On the Approximability of Train Routing and the Min-Max Disjoint Paths Problem

Umang Bhaskar
TIFR Mumbai, India

Katharina Eickhoff
RWTH Aachen University, Germany

Lennart Kauther
RWTH Aachen University, Germany

Jannik Matuschke
KU Leuven, Belgium

Britta Peis
RWTH Aachen University, Germany

Laura Vargas Koch
RWTH Aachen University, Germany

Abstract

In train routing, the headway is the minimum distance that must be maintained between successive trains for safety and robustness. We introduce a model for train routing that requires a fixed headway to be maintained between trains, and study the problem of minimizing the makespan, i.e., the arrival time of the last train, in a single-source single-sink network. For this problem, we first show that there exists an optimal solution where trains move in convoys – that is, the optimal paths for any two trains are either the same or are arc-disjoint. Via this insight, we are able to reduce the approximability of our train routing problem to that of the min-max disjoint paths problem, which asks for a collection of disjoint paths where the maximum length of any path in the collection is as small as possible.

While min-max disjoint paths inherits a strong inapproximability result on directed acyclic graphs from the multi-level bottleneck assignment problem, we show that a natural greedy composition approach yields a logarithmic approximation in the number of disjoint paths for series-parallel graphs. We also present an alternative analysis of this approach that yields a guarantee depending on how often the decomposition tree of the series-parallel graph alternates between series and parallel compositions on any root-leaf path.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Approximation algorithms analysis

Keywords and phrases Train Routing, Scheduling, Approximation Algorithms, Flows over Time, Min-Max Disjoint Paths

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.34

Related Version *Full Version*: `https://doi.org/10.48550/arXiv.2507.03687` [1]

Funding *Umang Bhaskar*: supported by the Department of Atomic Energy, Government of India, under Project No. RTI4001.
Katharina Eickhoff: supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 2236/2.
Lennart Kauther: supported by the German Centre for Rail Traffic Research (DZSF) at the Federal Railway Authority under research project "Methodik der Kapazitätsbewertung des Gesamtsystems und Knotenberechnung".
Jannik Matuschke: supported by Fonds Wetenschappelijk Onderzoek (FWO) under Research Project G072520N "Optimization and Analytics for Stochastic and Robust Project Scheduling".

Acknowledgements We thank Daniel Schmand, Khai Van Tran and Andreas Wiese for fruitful discussions, as well as Nils Nießen and his group for providing us with foundational insights into train planning. We also thank the participants of the Dagstuhl Seminar 24281 [8], the Oberwolfach Workshop in Combinatorial Optimization 2024 [25], and the Chile Summer Workshop on Combinatorial Optimization 2025 for many interesting comments and questions. Finally, we thank the three anonymous reviewers for their careful reading of our paper and many useful suggestions.

1 Introduction

Given a rail network and a set of trains, the problem of train routing and scheduling refers to the problem of determining a route as well as a schedule for the trains to optimise some objective, such as the total travel time or the maximum travel time (also denoted as makespan). Train routing usually involves a number of real-world constraints, including train acceleration and deceleration, stopping times at stations, bidirectional arcs, presence of other trains, etc. Practical approaches to solving these complex problems often include mixed integer programming, heuristics, and simulations [3, 4, 15, 16, 22, 24], see [21, 22, 5] for surveys and an overview. However, theoretical guarantees on solutions are hard to come by, apart from some results that show NP-hardness already for some very restricted cases [15, 19].

One of the most important constraints in train routing and scheduling is that of *headway* [6, 18], which determines the minimum gap between successive trains on a link. The headway thus corresponds to a safety distance between two successive trains and is important for at least two different reasons. The first is safety: Maintaining sufficient headway between trains reduces the chances of a collision between trains. The second reason is robustness: If a train is delayed somewhat, then the headway can help ensure that this delay is not propagated to all the following trains, and the schedule can recover quickly. In fact, the importance of maintaining a headway is not restricted to train routing. In vehicular traffic, such as cars on roads, traffic often naturally forms a *platoon* – a group of vehicles following each other closely. Here, headway is again essential in maintaining safety of vehicles in the platoon, see [26] for a survey.

In this paper, our objective is understanding the impact of headway on the complexity of train routing and scheduling.[1] While numerous papers study efficient algorithms for both dynamic and static routing problems, research on efficient algorithms that account for headway is scarce. Towards this, we introduce a basic model of train routing with an exogenously defined headway requirement Δ. In our model, we are given a directed network $D = (V, A)$, where each arc a has a travel time $\tau_a \in \mathbb{Z}_{\geq 0}$, and d trains that must travel from a source s to a sink t. Headway constraints require a minimum time separation of Δ between any two trains using the same arc. Our objective is minimising the makespan – the time the last train reaches the sink t. We call this problem Train Makespan Optimization (TMO) and refer to Section 1.1 for a formal description.

Two interesting special cases of TMO arise when the headway is either very large ($\Delta = \infty$) or very small ($\Delta = 1$). In the former case, no arc can be used by two trains, and hence the paths used by the trains must be pairwise arc-disjoint. The problem for $\Delta = \infty$ thus becomes the static problem of finding d arc-disjoint s-t-paths while minimizing the length of the longest path. This is known in the literature as the Min-Max Disjoint Paths (MinMaxDP) problem [20]. In particular, the temporal dynamics, i.e., the scheduling aspect of the problem,

[1] Although, as noted, headway plays an important role in routing other vehicles as well, we use the nomenclature for trains as this is what initially motivated our work.

become irrelevant. Conversely, if $\Delta = 1$, the problem is equivalent to finding an integral *quickest flow over time*, a problem widely studied in traffic networks and transportation [28]. Hence, as Δ reduces, temporal dynamics play a larger role.

The quickest flow over time problem with aggregated arc capacities [11] generalizes the quickest flow over time problem. Here, only a limited amount of flow is allowed to traverse an arc in each time interval of length Δ. By setting these capacities to one, TMO can be seen as an integral special case of the quickest flow over time problem with aggregated arc capacities.

When studying the computational complexity of the problem from a theoretical perspective, one should note that, for large d, the size of the input is $\log(d)$ rather than d. Hence, a polynomial time algorithm must run in time polynomial in $\log(d)$ (and the other parameters). Note that this is not even sufficient time to describe a path for each train. However, we show that optimal solutions can be represented efficiently, by establishing the existence of an optimal *convoy* solution to TMO that can be described by a polynomial number of different paths and does not require waiting on any intermediate nodes. Furthermore, we show that approximating TMO, for any value of Δ, is equivalent to approximating MinMaxDP. We further develop new approximation algorithms for MinMaxDP, which translate to bounds for TMO as well. This work thus provides new results on a classical routing problem and introduces a new model to capture an important parameter in traffic routing and scheduling.

1.1 Model

We are given a directed graph $D = (V, A)$ with travel times $\tau_a \in \mathbb{Z}_{\geq 0}$ for every $a \in A$, a source $s \in V$ as well as a sink $t \in V$. We say $|A| = m$. Furthermore, we are given a minimum headway time $\Delta \in \mathbb{Z}_{\geq 1}$ (between the heads of the trains) and a number d of identical trains which we want to send from s to t. An instance of train routing is thus denoted (D, τ, d, Δ). In the following, we use $[k] := \{1, \dots, k\}$ for $k \in \mathbb{Z}_{\geq 1}$ and $\delta^-(v)$ to denote the set of all arcs entering a node $v \in V$.

A *train routing* $(\boldsymbol{P}, \boldsymbol{\lambda})$ is a tuple consisting of a collection $\boldsymbol{P} = (P_i)_{i\in[d]}$ of s-t-paths P_i, where P_i consists of the sequence of arcs train i traverses on its way from s to t, together with a collection $\boldsymbol{\lambda} = (\lambda_i)_{i\in[d]}$ of *entry functions* $\lambda_i : P_i \to \mathbb{Z}_{\geq 0}$, where $\lambda_i(a)$ denotes the time when the nose of train i enters arc $a \in P_i$.

A train routing $(\boldsymbol{P}, \boldsymbol{\lambda})$ is *feasible* if it complies with the following two constraints:

- $\lambda_i(a) + \tau_a \leq \lambda_i(a')$ for all $i \in [d]$ and any pair of consecutive arcs a, a' on P_i,
- $|\lambda_i(a) - \lambda_{i'}(a)| \geq \Delta$ for all $i, i' \in [d]$ with $i \neq i'$ and $a \in P_i \cap P_{i'}$.

The first constraint implies that the entry function is consistent with the travel times along the path of each train. Note that this allows trains to wait at any node. The second constraint ensures that a feasible routing respects the minimum headway requirement, i.e., if two trains use the same arc, they can only do so with a temporal distance of at least Δ.

Given a feasible train routing $(\boldsymbol{P}, \boldsymbol{\lambda})$, we denote by $C_i(P_i, \lambda_i) := \max_{a\in P_i}(\lambda_i(a) + \tau_a)$ the time when the nose of train i arrives at t. The *makespan* $C_{\max}(\boldsymbol{P}, \boldsymbol{\lambda}) := \max_{i\in[d]} C_i(P_i, \lambda_i)$ is the time when the nose of the latest train reaches the sink t. We consider the Train Makespan Optimization (TMO) problem, where the objective is to find a feasible train routing that minimises the arrival of the last train at t. That is, we want to solve

$$\min C_{\max}(\boldsymbol{P}, \boldsymbol{\lambda}) \quad \text{s.t.} \quad (\boldsymbol{P}, \boldsymbol{\lambda}) \text{ is a feasible train routing.}$$

1.2 The min-max disjoint paths problem

An interesting special case of TMO arises when Δ is sufficiently large compared to the travel times. In that case any (even approximately) optimal train routing sends all trains along disjoint paths – assuming that the network supports d disjoint s-t-paths. This leads us to the MIN-MAX DISJOINT PATHS (MINMAXDP) problem, which can be formalized as follows: Given a directed graph $D = (V, A)$ with two designated vertices $s, t \in V$ and travel times $\tau : A \to \mathbb{Z}_{\geq 0}$, find k pairwise arc-disjoint s-t-paths $P_1, \ldots, P_k$ minimizing the maximum travel time (i.e., makespan) $\max_{i\in[k]} \tau(P_i)$, where we use $\tau(P) := \sum_{a\in P} \tau_a$ to shorten notation for each path P. We call $\boldsymbol{P} = (P_1, \ldots, P_k)$ a *path profile.*

Early work on MINMAXDP has focused on the case where k is constant. Indeed, Li et al. [20] – in the first paper that studied MINMAXDP – showed that, even when $k = 2$, the problem is strongly NP-hard for general digraphs and weakly NP-hard for directed acyclic graphs (DAGs). On the positive side, they also showed that MINMAXDP can be solved in pseudo-polynomial on DAGs when k is constant. By standard rounding techniques, this result can be turned into an FPTAS [13] for the same setting.

Special cases of MINMAXDP when k is part of the input have been studied under various names. Most notably, the MULTI-LEVEL BOTTLENECK ASSIGNMENT (MLBA) problem is equivalent to a special case of MINMAXDP in a DAG consisting of b layers. Dokka and Goerigk [10] recently established a strong inapproximability bound of $\Omega(b)$ for MLBA. In the full version of this article [1, Appendix A], we observe that their construction also implies that MINMAXDP (and, by extension, TMO) in DAGs does not admit approximation factors significantly better than logarithmic in k, unless every problem in NP can be solved in quasi-polynomial time (violating the exponential time hypothesis). This leads to the following theorem.

▶ **Theorem 1.** *There is no $\log^{1-\varepsilon}(k)$-approximation for MINMAXDP in DAGs for any $\varepsilon > 0$, unless NP $\subseteq$ QP.*

Moreover, MINMAXDP in bundle graphs, i.e., digraphs consisting of a sequence of parallel arcs, is equivalent to the so-called *complete* case of MLBA, from which it also inherits strong NP-hardness. For this problem, a long line of approximation algorithms starting in the 1980s [7, 17] has recently culminated in a PTAS by Das and Wiese [9] (formulated for an equivalent scheduling problem).

1.3 Contributions and overview

In the following, we provide an overview of our main contributions and the techniques used to derive these results.

Existence of optimal convoy routings. We show that any instance of TMO has an optimal solution in which any two trains either travel along the same path (with sufficient temporal distance), or follow disjoint paths (Theorem 2 in Section 2). We call such solutions *convoy routings.* In particular, convoy routings have a compact representation that is polynomial in the size of the input, even when the number of trains is exponential in the size of the network. A further consequence is that allowing trains to wait at intermediate nodes does not help to improve the optimal solution value. The proof of this result relies on an uncrossing algorithm that takes an arbitrary train routing and modifies it to ensure that any train that is the first to traverse the final arc of its path is also the first to traverse any other arc it uses. Conflicts, i.e., situations where this property is violated, are resolved iteratively, starting with those closest to the sink, with a potential function argument ensuring eventual termination of the procedure.

A flow-based additive approximation for TMO. We further provide a simple flow-based approximation algorithm for TMO that returns a convoy routing whose makespan is bounded by OPT $+\Delta$ (Theorem 4 in Section 3). This result is achieved by solving an appropriately defined quickest flow over time problem in the underlying network. Note that in practice, the headway Δ is typically much smaller than the total transit time of a train from source to sink, and hence the additive error Δ is small compared to OPT.

Reduction to MinMaxDP. Building on the previous two results, we establish a strong connection between MinMaxDP and TMO by showing that any α-approximation algorithm for MinMaxDP yields a $\max\{1+\varepsilon, \alpha\}$-approximation algorithm for TMO with polynomial runtime in the size of the input and $\frac{1}{\varepsilon}$ (Theorem 5 in Section 4). To achieve this result, we devise a gadget that encodes the choices on the number of trains on each path of a convoy routing into the routing decisions in the network. The size of this gadget is polynomial in the size of the original network and d. It thus provides an exact reduction of TMO to MinMaxDP when the number of trains d is polynomial in the size of the network. When d is large, on the other hand, there must be an arc traversed by a large number of trains and hence OPT is much larger than Δ. In this case, the additive guarantee of OPT $+\Delta$ of our flow-based algorithm for TMO mentioned above translates to a $(1+\varepsilon)$-approximation. As we mentioned previously, for sufficiently large values of Δ, TMO coincides with MinMaxDP. Thus, finding good approximation algorithms for MinMaxDP is not only sufficient but also necessary to obtain good approximations for TMO.

Approximation algorithms for MinMaxDP in series-parallel networks. In the remainder of this paper, we therefore focus on approximation algorithms for MinMaxDP. We show that a natural greedy approach achieves an approximation which is logarithmic in k on series-parallel (SePa) networks (Theorem 8 in Section 5), where k is the number of paths to be constructed. This constitutes the first non-trivial approximation result for MinMaxDP going beyond the aforementioned PTAS for bundle graphs [9], without assuming that either k or the number of layers of the graph is constant.

We prove the approximation guarantee via induction on a series-parallel decomposition of the graph. The key ingredient is showing that the greedy algorithm maintains a solution fulfilling a strong balance condition on the path lengths. Additionally, we provide an alternative analysis of the same approach that yields an upper bound of $\varphi(D)+1$ on the approximation factor, where $\varphi(D)$ denotes the number of changes from series to parallel composition on a root-leaf path in the binary tree describing the SePa-graph. To the best of our knowledge, the parameter $\varphi(D)$, which we demonstrate to be independent of the choice of the decomposition tree, has not been used in the literature before. We believe that this natural parameter can be of independent interest for designing and analysing algorithms for other problems in SePa-graphs. Our alternative analysis implies the classic greedy 2-approximation for bundle graphs by Hsu [17] as a special case. By the aforementioned reduction of TMO to MinMaxDP, the approximation results for MinMaxDP on SePa-graphs, translate to our train routing problem.

2 Existence of optimal convoy routings

Let (D, τ, d, Δ) be an instance of TMO. A *convoy routing* consists of a collection of $k \leq d$ pairwise arc-disjoint s-t-paths $\boldsymbol{P} = (P_1, \dots, P_k)$ in D, together with a vector $\boldsymbol{\sigma} \in \mathbb{Z}^k_{\geq 1}$ with $\sum_{i=1}^k \sigma_i = d$, whose i-th component σ_i denotes the number of trains using path P_i. Note that any such convoy routing $(\boldsymbol{P}, \boldsymbol{\sigma})$ can be turned into a (feasible) train routing $(\boldsymbol{P}, \boldsymbol{\lambda})$ by sending

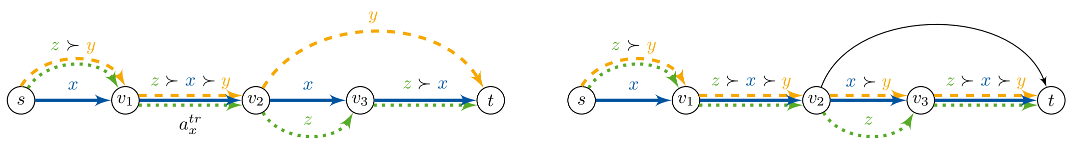

Figure 1 An example iteration of the uncrossing phase. Here $x \succ y$ means that x follows y. Train $x \in L$ is chosen by the uncrossing algorithm. As train y precedes x on a_x^{tr}, it is rerouted to use the same subpath as x from a_x^{tr} to t. The routing after this iteration is shown on the right.

σ_i trains along path P_i in single file with temporal spacing Δ between any two consecutive trains. Thus, the makespan of this train routing is $\max_{i\in[k]} \{\tau(P_i) + (\sigma_i - 1) \cdot \Delta\}$, where $\tau(P_i) := \sum_{a\in P_i} \tau_a$.

▶ **Theorem 2** (Convoy theorem). *Every instance of TMO admits an optimal solution that is a convoy routing.*

Our proof of Theorem 2 is constructive. Note that, since we allow waiting at nodes, there is always an optimal train routing $(\boldsymbol{P}, \boldsymbol{\lambda})$ which is acyclic in the sense that none of the paths contains a cycle. In order to show that any instance of TMO admits an optimal routing which is a convoy solution (i.e., to prove Theorem 2), we take an optimal acyclic train routing $(\boldsymbol{P}, \boldsymbol{\lambda})$, and modify this train routing step by step without increasing the makespan until the routing corresponds to a convoy routing.

The algorithm, which we from now on call **uncrossing algorithm**, proceeds in two phases. First, in the *uncrossing phase*, we reroute trains such that at the end of the phase, the trains that are the first ones to use their respective ingoing arc to the sink t, travel along disjoint paths. We call these trains *leaders*, and the remaining trains *non-leaders*. Afterwards, in the *assignment phase*, we assign every non-leader train a route based on the arc it uses to enter t. In particular, each such train follows its leader from s to t.

We now outline the algorithm in more detail and briefly sketch its correctness. A detailed description can be found in the full version of this article [1, Appendix B].

2.1 The uncrossing phase

We first introduce some notation. For an arc a entering t, i.e., $a \in \delta^-(t)$, we define the *leader* L_a on a as the first train using arc a. We denote by $L = \{L_a\}_{a\in\delta^-(t)}$ the set of leaders. We say that a leader x *has a conflict on arc* $a \in A$ with another train y if y uses some arc a before x. Note that this relation is not symmetric. For a leader $x \in L$, we define the *transition arc* a_x^{tr} as the first arc on which a conflict of x with another train exists when traversing P_x from t to s, i.e., in the reverse direction. Note that a_x^{tr} might not exist, in which case x is the first train on every arc on its path.

In the uncrossing phase, we pick a leader x for which a_x^{tr} exists. We reroute all trains preceding x on a_x^{tr} such that, after rerouting, they all use the same subpath as x from a_x^{tr} to t. The rerouted trains traverse every arc of this a_x^{tr}-t-path with the same temporal distance to train x as they had on arc a_x^{tr}. Hence, the rerouted trains keep their respective pairwise temporal distances on the entire path from a_x^{tr} to t and no new conflicts among the rerouted trains arise. Figure 1 shows an example of such a rerouting step. Note that this rerouting procedure also cannot create new conflicts with other trains as x was the first train on every arc of P_x from a_x^{tr} to t. Hence, the new routing is feasible and does not increase the makespan as x arrives after any rerouted train and the routing of x remains unchanged.

We iterate this procedure with the updated train routing (and the corresponding leaders and transition arcs) until every leader is the first train on all arcs of its path, which implies that all leaders use disjoint paths. In the proof of Lemma 3 below, we establish that this termination criterion is reached after a finite number of uncrossing steps.

2.2 The assignment phase

After successfully uncrossing the paths of all leaders, we receive an updated routing $(\boldsymbol{P}, \boldsymbol{\lambda})$. To achieve a convoy routing $(\boldsymbol{R}, \boldsymbol{\sigma})$, define $\boldsymbol{R} = (P_x)_{x \in L}$ where P_x is the s-t-path used by leader x in $\boldsymbol{P}$. We assign all trains that use the same arc as x to enter t in $(\boldsymbol{P}, \boldsymbol{\lambda})$ to P_x and define σ_x as the resulting number of trains that use P_x. As each train uses exactly one arc in $\delta^-(t)$, this induces a partition of $[d]$ and thus a feasible convoy routing.

To see why the makespan remains unchanged, note the following: Fix an arc $a \in \delta^-(t) \cap P_x$ for some $x \in L$. The first train from our convoy routing $(\boldsymbol{R}, \boldsymbol{\sigma})$ arrives at a no later than the corresponding leader x in $(\boldsymbol{P}, \boldsymbol{\lambda})$. This is because they use the same path, and the train from $(\boldsymbol{R}, \boldsymbol{\sigma})$ departs s immediately. Further, in both routings, the last train entering a must have a minimum temporal distance of $(\sigma_x - 1) \cdot \Delta$ to the first train entering a. In the convoy routing, this bound is tight and thus the assignment phase does not increase the makespan.

2.3 Analysis

By its construction, if the uncrossing algorithm terminates, it returns a convoy routing with a makespan no worse the one of the input routing. Theorem 2 thus follows as a consequence of the following lemma.

▶ **Lemma 3.** *The uncrossing algorithm terminates.*

Proof (sketch). We prove this using a potential function. To define this function, we extend the definition of a^{tr}_y to arbitrary, i.e., also non-leader trains, y. Informally, a^{tr}_y is the last arc on train y's path after which the set (or order) of trains driving before y changes.

The potential of a train y is the number of arcs of P_y from s to a^{tr}_y. We prove the following two statements: (i) When resolving a conflict for some leader x, we decrease its potential, (ii) The potential of all other trains does not increase when resolving a conflict of x. Together, (i) and (ii) imply that the sum of all potentials decreases in every iteration of the uncrossing phase, implying finite termination of this phase and thus of the entire algorithm. The proof of (i) and (ii) involves a careful analysis, which reveals that the transition node of a train is always on its path in the original routing, and a case distinction of possible configurations arising in the uncrossing steps. The details of this analysis can be found in the full version of this article [1, Appendix B]. ◀

3 A flow-based additive approximation for TMO

As mentioned in Section 1, TMO with $\Delta = 1$ reduces to Quickest Flow Over Time in a network with unit capacities. This problem asks for a flow over time of a given value (d units in our case) to be sent from a source to a sink in the shortest time T possible. For an in-depth discussion of flows over time, we refer to [28]. By guessing the optimal makespan T (e.g., by using parametric search [27]), Quickest Flow Over Time can, in turn, be reduced to Maximum Flow Over Time – i.e., sending a maximum amount of flow within a given time horizon. The latter problem can be solved efficiently using a classic result of Ford and Fulkerson [14]: Compute an appropriate static flow, decompose this flow into (disjoint, in the case of unit capacities) paths, and send one unit of flow along each path as long as it still reaches the sink within the time horizon.

A natural extension of this approach to the case $\Delta > 1$ is computing a quickest flow over time sending $\Delta \cdot d$ units in total, and transform it into a train routing by only sending one unit of flow (i.e., one train) every Δ time units along each path. Because, conversely, any routing of d trains also can be extended to a flow over time of value $\Delta \cdot d$ when allowing an extra period of Δ time units beyond its makespan, we obtain the following approximation guarantee for TMO. Again, a detailed proof can be found in the full version of this article [1, Appendix C].

▶ **Theorem 4.** *There is an algorithm that, given a TMO instance, computes in strongly polynomial time a convoy routing with makespan at most* $\mathrm{OPT} + \Delta$*, where* OPT *is the optimal makespan.*

4 Reduction to MinMaxDP

In this section, we examine the relationship between TMO and MinMaxDP. It is easy to see that any α-approximation for TMO implies an α-approximation for MinMaxDP, by choosing $\Delta = \alpha \cdot \sum_{a \in A} \tau_a$. Proving the converse is more involved. We use a bundle graph (see Figure 2) consisting of a sequence of parallel arcs as a gadget, attached to the source node for the MinMaxDP instance, for our reduction.

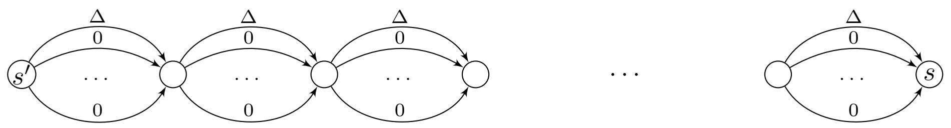

Figure 2 The gadget G_k for solving TMO via MinMaxDP. The gadget consists of a sequence of $(d-k)$ bundles. Each bundle contains an arc with travel time Δ and $k-1$ arcs with travel time 0.

▶ **Theorem 5.** *Let $\mathcal{A}$ be an α-approximation algorithm for MinMaxDP, and let $\mathcal{R}(m)$ denote its runtime dependent on $m := |A|$. There exists a* $\max\{\alpha, (1+\varepsilon)\}$*-approximation for TMO with runtime in* $\mathcal{O}\left(m \cdot \mathcal{R}\left(m^2(2+\frac{1}{\varepsilon})\right)\right)$.

Proof (sketch). We defer a formal proof to the full version of this article [1, Appendix D] and restrict ourselves to providing its intuition for now. First, consider the case $d > m(1+\frac{1}{\varepsilon})$, i.e., when d is large. Then, OPT is much greater than Δ because some arcs are used by more than $\lceil \frac{1}{\varepsilon} \rceil$ trains, and hence the flow-based approximation from Section 3 yields a $(1+\varepsilon)$-approximate solution.

Otherwise, when $d \le m(1+\frac{1}{\varepsilon})$, we utilise the given α-approximation $\mathcal{A}$ for MinMaxDP. We guess the number $k \in [m]$ of disjoint paths in an optimal convoy routing. For that k, we construct an auxiliary digraph D'_k by adding the gadget G_k described in Figure 2 prior to the source node s of the given digraph D. It is easy to see that the minimum makespan for MinMaxDP on D'_k (when k was guessed correctly) and the minimum makespan for the original TMO-instance coincide. Hence, $\mathcal{A}$'s approximation guarantee carries over to TMO. Note that the above case distinction is necessary to bound the size of D'_k polynomially. ◀

▶ Remark 6. Let $\mathcal{C}$ be a class of digraphs that is closed under adding a bundle graph prior to their source s. If $\mathcal{A}$ is an α-approximation algorithm for MinMaxDP on graph class $\mathcal{C}$, then Theorem 5 yields an approximation algorithm for TMO on graph class $\mathcal{C}$.

Bundle graphs and SePa-graphs are examples of such closed classes. Theorem 5 in conjunction with Theorem 1 from [9] yields the following stronger result for bundle graphs.

▶ **Corollary 7.** *There is a PTAS for TMO on bundle graphs.*

5 Approximation algorithm for MinMaxDP in series-parallel graphs

In this section, we describe and analyse an algorithm for approximating MinMaxDP in series-parallel graphs. Before we state our main result, we first provide a formal definition of series-parallel graphs and the parameter $\varphi(D)$ that is part of the approximation guarantee.

Series-parallel graphs. A *series-parallel digraph* $D = (V, A)$ (*SePa-graph*, for short) has a source s and a sink t and consists of either a single arc or can be obtained by a series or parallel composition of two series-parallel subgraphs D' and D''. The series composition $D = D' \odot_s D''$ merges the sink of D' with the source of D'', such that the source of D is the source of D' and the sink of D is the sink of D''. The parallel composition $D = D' \odot_p D''$ merges the two sources of D' and D'' to a single source of D and merges the sinks of D' and D'' to a single sink of D. Note that it is possible to have parallel arcs between two nodes.

The parameter $\varphi(D)$. Any SePa-graph can be described by a *binary decomposition tree*, where the leaves of the tree correspond to the arcs and the internal vertices correspond to either the series (labeled Ⓢ) or the parallel (labeled Ⓟ) composition of their children; see Figure 3 for an example. We can split the tree into arc-, S- or P-components. Such a component is a maximal connected subset of vertices in the binary decomposition tree with the same label. The *root* of a component C is the (unique) vertex $v \in C$ that is closest to the root of the binary decomposition tree. Moreover, we call the vertices that are not in the component C but whose parents are, the children of the component C. We let $\varphi(D)$ denote the maximum number of traversed S-components on a path from the root to a leaf. Note that, even if the binary decomposition tree of a SePa-graph is not unique, the number $\varphi(D)$ is (see full version [1, Appendix E]). Throughout this section, we assume that we are given a fixed binary decomposition tree of the input graph D (such a tree can be constructed in linear time [2]).

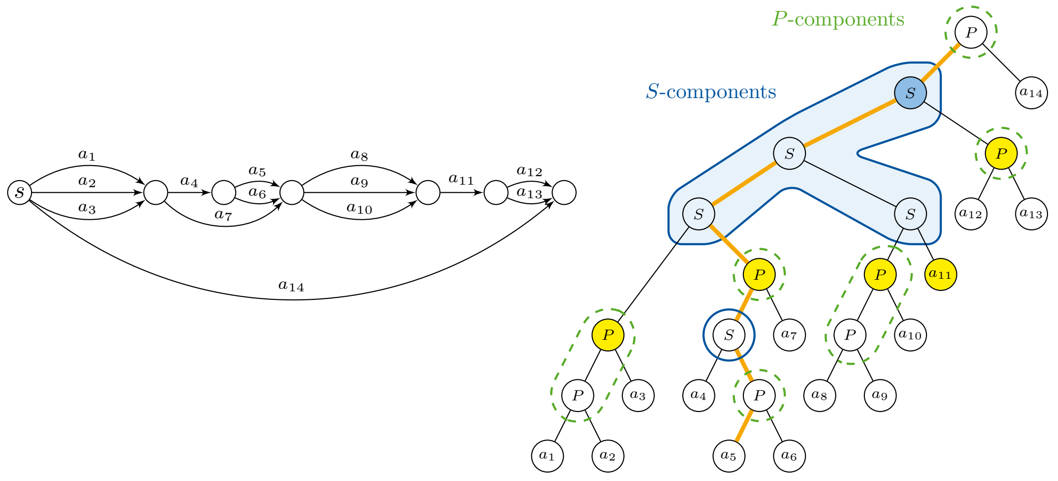

Figure 3 An example of a SePa-graph on the left. A corresponding series-parallel decomposition tree is shown on the right. Consider the large S-component (shaded in light blue), its root is highlighted blue and its children are highlighted yellow. Along the orange (bold) path (from the root to a leaf), we traverse two S-components. Also every other path traverses at most two, so $\varphi(D) = 2$.

Main result and overview. We can now formally state our main result for this section. Note that H_k denotes the k-th harmonic number, i.e., $H_k = \sum_{j=1}^{k} \frac{1}{j} \leq \log_2(k) + 1$.

▶ **Theorem 8.** *For every $\varepsilon > 0$ there is an algorithm that computes a $(\min\{H_k, \varphi(D) + 1\}) \cdot (1+\varepsilon)$-approximate path profile for* MinMaxDP *and runs in time polynomial in m, k, and $\frac{1}{\varepsilon}$.*

A central component of our algorithm is a *greedy composition* procedure that inductively combines path profiles in subgraphs based on the binary decomposition tree. In Section 5.1, we present this greedy composition procedure and several invariants maintained by the procedure that are crucial to our analysis. In particular, we show that if the invariants are fulfilled by the final solution, they imply the desired approximation ratio.

A remaining challenge is that it is not obvious how to initialise the procedure with solutions that satisfy the invariants at the beginning. In particular, some of the invariants are defined with respect to certain properties of the optimal solution and are thus not easy to verify. In Section 5.2, we show how to solve this issue by devising two dynamic programs, each optimising for one of the two terms whose minimum defines the approximation guarantee. Running these two DPs concurrently then yields the guarantee of the theorem (where ε is an error incurred by rounding so that the DPs run in polynomial time).

5.1 Greedy composition procedure

We use the following notation. We write $D' \sqsubseteq D$ for a series-parallel subgraph D' which corresponds to one of the vertices of the decomposition tree of D, i.e., the decomposition tree of D' is the subtree rooted at that vertex. Now consider any such subgraph $D' \sqsubseteq D$ with arc set A', source s', and sink t'. If an s-t-path P in a path profile $\boldsymbol{P}$ of D uses an arc in A', the series-parallel structure of D implies that $P \cap A'$ is an (s', t')-path in D'. We denote the length of a path P in D' by $\tau(P, D') := \sum_{a \in P \cap A'} \tau_a$, and the length of the longest subpath of $\boldsymbol{P}$ in D' by $C_{\max}(\boldsymbol{P}, D') := \max_{P \in \boldsymbol{P}} \tau(P, D')$. Moreover, we call $\theta(\boldsymbol{P}, D')$ the total length of arcs of path profile $\boldsymbol{P}$ in component D', that is $\theta(\boldsymbol{P}, D') := \sum_{P \in \boldsymbol{P}} \tau(P, D')$.

Greedy composition. Let D' and D'' be two SePa-graphs, and let $\boldsymbol{P}'$ and $\boldsymbol{P}''$ be path profiles of D' and D'' consisting of k' and k'' arc-disjoint paths, respectively. For a series composition $D = D' \odot_s D''$, if $k' = k''$, the *greedy composition* of $\boldsymbol{P}$ and $\boldsymbol{P}'$ is the path profile $\boldsymbol{P} = \boldsymbol{P}' \odot_s \boldsymbol{P}''$ of D that is derived by combining the longest path in $\boldsymbol{P}'$ with the shortest path in $\boldsymbol{P}''$, the second-longest path in $\boldsymbol{P}'$ with the second-shortest path in $\boldsymbol{P}''$, and so on. Note that this construction is only defined for $k' = k''$. For a parallel composition $D = D' \odot_p D''$, the greedy composition is given by $\boldsymbol{P} = \boldsymbol{P}' \odot_p \boldsymbol{P}''$ of D, where $\boldsymbol{P}$ simply consists of the union of the $k' + k''$ paths in $\boldsymbol{P}'$ and $\boldsymbol{P}''$.

We now discuss three properties that are maintained by the greedy composition procedure described above if they are fulfilled by the path profiles that are being combined. To define these properties, we fix an arbitrary optimal path profile $\boldsymbol{P}^*$ in the graph D that defines our MinMaxDP instance. The first property we consider is consistency.

Consistency. We call a path-profile $\boldsymbol{P}$ *consistent* with $\boldsymbol{P}^*$ in $D' \sqsubseteq D$ if $\boldsymbol{P}$ and $\boldsymbol{P}^*$ use the same number of paths in D' and if

$$\theta(\boldsymbol{P}, D') \leq \theta(\boldsymbol{P}^*, D'). \tag{1}$$

Note that consistency is maintained by greedy composition, as the arcs used by the disjoint paths in the composed profile are exactly the union of the paths in the two profiles that are being combined. We now define the second property, balancedness, and show that together with consistency it implies the first of our two approximation bounds.

Balancedness. Let $\boldsymbol{P}$ be a path profile, and let $D' \sqsubseteq D$ be traversed by $k' \leq k$ paths from $\boldsymbol{P}$. Let $p_1 \geq \ldots \geq p_{k'}$ be the lengths of those paths in D'. Then $\boldsymbol{P}$ is *balanced* in D' if

$$\frac{1}{i}\left(\sum_{j=1}^{i} p_j\right) - p_{i+1} \leq C_{\max}(\boldsymbol{P}^*, D') \qquad \text{for all } i \in [k'-1]. \tag{2}$$

We first establish that balancedness – when combined with consistency – is indeed maintained by greedy composition. While this is straightforward for parallel composition, the proof for series composition requires a more involved analysis that carefully combines (1) and (2). The details of this analysis are given in the full version of this article [1, Appendix F.1].

▶ **Lemma 9.** *Let $\odot \in \{\odot_s, \odot_p\}$ and let $D^{(1)} \odot D^{(2)} = D'$. If $\boldsymbol{Q}$ and $\boldsymbol{R}$ are consistent and balanced path profiles in $D^{(1)}$ and $D^{(2)}$, respectively, then the greedy profile $\boldsymbol{P} = \boldsymbol{Q} \odot \boldsymbol{R}$ is a consistent and balanced path profile in D'.*

Crucially, the following lemma reveals that if the final solution is balanced and consistent with $\boldsymbol{P}^*$, it is a logarithmic approximation for the latter. The lemma follows from an averaging argument over (2) for $i \in [k-1]$, which can be found in the full version of this article [1, Appendix F.2].

▶ **Lemma 10.** *Let $\boldsymbol{P}$ be a consistent and balanced path profile in D. Then $C_{\max}(\boldsymbol{P}, D) \leq H_k \cdot C_{\max}(\boldsymbol{P}^*, D)$.*

We now introduce the third property, φ-boundedness, which is likewise maintained by greedy composition and implies the second bound on the approximation guarantee when fulfilled by the final solution.

φ-boundedness. We call a path profile $\boldsymbol{P}$ *φ-bounded* in $D' \sqsubseteq D$ if

$$C_{\max}(\boldsymbol{P}, D') \leq (\varphi(D') + 1) \cdot C_{\max}(\boldsymbol{P}^*, D). \tag{3}$$

Note that in this definition, the left-hand side refers to the length of the longest path of $\boldsymbol{P}$ in the subgraph $D' \sqsubseteq D$, whereas the right-hand side refers to the optimal objective value for MinMaxDP on the entire graph D. The following lemma establishes that consistency and φ-boundedness are maintained by the greedy composition when executing all compositions that correspond to one component of the decomposition tree.

▶ **Lemma 11.** *Let $D' \sqsubseteq D$ be a subgraph corresponding to a root vertex of an S-component or P-component, respectively. Let $D^{(1)}, \ldots, D^{(i)}$ be the graphs corresponding to the children of the component. Further, let $\boldsymbol{P}^{(j)}$ be a consistent and φ-bounded path profile in $D^{(j)}$ for $j \in [i]$. Then, the greedy series, respectively parallel, composition $\boldsymbol{P}$ of $\boldsymbol{P}^{(1)}, \ldots, \boldsymbol{P}^{(i)}$ is a consistent and φ-bounded path profile in D'.*

The proof of this lemma involves an induction over the components of the decomposition tree of D and can be found in the full version of this article [1, Appendix F.3].

5.2 Dynamic programs

The results in the preceding section imply that if we start from path profiles for the leaves of the decomposition tree of D that are consistent with an optimal solution and balanced or φ-bounded, respectively, then iteratively applying greedy composition will yield a path profile for D that is consistent and balanced or φ-bounded, respectively. In particular, such a profile then yields an H_k-approximation or $(\varphi(D) + 1)$-approximation.

However, since we do not know the optimal solution, we do not know which path profiles fulfil these requirements. A natural approach seems to be to restrict ourselves to a solution with minimum total length (i.e., an integral minimum-cost s-t-flow of value k). However, there is a difficult trade-off between minimising the length of a longest path or minimising the total length of the solution in a subgraph, as the following example shows.

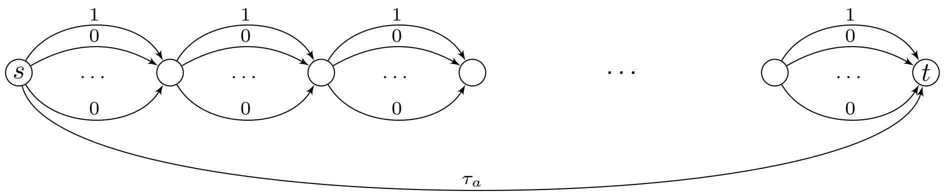

Figure 4 The depicted graph D is built as follows. Let D' be a graph with k parallel arcs (one arc with length 1 and $k-1$ arcs with length 0). We obtain D by connecting k copies of D' in series plus adding an additional arc from s to t with length τ_a.

▶ **Example 12.** Consider the graph D' depicted in Figure 4. If we set $\tau_a := k-1$, the path profile with minimal total length consists of $k-1$ paths of length 0 and one path with length $k-1$, leading to the objective value $k-1$. However, the optimal path profile consists of k paths all having length 1. On the other hand, assume that $\tau_a := 1+\varepsilon$ and create a graph D that consists of k copies of D' composed in series. In this case, solving the MinMaxDP in each subgraph D' results in all paths having length 1 inside each copy of the example, yielding to the objective value k when combining them in series. However, the optimal path profile uses exactly one arc of length $1+\varepsilon$ in each path and otherwise only length-0 arcs, yielding an objective value of $1+\varepsilon$. Thus, an algorithm which locally only considers the length of the longest path in the profile, would be a factor of k apart form the optimal value.

DP framework. In the following, we show how to resolve this issue by means of two dynamic programs. In both cases, our dynamic programming table will have the following attributes per cell:

- a subgraph $D' \sqsubseteq D$,
- the number of paths $k' \in [k]$ used in D',
- the total length $\theta' \in \Theta$ of the path profile in D'.

Recall that the total length of a path profile $\boldsymbol{P}$ in subgraph D' was defined as the sum of the lengths of the paths in D', i.e., $\theta(\boldsymbol{P}, D') := \sum_{P \in \boldsymbol{P}} \sum_{a \in A[D'] \cap P} \tau_a$. We let Θ denote the set containing all possible values that might appear as total costs.

To fill the cells of the dynamic programming table, we start with the subgraphs consisting of single arcs, i.e., $D' = (\{v,w\}, \{a = (v,w)\})$. Here, the non-empty entries of the table corresponding to D' are $(D', 1, \tau_a) = \{(a)\}$ and $(D', 0, 0) = \{\emptyset\}$, while all remaining entries corresponding to D' are empty (which is different from containing an empty path profile).

Given a cell $(D' \odot D'', \tilde{k}, \tilde{\theta})$, we combine all path profiles $\boldsymbol{P}'$ and $\boldsymbol{P}''$ of D' and D'' consisting of k' and k'' paths of total length θ' and θ'', respectively, if and only if $\tilde{\theta} = \theta' + \theta''$ and $\tilde{k} = k' + k''$ (in case of a parallel composition), or $\tilde{k} = k' = k''$ (in case of a series composition). We build path profiles for $D' \odot D''$ according to the greedy composition of the path profiles for D' and D'' as described above.

With this procedure, we obtain multiple candidates for a path profile in table entry $(D' \odot D'', \tilde{k}, \tilde{\theta})$. Below, we analyse two different strategies to pick a *good* candidate that lead to different approximation guarantees. After filling the complete table of the dynamic program, there is a path profile $\boldsymbol{P}_\theta$ with k paths in D for each possible total-cost value $\theta \in \Theta$. In the last step, the algorithm will choose the path profile for which the longest path is as short as possible, i.e. the algorithm chooses $\arg\min_{\theta \in \Theta} C_{\max}(\boldsymbol{P}_\theta, D)$.

DP strategy for balancedness. Our first strategy for selecting the candidates for the intermediate steps aims for balancedness of the solutions and is formalised in the following lemma, that reveals that the strategy yields an H_k-approximation.

▶ **Lemma 13.** *The dynamic program which always selects the candidate $\boldsymbol{P}$ with k' paths of lengths $p_1 \geq \dots \geq p_{k'}$ that minimises*

$$\max_{i \in \{1,\dots,k'-1\}} \left\{ \frac{1}{i} \sum_{j=1}^{i} p_j - p_{i+1} \right\}, \tag{4}$$

computes a solution with $DP \leq H_k \cdot \mathrm{OPT}$, where DP is the objective value of the computed solution and OPT *is the objective value of an optimal solution.*

The proof of this lemma is by induction on the table entries of the optimal solution and by using Lemma 9 to show that we always have a balanced candidate. Together with the choice (4), the lemma follows. A formal proof can be found in the full version of this article [1, Appendix F.4].

DP strategy for boundedness. Our second candidate selection strategy aims at φ-boundedness of the intermediate solutions. This is formalised in the following lemma, which shows that this strategy yields a $(\varphi(D) + 1)$-approximation.

▶ **Lemma 14.** *The dynamic program that always selects the candidate $\boldsymbol{P}$ with k' paths of lengths $p_1 \geq \dots \geq p_{k'}$ that minimises*

$$p_1 - p_{k'} \quad \text{in case of a series combination, and} \tag{5}$$

$$p_1 \quad \text{in case of a parallel composition} \tag{6}$$

computes a solution with $DP \leq (\varphi(D) + 1) \cdot \mathrm{OPT}$, where DP is the objective value of the computed solution and OPT *is the objective value of an optimal solution.*

Similarly to the proof of Lemma 13 we follow the table entries of an optimal solution. However, here we need to consider a component of the decomposition tree of D at once and show that the entries for the subgraphs corresponding to segment roots are φ-bounded. The formal proof of this lemma can be found in the full version of this article [1, Appendix F.5].

Running time. From the size of the dynamic programming table, we can obtain the bound on its running time formalised in the lemma below. By a standard rounding approach, we can further ensure that the number of possible total-cost values $|\Theta|$ is polynomial in the size of the graph at a loss of a $(1 + \varepsilon)$-factor in the approximation ratio, obtaining the desired polynomial running time; see the full version of this article [1, Appendix F.6] for details.

▶ **Lemma 15.** *The dynamic program runs in time $\mathcal{O}(m \cdot k^4 \cdot |\Theta|^2)$.*

6 Conclusion

In this paper, we introduced TMO as a natural optimisation problem that captures some of the fundamental challenges of routing trains with a headway constraint. By showing the existence of optimal train routings with a convoy structure, we linked the approximability of the problem to that of MinMaxDP, for which we then provided new approximation results in SePa-graphs. Several interesting questions arise from our work:

- Is it possible to extend the logarithmic approximation guarantee for MinMaxDP in SePa-graphs to arbitrary DAGs (where it would be best possible due to Theorem 1)?
- Conversely, can the inapproximability result from [10] for DAGs be extended to SePa-graphs (again implying that the greedy approach is best possible in those graphs)?
- Which of the results presented in this paper can be extended to the setting in which trains might have distinct sources and sinks? Note that for general DAGs even finding disjoint paths for multiple source-sink pairs is NP-hard [12] and that this still holds true in undirected SePa-graphs [23]; however, this hardness result does not carry over to directed SePa-graphs, so there is hope that our approximation results for those graphs extend to the multi commodity case.

References

1 Umang Bhaskar, Katharina Eickhoff, Lennart Kauther, Jannik Matuschke, Britta Peis, and Laura Vargas Koch. On the Approximability of Train Routing and the Min-Max Disjoint Paths Problem, 2025. doi:10.48550/arXiv.2507.03687.

2 Hans L. Bodlaender and Babette de Fluiter. Parallel algorithms for series parallel graphs. In Josep Díaz and Maria J. Serna, editors, *Algorithms — ESA '96*, Lecture Notes in Computer Science, pages 277–289, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. doi:10.1007/3-540-61680-2_62.

3 Ralf Borndörfer, Torsten Klug, Thomas Schlechte, Armin Fügenschuh, Thilo Schang, and Hanno Schülldorf. The freight train routing problem for congested railway networks with mixed traffic. *Transportation Science*, 50(2):408–423, 2016. doi:10.1287/trsc.2015.0656.

4 Valentina Cacchiani, Alberto Caprara, and Paolo Toth. Scheduling extra freight trains on railway networks. *Transportation Research Part B: Methodological*, 44(2):215–231, 2010. doi:10.1016/j.trb.2009.07.007.

5 Gabrio Caimi, Leo Kroon, and Christian Liebchen. Models for railway timetable optimization: Applicability and applications in practice. *Journal of Rail Transport Planning & Management*, 6(4):285–312, 2017. doi:10.1016/j.jrtpm.2016.11.002.

6 Malachy Carey. Ex ante heuristic measures of schedule reliability. *Transportation Research Part B: Methodological*, 33(7):473–494, 1999. doi:10.1016/S0191-2615(99)00002-8.

7 Edward G. Coffman, Jr. and Mihalis Yannakakis. Permuting elements within columns of a matrix in order to minimize maximum row sum. *Mathematics of Operations Research*, 9(3):384–390, 1984. doi:10.1287/moor.9.3.384.

8 José Correa, Carolina Osorio, Laura Vargas Koch, David Watling, and Svenja Griesbach. Dynamic traffic models in transportation science (dagstuhl seminar 24281). *Dagstuhl Reports*, 14(7):1–16, 2024. doi:10.4230/DagRep.14.7.1.

9 Syamantak Das and Andreas Wiese. On minimizing the makespan when some jobs cannot be assigned on the same machine. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *LIPIcs*, pages 31:1–31:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ESA.2017.31.

10 Trivikram Dokka and Marc Goerigk. Multi-level bottleneck assignment problems: Complexity and sparsity-exploiting formulations. *Computers & Operations Research*, 154:106213, 2023. doi:10.1016/j.cor.2023.106213.

11 Daniel Dressler and Martin Skutella. An FPTAS for flows over time with aggregate arc capacities. In Klaus Jansen and Roberto Solis-Oba, editors, *International Workshop on Approximation and Online Algorithms*, volume 6534 of *Lecture Notes in Computer Science*, pages 106–117. Springer, 2010. doi:10.1007/978-3-642-18318-8_10.

12 Shimon Even, Alon Itai, and Adi Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5(4):691–703, 1976. doi:10.1137/0205048.

13 Rudolf Fleischer, Qi Ge, Jian Li, and Hong Zhu. Efficient algorithms for k -disjoint paths problems on dags. In Ming-Yang Kao and Xiang-Yang Li, editors, *Algorithmic Aspects in Information and Management*, volume 4508 of *Lecture Notes in Computer Science*, pages 134–143. Springer Berlin Heidelberg, 2007. doi:10.1007/978-3-540-72870-2_13.

14 Lester Randolph Ford, Jr. and Delbert Ray Fulkerson. Constructing maximal dynamic flows from static flows. *Operations research*, 6(3):419–433, 1958. doi:10.1287/opre.6.3.419.

15 T. Godwin, Ram Gopalan, and T. T. Narendran. Freight train routing and scheduling in a passenger rail network: Computational complexity and the stepwise dispatching heuristic. *Asia-Pacific Journal of Operational Research*, 24(4):499–533, 2007. doi:10.1142/S0217595907001358.

16 Rebecca Haehn, Erika Ábrahám, and Nils Nießen. Freight train scheduling in railway systems. In Holger Hermanns, editor, *Measurement, Modelling and Evaluation of Computing Systems*, volume 12040, pages 225–241. Springer, 2020. doi:10.1007/978-3-030-43024-5_14.

17 Wen-Lian Hsu. Approximation algorithms for the assembly line crew scheduling problem. *Mathematics of Operations Research*, 9(3):376–383, 1984. doi:10.1287/moor.9.3.376.

18 Fahimeh Khoshniyat. *Optimization-Based Methods for Revising Train Timetables with Focus on Robustness*. Linköping University Electronic Press, 2016. doi:10.3384/lic.diva-132920.

19 Leo G. Kroon, H. Edwin Romeijn, and Peter J. Zwaneveld. Routing trains through railway stations: complexity issues. *European Journal of Operational Research*, 98(3):485–498, 1997. doi:10.1016/S0377-2217(95)00342-8.

20 Chung-Lun Li, S. Thomas McCormick, and David Simchi-Levi. The complexity of finding two disjoint paths with min-max objective function. *Discrete Applied Mathematics*, 26(1):105–115, 1990. doi:10.1016/0166-218X(90)90024-7.

21 Richard M. Lusby, Jesper Larsen, Matthias Ehrgott, and David Ryan. Railway track allocation: models and methods. *OR Spectrum*, 33(4):843–883, 2011. doi:10.1007/s00291-009-0189-0.

22 Shi Mu and Maged Dessouky. Scheduling freight trains traveling on complex networks. *Transportation Research Part B: Methodological*, 45(7):1103–1123, 2011. doi:10.1016/j.trb.2011.05.021.

23 Takao Nishizeki, Jens Vygen, and Xiao Zhou. The edge-disjoint paths problem is NP-complete for series–parallel graphs. *Discrete Applied Mathematics*, 115(1):177–186, 2001. doi:10.1016/S0166-218X(01)00223-2.

24 Bianca Pascariu, Marcella Samà, Paola Pellegrini, Andrea d'Ariano, Joaquin Rodriguez, and Dario Pacciarelli. Formulation of train routing selection problem for different real-time traffic management objectives. *Journal of Rail Transport Planning & Management*, 31:100460, 2024. doi:10.1016/j.jrtpm.2024.100460.

25 Thomas Rothvoss, Laura Sanità, and Robert Weismantel. Combinatorial optimization. Technical Report 50/2024, Mathematisches Forschungsinstitut Oberwolfach, 2024. doi:10.14760/OWR-2024-50.

26 Sedigheh Sadeghhosseini. *Time headway and platooning characteristics of vehicles on interstate highways*. University of Illinois at Urbana-Champaign, 2002. URL: https://hdl.handle.net/2142/83185.

27 Masahide Saho and Maiko Shigeno. Cancel-and-tighten algorithm for quickest flow problems. *Networks*, 69(2):179–188, 2017. doi:10.1002/net.21726.

28 Martin Skutella. An introduction to network flows over time. In William Cook, László Lovász, and Jens Vygen, editors, *Research Trends in Combinatorial Optimization: Bonn 2008*, pages 451–482. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-540-76796-1_21.