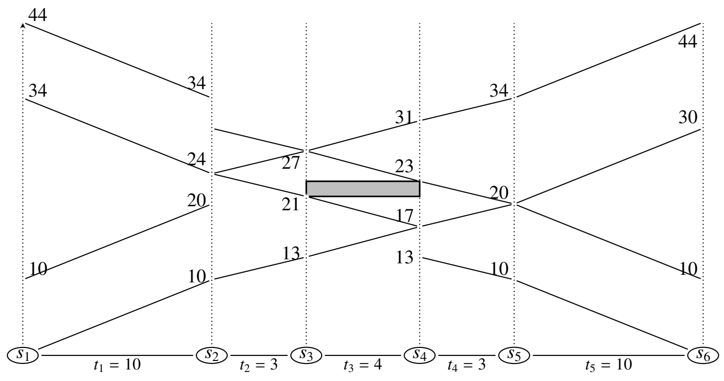

Figure 4: Sketch for scheduling strategy for Example 4.9

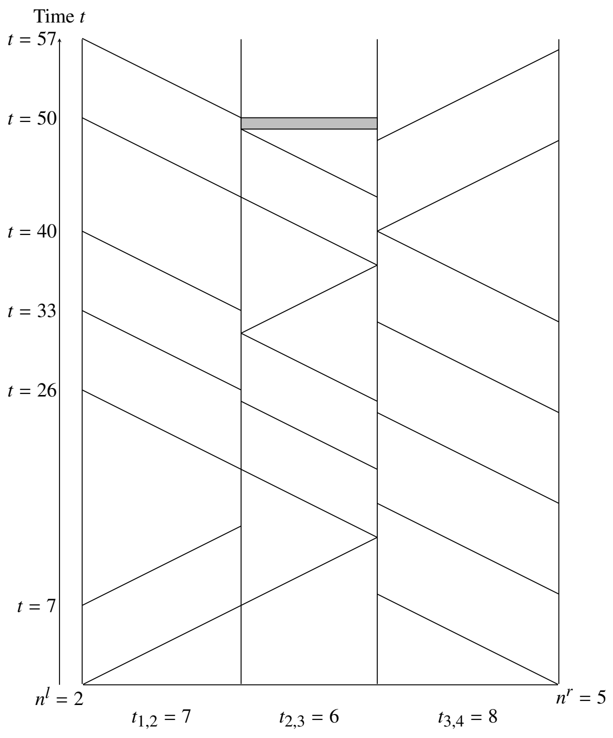

Figure 5: Sketch for scheduling strategy for Example 4.10

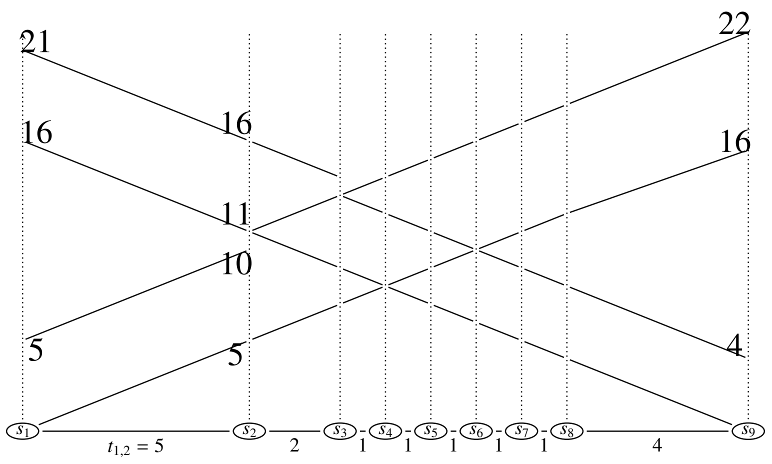

Figure 6: Lower bound T^{lo} not reached even though longest block is bottleneck

Observation 4.11. If there exists a $k = 1, 2, \ldots, 2n$ such that at time $\min\{t_{1,i-1}, t_{i+1,m}\} + (k-1)t_i$ 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 4.12. Let $n^l = n^r = 2$ and $(t_1, t_2, \ldots, t_8) = (5, 2, 1, 1, 1, 1, 1, 4)$. Then the bottleneck block is obtained as b_1 with $T^{lo}(I) = 20$. 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 6 where the scheduling strategy is depicted.

4.3 A Pseudo-Polynomial Algorithm for Fixed Block Number

In this section we develop a dynamic programming formulation for the problem (STTS). Based on this formulation, we can show that for fixed number of blocks, the problem (STTS) can be solved in pseudo-polynomial time. As will become clear in Lemma 4.15, the value of this lemma is mainly theoretic, since the running time of the dynamic algorithm is of high polynomial order and thus increases very fast with increasing problem size.

Lemma 4.13. *If block lengths are integer, (STTS) can be solved as a shortest-path problem in an acyclic graph.*

Proof. Denote by $L := t_{1,m}$ the total length of the track considered. We divide the blocks 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 $j = m + 1$) 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

$$X := \begin{array}{c|c|c|c|c|c} x_0^l & x_1^l & x_2^l & \dots & x_{L-1}^l & x_L^l \\ \hline x_0^r & x_1^r & x_2^r & \dots & x_{L-1}^r & x_L^r \end{array}. \tag{14}$$

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 $[n] := \{0, 1, 2, \dots, n\}$.

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

- At the beginning, all trains passing from left are on the left of the track, i.e., $x_0^l = n^l$ for all feasible states X, analogously $x_L^r = n^r$ for all feasible states X.
- Since trains pass from left to right, we have $x_i^l \geq x_{i+1}^l$ for all $i = 0, \dots, L$, analogously $x_i^r \leq x_{i+1}^r$.
- Only one train can be on a block at a time.

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

Only nodes representing feasible states are 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 $i = 0, \dots, L$, $k \in \{l, r\}$, 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 $j = 1, \dots, m$, 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

$$\begin{array}{c|c|c|c|c|c} n^l & 0 & 0 & \dots & 0 & 0 \\ \hline 0 & 0 & 0 & \dots & 0 & n^r \end{array}$$

to node

$$\begin{array}{c|c|c|c|c|c} n^l & n^l & n^l & \dots & n^l & n^l \\ \hline n^r & n^r & n^r & \dots & n^r & n^r \end{array}$$

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 4.14. *If block lengths are integer, the graph described in Lemma 4.13 has* $O\left(\binom{n^l+m}{m}\binom{n^r+m}{m}\max\{t_i\}\right)$ *nodes and* $O\left(3^m\binom{n^l+m}{m}\binom{n^r+m}{m}\max\{t_i\}\right)$ *arcs.*

Proof. As stated in the proof of Lemma 4.13, the nodes of the graph correspond to feasible states X (as described in (14)). We now make an attempt on bounding the number of feasible states from above.

We denote by y^l_j the number of trains which have passed the full block b_j from left to right. y^r_j analogously denotes the number of trains which have passed the full block b_j from right to left. I.e., $y^l_j = x^l_{i_{\text{last}^j}}$ and $y^r_j = x^l_{i_{\text{first}^j}}$, 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^l_j/y^r_j) there are $2t_j - 1$ feasible combinations of subindices:

$$\begin{array}{c|c|c|c|c|c} y^l_j = x^l_{i_{\text{first}^j}} & x^l_{i_{\text{first}^j}+1} & x^l_{i_{\text{first}^j}+2} & \dots & x^l_{i_{\text{last}^j}-1} & y^l_{j+1} = x^l_{i_{\text{last}^j}} \\ \hline y^r_j = x^r_{i_{\text{first}^j}} & x^r_{i_{\text{first}^j}+1} & x^r_{i_{\text{first}^j}+2} & \dots & x^r_{i_{\text{last}^j}-1} & y^r_{j+1} = x^r_{i_{\text{last}^j}} \end{array},$$

namely

$$\begin{array}{c|c|c|c|c|c} y^l_j & y^l_j-1 & y^l_j-1 & \dots & y^l_j-1 & y^l_j-1 \\ \hline y^r_j & y^r_j & y^r_j & \dots & y^r_j & y^r_j \end{array}, \quad \begin{array}{c|c|c|c|c|c} y^l_j & y^l_j & y^l_j-1 & \dots & y^l_j-1 & y^l_j-1 \\ \hline y^r_j & y^r_j & y^r_j & \dots & y^r_j & y^r_j \end{array}, \dots, \begin{array}{c|c|c|c|c|c} y^l_j & y^l_j & y^l_j & \dots & y^l_j & y^l_j \\ \hline y^r_j & y^r_j & y^r_j & \dots & y^r_j & y^r_j, \end{array},$$

and

$$\begin{array}{c|c|c|c|c|c} y^l_j & y^l_j & y^l_j & \dots & y^l_j & y^l_j \\ \hline y^r_j-1 & y^r_j-1 & y^r_j-1 & \dots & y^r_j-1 & y^r_j \end{array}, \dots, \begin{array}{c|c|c|c|c|c} y^l_j & y^l_j & y^l_j & \dots & y^l_j & y^l_j \\ \hline y^r_j-1 & y^r_j & y^r_j & \dots & y^r_j & y^r_j \end{array}, \begin{array}{c|c|c|c|c|c} y^l_j & y^l_j & y^l_j & \dots & y^l_j & y^l_j \\ \hline y^r_j & y^r_j & y^r_j & \dots & y^r_j & y^r_j, \end{array},$$

(one case is listed twice here), if $y^l_j, y^r_j > 0$. If one or both values are 0, there are less.

Furthermore, there are

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

feasible combinations of the y-values. Each nested sum

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

then gives a binomial coefficient, see Butler and Karasik (2010) for an explanation. This leads to $O\left(\binom{n^l+m}{m}\binom{n^r+m}{m}\max\{t_i\}\right)$ 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 m blocks, each node X has hence at most 3^m successors (and most have much less). Thus, there are at most $O\left(3^m\binom{n^l+m}{m}\binom{n^r+m}{m}\max\{t_i\}\right)$ arcs in the graph. □

Lemma 4.15. *The (STTS) can be solved in* $O\left(3^m\binom{n^l+m}{m}\binom{n^r+m}{m}\max\{t_i\}\right)$.

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., Cormen et al. (2001). □

Of course, for large numbers of blocks m this is impractical and we would probably be better off using an integer programming approach as described, e.g., in Cacchiani and Toth (2012). However, from a theoretical point of view, the following conclusion is interesting:

Corollary 4.16. For a fixed number of blocks, the (STTS) can be solved in pseudo-polynomial time, i.e., it is polynomial in n^l, n^r and $\max_j t_j$.

Note that the time complexity for (STTS) with three blocks stated in Theorem 4.8 is considerably smaller than the one stated in Lemma 4.15. For $m = 3$ and $n^l = n^r = n$ we obtain a complexity from Theorem 4.8 of

$$O\left(n^2\left(\log^2 n + \log n \log\left(\max\{t_1, t_3\}\right)\right)\right)$$

which is much smaller than what is obtained from Lemma 4.15:

$$O\left(n^6 \max_{i=1,\dots,3}\{t_i\}\right)$$

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 n^l the number of departures from left and by n^r the number of departures from right. Introducing only nodes X which fulfill

$$x^l_2 - x^r_1 \leq \text{initial number of trains on left}$$

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^r_0 are too different, w.r.t. to the above stated balancedness, or where the difference between x^l_i and x^r_i 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 pseudo-polynomial 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.

5 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 and an upper bound on the minimum makespan of the (STTS) and identify several special cases where the 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. Even more, we found that the job shop scheduling problem with two counter routes, no preemption and three machines can be solved in polynomial time in the number of trains and in logarithmic time in the traversal times.

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.

Acknowledgement

The authors were partially funded by the DFG under grant number SCHO1140/3-2 and by the European Union Seventh Framework Programme (FP7-PEOPLE-2009-IRSES) under grant number 246647 with the New Zealand Government (project OptALI). We also thank the Simulationswissenschaftliches Zentrum Clausthal-Göttingen (SWZ) for financial support.

Notation	Description
m	Number of blocks
G	Graph G consisting of nodes (stations) V and blocks B
V	Nodes (stations) of graph G, with $V = \{s_1, \ldots, s_{m+1}\}$
B	Blocks (tracks) of graph B, with $B = \{b_1, \ldots, b_m\}$
s_i	A particular node in graph G
b_i	A particular block in graph G
n^l	Number of trains from the left side
n^r	Number of trains from the right side
t_i	Traversal time for a train on block b_i
$t_{j,k}$	Traversal time for all blocks $b_j, \ldots, b_k$, i.e., $t_{j,k} = \sum_{i=j}^{k} t_i$

Table 1: Important notations of the discussed (STTS) problem.

References

A. I. Babushkin, A. A. Bashta, and I. S. Belov. Scheduling for the problem with oppositely directed routes. *Kibernetika*, 7:130–135, 1974.

A. I. Babushkin, A. A. Bashta, and I. S. Belov. Scheduling for problems of counterroutes. *Cybernetics and Systems Analysis*, 13(4):611–617, 1977.

E. Balas. Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. *Operations research*, 17(6):941–957, 1969.

C. Bersani, S. Qiu, R. Sacile, M. Sallak, and W. SchÃ¶n. Rapid, robust, distributed evaluation and control of train scheduling on a single line track. *Control Engineering Practice*, 35:12 – 21, 2015.

R. Borndörfer, B. Erol, T. Graffagnino, T. Schlechte, and E. Swarat. Optimizing the simplon railway corridor. *Annals of Operations Research*, 218(1):93–106, 2014.

P. Brucker, S. Heitmann, and S. Knust. Scheduling railway traffic at a construction site. In Hans-Otto Günther and Kap Hwan Kim, editors, *Container Terminals and Automated Transport Systems*. Springer Berlin Heidelberg, 2005.

S. Butler and P. Karasik. A note on nested sums. *Journal of Integer Sequences*, 13(2):3, 2010.

V. Cacchiani and P. Toth. Nominal and robust train timetabling problems. *European Journal of Operational Research*, 219(3):727–737, 2012.

J. Carlier. The one-machine sequencing problem. *European Journal of Operational Research*, 11(1):42–47, 1982.

E. Castillo, I. Gallego, J. M. Ureña, and J. M. Coronado. Timetabling optimization of a single railway track line with sensitivity analysis. *TOP*, 17(2):256–287, 2009.

B. Chen, C. Potts, and G. Woeginger. A Review of Machine Scheduling: Complexity, Algorithms and Approximability. *Handbook of Combinatorial Optimization*, 1998.

J.-F. Cordeau, P. Toth, and D. Vigo. A survey of optimization models for train routing and scheduling. *Transportation Science*, 32(4):380–404, 1998.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*, volume 2. MIT press Cambridge, 2001.

Y. Disser, M. Klimm, and E. Lübbecke. Scheduling bidirectional traffic on a path. In M. M. Halldórsson, K. Iwama, N. Kobayashi, and B. Speckmann, editors, *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 406–418. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

B. I. Dushin. An algorithm for the solution of the two-route johnson problem. *Cybernetics and Systems Analysis*, 24(3):336–343, 1988.

O. Frank. Two-way traffic on a single line of railway. *Operations Research*, 14(5):801–811, 1966.

M. R. Garey, D. S. Johnson, and R. Sethi. The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research*, 1976.

T. Gonzalez. Unit execution time shop problems. *Mathematics of Operations Research*, 7(1):57–66, 1982.

A. Higgins, E. Kozan, and L. Ferreira. Optimal scheduling of trains on a single line track. *Transportation Research Part B: Methodological*, 30(2):147–161, 1996.

E. Higgins, A.and Kozan and L. Ferreira. Heuristic techniques for single line train scheduling. *Journal of Heuristics*, 3(1):43–62, 1997.

J. Hromkovič, T. Mömke, K. Steinhöfel, and P. Widmayer. Job shop scheduling with unit length tasks: bounds and algorithms. *Algorithmic Operations Research*, 2(1), 2007.

J.R. Jackson. An extension of johnson's result on job lot scheduling. *Naval Research Logistics Quarterly*, 1956.

M. Janić. Single track line capacity model. *Transportation Planning and Technology*, 9(2):135–151, 1984.

D. Kraay, P. T. Harker, and B. Chen. Optimal pacing of trains in freight railroads: Model formulation and solution. *Operations Research*, 39(1):pp. 82–99, 1991.

J. K. Lenstra and A. R. Kan. Computational complexity of discrete optimization problems. *Annals of Discrete Mathematics*, 4:121–140, 1979.

E. Lübbecke. *On-and Offline Scheduling of Bidirectional Traffic*. Logos Verlag Berlin GmbH, 2015.

E. Lübbecke, M. E. Lübbecke, and R. H. Möhring. Ship traffic optimization for the kiel canal. In *Technical Report 4681*. (Optimization Online), 2014.

A. S. Manne. On the job-shop scheduling problem. *Operations Research*, 8(2):219–223, 1960.

J. A. Mesa, F. A. Ortega, and M. A. Pozo. Locating optimal timetables and vehicle schedules in a transit line. *Annals of Operations Research*, 222(1):439–455, 2014.

E. R. Petersen. Over-the-road transit time for a single track railway. *Transportation Science*, 8(1):65–74, 1974.

S. A. A. Rahman. *Freight Train Scheduling on a Single Line Network*. PhD thesis, The University of New South Wales, 2013.

D Šemrov, R Marsetič, M Žura, L Todorovski, and A Srdic. Reinforcement learning approach for train rescheduling on a single-track railway. *Transportation Research Part B: Methodological*, 86:250–267, 2016.

S.V. Sevast'janov. On some geometric methods in scheduling theory: a survey. *Discrete Applied Mathematics*, 55(1):59 – 82, 1994.

B. Simons. A fast algorithm for single processor scheduling. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 246–252, 1978.

Y. N. Sotskov and O. Gholami. Shifting bottleneck algorithm for train scheduling in a single-track railway. *IFAC Proceedings Volumes*, 45(6):87 – 92, 2012.

L. Yang, K. Li, and Z. Gao. Train timetable problem on a single-line railway with fuzzy passenger demand. *IEEE Transactions on Fuzzy Systems*, 17(3):617–629, June 2009.

L. Yang, Z. Gao, and K. Li. Passenger train scheduling on a single-track or partially double-track railway with stochastic information. *Engineering Optimization*, 42(11):1003–1022, 2010.

X. Zhou and M. Zhong. 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.