



# Railway timetabling: a maximum bottleneck path algorithm for finding an additional train path

Fredrik Ljunggren<sup>1,2</sup> · Kristian Persson<sup>1,3</sup> · Anders Peterson<sup>1</sup> ·  
Christiane Schmidt<sup>1</sup> 

Accepted: 31 August 2020 / Published online: 27 September 2020  
© The Author(s) 2020

## Abstract

We present an algorithm to insert a train path in an existing railway timetable close to operation, when we want to affect the existing (passenger) traffic as little as possible. Thus, we consider all other trains as fixed, and aim for a resulting train path that maximizes the bottleneck robustness, that is, a train path that maximizes the temporal distance to neighboring trains in the timetable. Our algorithm is based on a graph formulation of the problem and uses a variant of Dijkstra's algorithm. We present an extensive experimental evaluation of our algorithm for the Swedish railway stretch from Malmö to Hallsberg. Moreover, we analyze the size of our constructed graph.

**Keywords** Railway timetabling · Robust train path · Bottleneck train path · Network algorithm · Freight transportation

---

This work subsumes the extended abstract that appeared in the Proceedings of the 14th International Conference on Advanced Systems in Public Transport and TransitData 2018 (Ljunggren et al. 2018). This research is a result of a collaboration between Linköping University and Trafikverket, and part of the EU H2020 Shift2Rail subprojects ARCC (Grant no. 730813) and FR8HUB (Grant no. 777402), and partially funded by Trafikverket (Dnr TRV 2016/75881, and Dnr TRV 2017/68055). The authors are grateful to Magnus Wahlborg (Trafikverket) for fruitful discussions and Martin Aronsson (SICS RISE) for timetable data.

---

✉ Christiane Schmidt  
christiane.schmidt@liu.se

Fredrik Ljunggren  
fredrik.ljunggren@trafikverket.se

Kristian Persson  
kristian.persson@sweco.se

Anders Peterson  
anders.peterson@liu.se

<sup>1</sup> Communications and Transport Systems, ITN, Linköping University, Norrköping, Sweden

<sup>2</sup> Present Address: Trafikverket, Stockholm, Sweden

<sup>3</sup> Present Address: Sweco, Stockholm, Sweden

## 1 Introduction

Over the last decades, both passenger traffic and freight traffic volumes in Sweden increased—from 1996 to 2016 by 82% (from about 7000 to 12800 passenger kilometers) and by 23% (from about 55 to about 68 million ton-kilometers), respectively, see (Grimm 2012; Trafikanalys 2017a, b). In all of Europe, freight traffic volume increases, and while the volume transported via railway within the EU has stagnated over the last years, see Eurostat (2017), the European commission sees the potential to revitalize rail freight (Commission of the European communities 2007): road congestion and the high oil price make road transport more expensive, railway transport is much safer, and increasing environmental concerns favor railway over road traffic. On the other hand, today's wagon load traffic is ineffective and marshalling complicates the transport—these problems need to be alleviated to comply with a political vision of increased freight traffic volumes.

One problem for the freight traffic is the long planning horizon. Today, an annual timetable is constructed, for which train path requests should be submitted up to more than 1.5 years in advance. Freight train operators often have a hard time to estimate transport volumes, and need to make frequent updates of the timetable ad-hoc. Further, the shunting plans for yards and terminals are normally made on a daily basis.

As a consequence, a typical real-world scenario is a train completed at the departing yard one or more hours ahead of its scheduled departing time. Sending it off earlier would not only release capacity at the shunting yard and in the network, but also give a valuable buffer for subsequent delays. Both the infrastructure manager (IM) and the railway undertaking (RU) are interested in this. Today, such a request is answered manually by looking a few stations ahead, and if the freight train will not interrupt operations on this limited considered stretch, an earlier departure will be permitted. This procedure hardly takes into account the already congested rail network, where freight traffic interacts with passenger traffic with much higher requirements on punctuality, nor does it ensure that the receiving yard or terminal has capacity to accommodate the train when arriving.

To make sure both that the existing (passenger) traffic is not affected by the train path of the freight train and that the freight train actually obtains a feasible train path to its destination, it is essential to optimize the process. We aim at a method for finding that train path, which minimizes the risk that other traffic in the deregulated market is affected. If that train path is still insufficiently robust, we advise the IM to reject the late application.

In this paper, we propose an algorithm that computes a maximum robust train path for inserting an additional train into an existing railway timetable (at a time). That is, the algorithm computes the train path with maximum temporal distance to all other existing trains, in particular, to the two neighboring trains on each line section, in the timetable (we further discuss the relation between robustness and the risk for disturbances in Sect. 2.1).

We transform our problem to an equivalent network formulation and use a variant of Dijkstra's algorithm to compute the maximum robust train path by

computing the maximum bottleneck path. Nodes are possible departing times at stations, edges represent feasible spatial or temporal movements, where edge costs give the temporal distance to the surrounding traffic. The objective of maximum robustness (plus using a graph representation of the existing timetable) is the main novelty of this paper.

This algorithm might be used for a freight train in the scenario described above, with the objective to influence the already scheduled trains as little as possible. Moreover, it may also be used for special passenger trains, which need to be added, but where the insertion follows the same objective of not disturbing existing trains. We assume that all other trains are fixed, that is, their train paths may not be altered.

In addition, because of our network formulation,—though we do not do so in the experiments in this paper—we can easily consider alternative routes within or between stations, which often causes problems when using Mixed Integer Linear Program (MILP) formulations. Our graph formulation inserts edges between stations, whenever inserting a train path between two adjacent trains on that line section is possible. Here, it does not play a role if all edges use the same sequence of stations or not. Hence, a different geometry, that is, an alternative route via different stations, can easily be integrated in the graph formulation, and does not influence the algorithm.

In general, several optimality criteria could be considered: we could aim for the earliest possible arrival time of the inserted train, or the shortest possible runtime of the train between the starting and end station, etc.

The remainder of this paper is organized as follows: In Sect. 2 we present related work, Sect. 3 gives necessary notation for our problem. We present an algorithm to compute the maximum bottleneck robust train path for the inserted freight train in Sect. 4. In Sect. 5 we present detailed experiments for the Swedish railway stretch between Malmö and Hallsberg for our algorithm and analyze its runtime in Sect. 6. We give an improvement on the actual path selection of our algorithm in Sect. 7, before we conclude in Sect. 8.

## 2 Related work

In this section we give a literature overview on timetable planning with focus on robustness, followed by some few examples from the literature on methods for inserting an additional train in an existing timetable.

### 2.1 Robust railway timetabling

Railway timetabling is a problem that has been extensively studied for many years, see e.g. Hansen and Pachl (2014), Harrod (2012), or Törnquist (2006) for an overview. Normally a complete timetable is constructed in one single timetabling process, where the infrastructure capacity is assigned to rail undertakings and maintenance contractors according to some principles. Often, the problem is formulated as a MILP, where ordering variables are integer (binary), and time is

continuous. When the timetable is assumed to have a cyclic structure, as is often the case in passenger traffic, a Period Event Scheduling Problem (PESP) can be formulated, see, e.g., Liebchen (2008).

Models applied closer to real-time are typically focused on rescheduling in case of disturbances and disruptions. The aim then often is formulated as to quickly re-obtain a feasible timetable of sufficient quality. For an overview of models of this kind, we refer to, e.g., the survey article by Cacchiani et al. (2014). Andersson et al. (2013) and Khoshniyat and Peterson (2017) show two examples of how rescheduling methods can be used on a more tactical level to redistribute available runtime margin and buffer time to increase stability.

Improving stability, however, may impair other timetable qualities, such as the average speed, heterogeneity and capacity utilization. UIC (2004) illustrate, in a frequently cited “Figure 1—Capacity balance”, how railway capacity depends on the balance of these four factors. Zhang (2015) has developed a mathematical programming framework for line capacity studies, which aims at the minimization of heterogeneity and running time as well as the maximization of reliability and analyses Pareto optimal solutions in an application for the Beijing–Shanghai high-speed railway.

When we are inserting a new train into an existing timetable, capacity utilization will inevitably increase. Heterogeneity is an expression for the spread in travel speed among the trains in the timetable, and the effect of inserting an additional train depends on the difference in speed profile to adjacent trains sharing the same infrastructural resources. Homogenizing, i.e., decreasing heterogeneity, is considered to be desirable in timetabling. An attempt to quantify the heterogeneity in a timetable is made by Vromans et al. (2006). To the best of our knowledge this measure solely has never been used as target in timetable optimization problems.

In our model, the train paths for already existing trains are assumed to be fixed, and the new path must be inserted in a way that minimizes the expected impact to other traffic. Therefore, we want to find the most robust train path, and if that is not sufficiently robust, we may consider rejecting the additional request. The price for the robustness is a longer travel time, which, however, is bounded by time-windows for departure from origin and arrival to destination. Robustness might be defined in various ways, see e.g. Kroon et al. (2008). Goverde and Hansen (2013) distinguish between timetable stability, feasibility, robustness, and resilience. They define *timetable robustness* as “the ability of a timetable to withstand design errors, parameter variations, and changing operational conditions” and note that robustness depends on the stochastic behavior of all the underlying processes.

To assess the robustness, we must make some assumption on the stochasticity of the disturbance processes, which the timetable should be robust against. If we assume the disturbances to be uniformly distributed, identifying the train path with maximum bottleneck size, will indeed also give the most robust train path. This result follows from Goerigk and Schöbel (2014), and is essential in the motivation of our algorithm.

## 2.2 Methods for inserting a train in an existing timetable

Various authors also considered adding a new train to an existing timetable, amongst others (Burdett and Kozan 2009). Flier (2009) (see also Flier 2011) present a shortest path model using a time-expanded graph, which integrates linear regression models based on extensive historical delay data, that gives Pareto optimal train paths w.r.t. travel time and risk of delay.

Ingolotti et al. (2004) consider adding new trains to a heterogeneous, heavily loaded railway network, and aim to minimize the traversal time for each additional train.

Jiang et al. (2017) also add (passenger) trains: they consider a highly congested double track network line and aim to meet the passenger demand with the added trains; to that end, they can make limited alterations to the dwelling times for other, already scheduled, trains. They propose a heuristic to solve this problem.

Cacchiani et al. (2010) also consider the problem of inserting a single freight train into an existing schedule of fixed passenger trains. They assume that the operator specifies an ideal time table that the IM can modify, which also includes the use of a different path. Cacchiani et al. aim to add the maximum number of new freight trains, such that their time table is as close as possible to the ideal one. To do so, they use a heuristic algorithm based on a Lagrangian relaxation of an Integer Linear Program (ILP).

To the best of our knowledge, the objective of maximizing the robustness of a train path for a train to be inserted in an existing timetable has never been considered. Moreover, other approaches either allow to adapt existing trains (which we aim to circumnavigate due to our application) and/or use heuristic approaches (instead of algorithms that guarantee to find the optimal solution).

## 3 Freight train insertion problem

In this section we formally define our problem.

### Freight Train Insertion Problem (FTI):

Given A freight train  $\varphi$ ; a starting station  $s_0$  and an end station  $s_M$ ; a desired route for  $\varphi$  from  $s_0$  to  $s_M$ , given by a sequence of stations  $\mathcal{S}_\varphi = (s_0, s_1, \dots, s_M)$ , when clear from content, we only refer to the stations by  $0, \dots, M$ ; time windows  $w_s = [w_s^a, w_s^d]$  for earliest arrival and latest departure of  $\varphi$  at station  $s$  for all, or some of the, stations  $s \in \mathcal{S}_\varphi$ ; the train-specific running times  $t_{i,i+1}$  for train  $\varphi$  from station  $i$  to  $i+1 \forall i \in \{0, \dots, M-1\}$ ; the timetable of all trains in the set  $\mathcal{T}$ : all trains that run in  $[w_0^a - \varepsilon_1, s_M^d + \varepsilon_2]$ , where  $\varepsilon_i$  is defined such that the trains that depart before or arrive after a possible path for  $\varphi$  at any station are included; the required safety distance  $c_{\tau,\varphi,s}$  (sometimes referred to as headway with a certain buffer or clearance time) between any other train  $\tau$  and train  $\varphi$  at station  $s$ ; and an objective function  $\mathcal{F}$ .

Remarks: In this paper, we define the train-specific running times by the trains' respective maximum speed and some driving margin, i.e., we do not include the option of letting trains run slower. For  $s=0$  time window  $w_s$  describes all possible departure times from the origin, and for  $s=M$  the time window describes all

possible arrival times at the destination. A time window at an intermediate station may also be given, e.g., due to staff schedule or wagon coupling/uncoupling.

Find A train path for  $\varphi$  given by arrival times  $a_{\varphi,s}$  and departure times  $d_{\varphi,s}$  for all stations in  $\mathcal{S}_\varphi$  within the time windows  $w_s = [w_s^a, w_s^d] \forall s \in \mathcal{S}_\varphi$ , that meets the distances  $c_{\tau,\varphi,s} \forall \tau \in \mathcal{T}, \forall s \in \mathcal{S}$ , and the  $t_{i,i+1} \forall i \in \{0, \dots, M - 1\}$ , and optimizes  $\mathcal{F}$ .

In this paper, we use *robustness* as  $\mathcal{F}$ .

## 4 Maximum bottleneck robust train path

In this section, we describe how we compute a maximum robust train path for a freight train close to operation, given that the train paths of all other trains are fixed. That is, we solve FTI with the objective to maximize the robustness. In a first step, we transform our problem to an equivalent graph problem, see Sect. 4.1. We then show that we can use a variant of Dijkstra’s algorithm to compute the maximum robust train path, see Sect. 4.2.

### 4.1 From timetable to input graph

We generate a graph with a set of vertices,  $V_s$ , for each station, where a vertex represents a feasible departure interval at that station. We insert inter-station edges, where the robustness of an edge is always determined by the earliest and latest possible departure from the vertex at its originating station. That is, an edge, representing a feasible train path from station  $s$  to  $s + 1$  gets assigned a weight of the time difference between earliest and latest departure from station  $s$ . As we assume linear train paths, the minimum robustness will always be assumed at a station, and for any path (selected by edges) the edge weights will reflect the robustness intervals along the complete train path. Moreover, we introduce intra-station edges that represent waiting at a station for  $\varphi$ , which is important to allow overtaking, with robustness of infinity, as the robustness of any train path is not limited by waiting at a station. The vertices and edges define our graph as  $G = (V, E)$  with  $V = \cup_{s=0}^M V_s$ .

In the following, we give a formal description of our input graph creation: We generate a set of vertices  $V_s$  for each station  $s \in \mathcal{S}_\varphi$ . Let  $\tau_k, \tau_{k+1} \in \mathcal{T}$  be two trains that depart from  $s$  consecutively. We add a vertex to  $V_s$  that represents the gap between  $\tau_k$  and  $\tau_{k+1}$  if (a)  $(d_{\tau_{k+1},s} - d_{\tau_k,s}) \geq (c_{\tau_{k+1},\varphi,s} + c_{\tau_k,\varphi,s})$  (that is, if the temporal gap between trains  $\tau_k$  and  $\tau_{k+1}$  at station  $s$  is larger than the necessary safety distances between  $\tau_k$  and  $\varphi$ , as well as between  $\varphi$  and  $\tau_{k+1}$  at  $s$ , i.e., if  $\varphi$  can depart from  $s$  between  $\tau_k$  and  $\tau_{k+1}$ ), if (b)  $((a_{\tau_{k+1},s+1} - c_{\tau_{k+1},\varphi,s} - t_{s,s+1}) \geq (a_{\tau_k,s+1} + c_{\tau_k,\varphi,s} - t_{s,s+1}))$  (that is, if the temporal distance between the arrival of train  $\tau_k$  at station  $s + 1$  plus the safety distance to  $\varphi$  and the arrival time of train  $\tau_{k+1}$  at station  $s + 1$  minus the safety distance to  $\varphi$  is positive, i.e., if  $\varphi$  can arrive at  $s + 1$  between  $\tau_k$  and  $\tau_{k+1}$ ), and if (c) the intervals defined by (a)  $[\alpha_1, \beta_1] = [d_{\tau_k,s} + c_{\tau_k,\varphi,s}, d_{\tau_{k+1},s} - c_{\tau_{k+1},\varphi,s}]$  and (b)  $[\alpha_2, \beta_2] = [a_{\tau_k,s+1} + c_{\tau_k,\varphi,s} - t_{s,s+1}, a_{\tau_{k+1},s+1} - c_{\tau_{k+1},\varphi,s} - t_{s,s+1}]$  overlap with each other and with  $w_s = [w_s^a, w_s^d]$  (that is, if the possible departure times for  $\varphi$  between trains

$\tau_k$  and  $\tau_{k+1}$  at  $s$  and the departure times from  $s$  that lead to possible arrival times between trains  $\tau_k$  and  $\tau_{k+1}$  at  $s + 1$  overlap with the given feasible time window for earliest arrival and latest departure for  $\varphi$  at  $s$ ). That is, in case the time gap between the two trains is large enough to accommodate a departure for train  $\varphi$  and these departure times fall in  $w_s = [w_s^a, w_s^d]$ . Inequality (a) determines a feasible interval for departures limited by the center gray and the leftmost pink point in Fig. 1 ( $[\alpha_1, \beta_1]$ ), inequality (b) determines a feasible interval for departures limited by the rightmost gray and the center pink point in Fig. 1 ( $[\alpha_2, \beta_2]$ ), and the feasible time window for earliest arrival and latest departure for  $\varphi$  at  $s$  is given by the leftmost gray and the rightmost pink point in Fig. 1 ( $w_s = [w_s^a, w_s^d]$ ). Hence, feasible departures for all three criteria can take place earliest at the rightmost gray and latest at the leftmost pink point, that is, within  $[\alpha_2, \beta_1]$ . Of course, the  $\alpha_i$  and  $\beta_i$  depend on  $s$ , the safety distances and the trains  $\tau_k$  and  $\tau_{k+1}$ , we only introduce them here to enhance readability of Fig. 1. In general, the earliest departure time for  $\varphi$  for this gap,  $D_{s, \tau_k, \tau_{k+1}}^e (= \max\{\alpha_1, \alpha_2, w_s^a\})$ , is given by

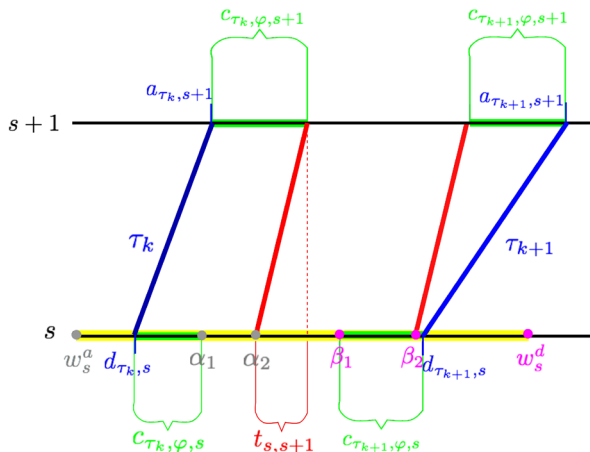
$$D_{s, \tau_k, \tau_{k+1}}^e = \max \{d_{\tau_k, s} + c_{\tau_k, \varphi, s}, a_{\tau_k, s+1} + c_{\tau_k, \varphi, s} - t_{s, s+1}, w_s^a\}, \tag{1}$$

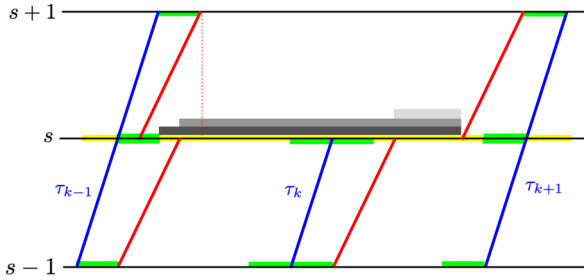
the latest departure time for  $\varphi$  for this gap,  $D_{s, \tau_k, \tau_{k+1}}^l (= \min\{\beta_1, \beta_2, w_s^d\})$ , is given by

$$D_{s, \tau_k, \tau_{k+1}}^l = \min \left\{ d_{\tau_{k+1}, s} - c_{\tau_{k+1}, \varphi, s}, a_{\tau_{k+1}, s+1} - c_{\tau_{k+1}, \varphi, s+1} - t_{s, s+1}, w_s^d \right\}. \tag{2}$$

A possible departure time for train  $\varphi$  is not only determined by  $\tau_k, \tau_{k+1}$  departing from  $s$ , but also by trains arriving at station  $s$ , see Fig. 2 for an illustration of the following description. If we simply consider the trains that run between stations  $s$  and  $s + 1$ , the interval for possible departures for the vertex we introduced is given by the dark gray interval in Fig. 2; however, departing in-between trains from station  $s - 1$  we might not actually arrive early enough at station  $s$  to allow for a departure in an arbitrary point in the dark gray interval. If we depart

**Fig. 1** Two stations,  $s$  and  $s + 1$ , are shown as horizontal black lines, time is depicted along the x-axis. Two existing, consecutive trains,  $\tau_k$  and  $\tau_{k+1}$ , are shown in blue, the safety distances (depending on the station and the trains) in green, and the feasible time window on station  $s$  in yellow. The earliest departure and the latest departure time for  $\varphi$  on  $s$  is the maximum of the three points in time marked in gray,  $\alpha_2$ , and the minimum of the three points in time marked in magenta,  $\beta_1$ , respectively





**Fig. 2** Three stations,  $s - 1$ ,  $s$  and  $s + 1$ , are shown as horizontal black lines, time is depicted along the  $x$ -axis. Existing trains are shown in blue, the safety distances in green,  $w_s = [w_s^d, w_s^a]$  in yellow, and earliest departure times for inserted trains in red. Possible departure intervals at station  $s$  as defined by the arrival times from station  $s - 1$  are shown in shades of gray

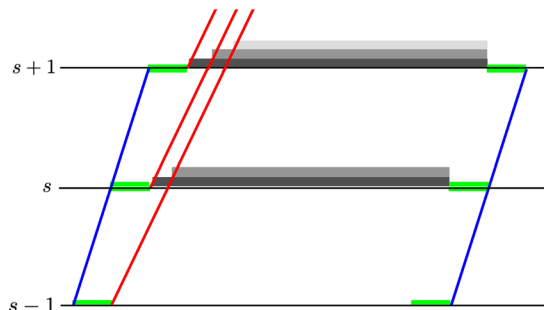
between  $\tau_{k-1}$  and  $\tau_k$  from station  $s - 1$ , only departures in the gray interval are possible, if we depart between  $\tau_k$  and  $\tau_{k+1}$  from station  $s - 1$ , only departures in the light gray interval are possible—if we depart before  $\tau_{k+1}$ , otherwise, departure intervals to the right of  $\tau_{k+1}$  are of interest. On the other hand, we need to keep the dark gray interval, as an earlier departure is possible, in case our train  $\varphi$  arrives at station  $s$  already before  $\tau_{k-1}$ , and gets passed at  $s$  before departing. Hence, in this case we need three vertices with the appropriate departure intervals.

Similarly, due to train heterogeneity, the earliest possible departure of  $\varphi$  from different stations implies different earliest departure times from consecutive stations, see Fig. 3, where three different departure intervals are defined. We add new vertices with adapted intervals for departure iteratively, from the first to the last station.

Thus, for each vertex  $v$  in  $V_{s-1}$  (let this represent a departure between trains  $\tau_k$  and  $\tau_{k+1}$ ), we define the earliest arrival at  $s$  as  $A_{v,s}^e = D_{s-1,\tau_k,\tau_{k+1}}^e + t_{s-1,s}$  and a new earliest departure time from station  $s$  as  $D_{s,\tau_k,\tau_{k+1}}^{e'} = \max\{A_{v,s}^e, d_{\tau_k,s} + c_{\tau_k,\varphi,s} + c_{\tau_k,\varphi,s} + c_{\tau_k,\varphi,s} - t_{s,s+1}, w_s^d\}$ . We add vertices and edges, the robustness of an edge is always determined by the earliest and latest departure from the vertex at its originating station (we distinguish whether a sidetrack is available for overtaking, because only with an available sidetrack waiting at a station while another train overtakes  $\varphi$  is possible):

$$E = \emptyset$$

**Fig. 3** Three stations,  $s - 1$ ,  $s$  and  $s + 1$ , are shown as horizontal black lines, time is depicted along the  $x$ -axis. Existing trains are shown in blue, the safety distances in green, earliest departure times for inserted trains in red. Earliest departure times from different stations imply different earliest departure times from consecutive stations





**IF**  $D_{s, \tau_k, \tau_{k+1}}^{e'} \leq D_{s, \tau_k, \tau_{k+1}}^{e'}$   
**IF** there exists a vertex  $w \in V_s$  with departure interval  $[D_{s, \tau_k, \tau_{k+1}}^{e'}, D_{s, \tau_k, \tau_{k+1}}^{e'}]$   
 Add an edge  $e$  from  $v$  to  $w$  to  $E$ .  
 Set the robustness of  $e$ :  $r_e = D_{s-1, \tau_k, \tau_{k+1}}^{e'} - D_{s-1, \tau_k, \tau_{k+1}}^{e'}$ .  
**ELSE**  
 Create a new vertex  $w_n \in V_s$ , add an edge  $e$  from  $v$  to  $w_n$  to  $E$ .  
 Set the robustness of  $e$ :  $r_e = D_{s-1, \tau_k, \tau_{k+1}}^{e'} - D_{s-1, \tau_k, \tau_{k+1}}^{e'}$ .  
 The interval departure times for  $w_n$  are  $[D_{s, \tau_k, \tau_{k+1}}^{e'}, D_{s, \tau_k, \tau_{k+1}}^{e'}]$ .  
**IF** There is a sidetrack available at  $s - 1$  in  $[D_{s-1, \tau_k, \tau_{k+1}}^{e'}, D_{s-1, \tau_k, \tau_{k+1}}^{e'}]$   
 Let  $v_{\text{next}}$  be the successor vertex from  $v$  on  $s - 1$ .  
 Add an edge  $e$  from  $v$  to  $v_{\text{next}}$  to  $E$ .  
 Set the robustness of  $e$ :  $r_e = D_{s-1, \tau_k, \tau_{k+1}}^{e'} - D_{s-1, \tau_k, \tau_{k+1}}^{e'}$ .

The first if-condition checks whether the earliest departure time for  $\varphi$  is smaller or equal than the latest departure time for  $\varphi$ . If this condition is fulfilled, which allows  $\varphi$  to depart between trains  $\tau_k$  and  $\tau_{k+1}$ , we need to have a vertex that represents the arrival of this feasible departure. If a vertex for the departure interval already exists, we add an edge to that vertex, and set the robustness of that edge to the difference between latest departure time and earliest departure time for  $\varphi$  between trains  $\tau_k$  and  $\tau_{k+1}$ . If no such vertex exists, we create a vertex  $w_n$  in the vertex set of station  $s$ ,  $V_s$ , add an edge to it and set the robustness accordingly. If a departure of  $\varphi$  between  $\tau_k$  and  $\tau_{k+1}$  from  $s - 1$  to  $s$  is not possible, we check whether a side track is available at  $s - 1$  and if it is, which indicates that  $\varphi$  can use this sidetrack and be overtaken by another train/other trains, we add an edge to the next vertex on  $s - 1$ . Note that the robustness depends on  $t_{s, s+1}$ , and as we define the train-specific running times by the maximum speed of the train (see Sect. 3), the robustness uses these fixed travel times between stations.

The final step already introduced some intra-station edges, that is, edges for which both endpoints are on the same station (both in  $V_{s-1}$ ). We introduce further intra-station edges to  $E$ , these always represent waiting at a station for  $\varphi$ , which is important to allow overtaking. The robustness of these edges is set to infinity ( $r_e = \infty$ ), as the robustness of any train path is not limited by waiting at a station. As intra-station edges represent waiting for overtaking, we may not add intra-station edges for stations without sidings or we may only add them if a siding is available at a specific station and time. We can now define our graph as  $G = (V, E)$  with  $V = \cup_{s=0}^M V_s$ .

Finally, we apply a postprocessing step to  $G$  and iteratively delete all vertices with either indegree zero (a vertex that cannot be reached) or outdegree zero (a vertex that cannot be left), as these cannot be part of any path from  $s_0$  to  $s_M$ .

### 4.2 Algorithm for bottleneck train path

Given the graph defined in Sect. 4.1, we want to find a path from the first vertex on station  $s_0$  to the last vertex on station  $s_M$ . Any such path would be feasible, but

we do not only aim for a feasible path, but for an optimal path. In this paper, we define the optimum as the maximum robustness, that is, we want to solve FTI with  $\mathcal{F} = \text{robustness}$ . Thus, we want to find a feasible path that maximizes the temporal distance to neighboring trains in the timetable. This translates to finding a maximum bottleneck path: only the smallest time interval to the neighboring trains on the complete path defines the robustness. This problem is also known as the *maximum capacity route problem*, see Pollack (1960), or the *widest path problem*, and can be solved by a variant of Dijkstra's shortest path algorithm (Dijkstra 1959); the pseudocode is given in Algorithm 1.

In line 4, we initialize the values: at the starting vertex of the first station the robustness is not limited (it will be by traversing the line section to the second station between two given trains), its  $r$ -value is set to infinity, for all other vertices, we want to find the maximum value, and without any initial knowledge, we set their  $r$ -values to zero (and will successively increase this value until its maximum).  $R$  is the set of already considered vertices, that is, those vertices for which we considered already all outgoing edges. In the beginning we have not considered any edges, hence,  $R$  is initialized as the empty set. In line 5, we pick the so far not considered vertex ( $v \in V(G) \setminus R$ ) with maximum  $r$ -value. This uses subpath-optimality: a path via a vertex not in  $R$  can never increase the  $r$ -value of the picked vertex. After this step  $v$  is added to  $R$  (line 6), as we consider all outgoing edges from  $v$  to a vertex  $w$  in  $V(G) \setminus R$  in lines 7–10: for each such edge  $(v, w)$  we check whether the  $r$ -value of the old path to  $w$  ( $r(w)$ ) is smaller than that of the path via  $v$  (which is the minimum of the  $r$ -value of the path to  $v$ ,  $r(v)$ , and the weight of the edge from  $v$  to  $w$ ,  $c((v, w))$ ). If this check is true, we need to update the  $r$ -value of  $w$  (using the path via  $v$  results in a path with higher bottleneck robustness to  $w$ ) in line 9 and set the predecessor of  $w$  to  $v$  in line 10. If we have not yet considered all vertices ( $R \neq V(G)$ ), we go back to line 5, otherwise, all  $r$ -values are set to the optimal value.

---

### Algorithm 1: Maximum Bottleneck Path

---

**INPUT** : Directed graph  $G$ , edge weights  $c : E(G) \rightarrow \mathbb{R}_+$ , start vertex  $s \in V(G)$ .

**OUTPUT**: Maximum bottleneck path from  $s$  to all vertices  $v \in V(G)$  and their value.

- 1  $\forall v \in V(G)$ :
- 2  $r(v)$  - the value of a maximum bottleneck  $s$ - $v$ -path.
- 3  $p(v)$  - the predecessor of  $v$  on a maximum bottleneck  $s$ - $v$ -path.
- 4 1. Set  $r(s) := \infty, r(v) := 0 \ \forall v \in V(G) \setminus \{s\}, R := \emptyset$ .
- 5 2. Find a vertex  $v \in V(G) \setminus R$  with  $r(v) = \max_{w \in V(G) \setminus R} r(w)$ .
- 6 3. Set  $R := R \cup \{v\}$ .
- 7 4. **for**  $\forall w \in V(G) \setminus R$  with  $(v, w) \in E(G)$  **do**
- 8     **if**  $(r(w) < \min\{r(v), c((v, w))\})$  **then**
- 9         Set  $r(w) = \min\{r(v), c((v, w))\}$ .
- 10          $p(w) = v$ .
- 11 5. **if**  $R \neq V(G)$  **then**
- 12     **THEN GOTO** 2

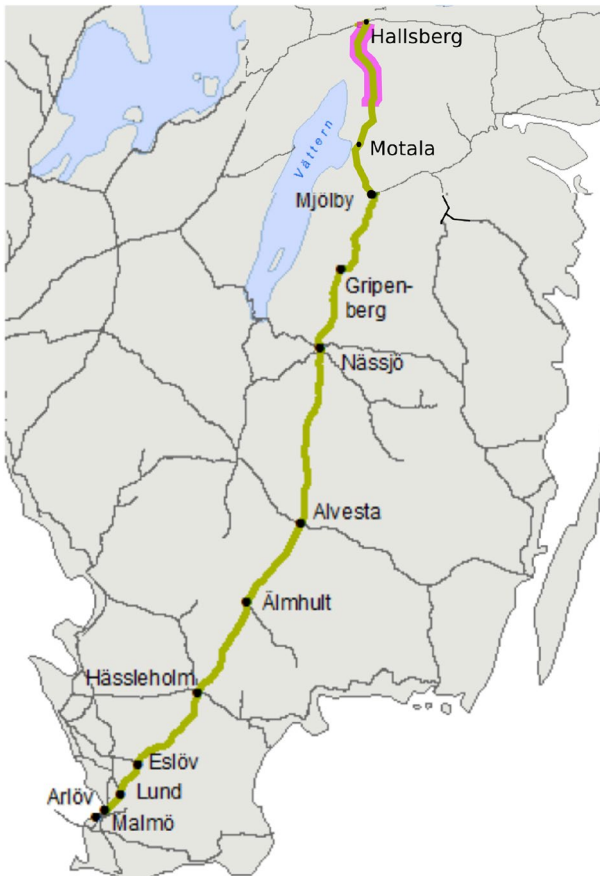
---

## 5 Experimental study: Malmö–Hallsberg

We test our method on the Swedish railway stretch between Malmö and Hallsberg, see Fig. 4. It has a length of 447 km and covers 76 stations. Between the first and the 65th station (400 km), it is double-track, while the last 11 stations are covered by single-track railway.

The studied railway stretch is part of the Scandinavian–Mediterranean freight rail corridor, and connects continental Europe to Hallsberg, the largest marshalling yard in Scandinavia; substretches are, for example, considered in Khoshniyat and Peterson (2017) and Solinen et al. (2017), and references therein.

Both the number of trains and the heterogeneity of these trains on the stretch contribute to congestion problems. According to Trafikverket, the Swedish Transport Administration, the capacity is used to more than 80% between Malmö and Hässleholm, 61–80% between Hässleholm and Tranås (located 2km North of Gripenberg),



**Fig. 4** Map of the Swedish railway stretch, with a selection of all stations marked, between Malmö and Hallsberg, single-track is marked in pink. (Source of figures: trafikverket.se)

and below 61% between Tranås and Mjölby. Between Mjölby and Hallsberg, the double-track stretch's capacity is used below 61%, the single track stretch's capacity is used to more than 80% (both when considering a 24-h and the most congested 2-h period during the day). Here, Trafikverket uses the capacity model suggested by UIC (2004).

Some of the stations along the complete stretch have additional sidetracks that enable overtaking, but not all of these are suitable to use in our case. For our algorithm, a station is considered to have a sidetrack, if:

1. The complete track is electrified.
2. It is possible to both enter and leave the sidetrack without changing direction.
3. The siding is to the left of the main tracks in case the adjacent sections are double-track (a sidetrack to the right would require the train to cross the track of trains running in the opposite direction, which is difficult in a congested environment and we do not consider it here).

A station that does not fulfill these criteria does not allow  $\varphi$  to overtake other trains (with sufficiently long scheduled stops) or to be overtaken by other trains, we mark it as a **no-wait**-station. Moreover, we distinguish the stations by the number of sidetracks. We make a simplified classification: stations with exactly one track matching the criteria, where a train can wait if no other train occupies the track, are called **single sidetrack** stations; and stations with more than one track matching the criteria, are called **multiple sidetrack** stations. Table 1 shows how many of the 76 stations fall into which class. We assume multiple sidetrack stations to have enough capacity, but check the availability for single sidetrack stations.

The remainder of this section is organized as follows: we present our base scenario in Sect. 5.1, experiments with varied time windows in Sect. 5.2, experiments with different train types in Sect. 5.3, and experiments with different numbers of total train departures in Sect. 5.4.

## 5.1 Base scenario

For our first set of experiments, we set the following parameters:

- Historical train data used: Tuesday, February 24, 2015 (a representative weekday)

**Table 1** Number of no-wait, single and multiple sidetrack stations

	Double-track station	Single-track station
No-wait station	49	4
Single sidetrack	8	4
Multiple sidetrack	9	2

**Table 2** Station number, station code, and number of train departures for all stations on the stretch from Malmö to Hallsberg on February 24, 2015

S	C	D	S	C	D	S	C	D
1	MGB	145	27	TUN	47	53	RAS	28
2	AL	140	28	KR	47	54	FRD	28
3	BLV	140	29	ÄH	46	55	GP	28
4	ÅK	139	30	DIS	46	56	TNS1	29
5	ÅKN	139	31	DIÖ	46	57	TNS	29
6	HJP	139	32	ERA	46	58	SMN	32
7	FLP	138	33	VS	46	59	BX	32
8	LU	82	34	BLD	46	60	LKN	32
9	THL	83	35	AV	34	61	MY	33
10	STB	83	36	GÅP	34	62	SKN	34
11	Ö	82	37	MO	32	63	FGL	34
12	DAT	82	38	LNS	32	64	MOT	11
13	E	82	39	GRD	32	65	ÖNA	11
14	SG	82	40	LH	32	66	D	24
15	HÖ	70	41	RK	31	67	GO	23
16	TÖ	69	42	SY	31	68	JHO	23
17	VÄD	69	43	AHM	31	69	MDM_ L3	23
18	SÖLA	69	44	SÄ	31	70	MDM	23
19	MLB	68	45	UTP	30	71	RH_L3	23
20	HM	47	46	BDF	30	72	RH	23
21	HM2	47	47	GT	29	73	Å_L	25
22	BL	47	48	N	35	74	Å	25
23	MUD	47	49	GMP	30	75	SKMS	25
24	HV	47	50	VIM	30	76	HRBG	-
25	O	47	51	FLS	29			
26	O1	47	52	ANY	28			

S station#, C station code, D # departures

- Train type: GR421410, a freight train with a maximum weight of 1400 tons, a maximum speed of 100km/h, and multiple locomotives of type Rc4
- Earliest allowed departure time from Malmö freight terminal: 25,200 s, 07:00 AM
- Latest allowed departure time from Malmö freight terminal: 47,763 s, 01:16 PM
- Latest allowed arrival time to Hallsberg marshalling yard: 64,800 s, 06:00 PM
- Critical distance:  $c_{\tau,\varphi,s} = 180 \text{ s } \forall \tau \in \mathcal{T}, \forall s \in \mathcal{S}_\varphi$

The uninterrupted travel time for our chosen train type from Malmö to Hallsberg is 4 h 44 min. On February 24, 2015, 394 unique trains ran along at least one section of the route, altogether they constituted 3830 train departures from all stations. (each train is counted for all stations along this stretch that it passes). Table 2 gives the

number of train departures for each of the 76 stations between Malmö and Hallsberg on February 24, 2015.

The train path we obtain in this base scenario is shown in Fig. 5: red and light blue give the earliest and latest possible running time for  $\varphi$ , respectively. The bottleneck for the robustness is located between the stations FLP and LU, the 7th and 8th station on the stretch from Malmö to Hallsberg; the bottleneck robustness is 300 s. The train  $\varphi$  departs from Malmö at 10:57:58 and arrives in Hallsberg at 17:53:24. This results in a travel time of 6 h and 55 min.

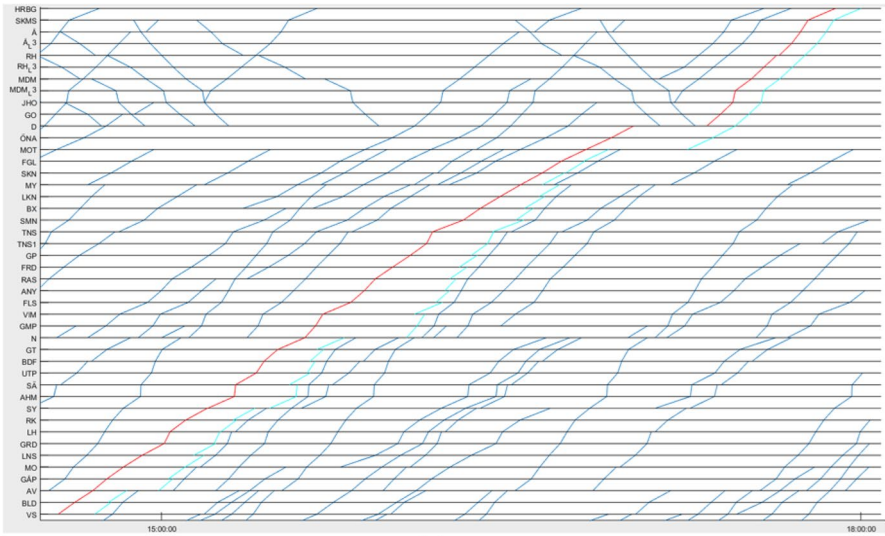
## 5.2 Variation of time windows

Intuitively, reducing the time windows for allowed departures from Malmö and allowed arrivals at Hallsberg will lead to less robust train paths: fewer train paths will be feasible, thus, a former optimal train path might no longer be available, but all other paths have smaller or equal robustness. Moreover, the temporal location of the time windows is important, that is, the same-sized time windows will lead to train paths of different robustness depending on the time of day, as the existing congestion in the train network varies. In this subsection, we investigate the relation between time window size and resulting train path robustness.

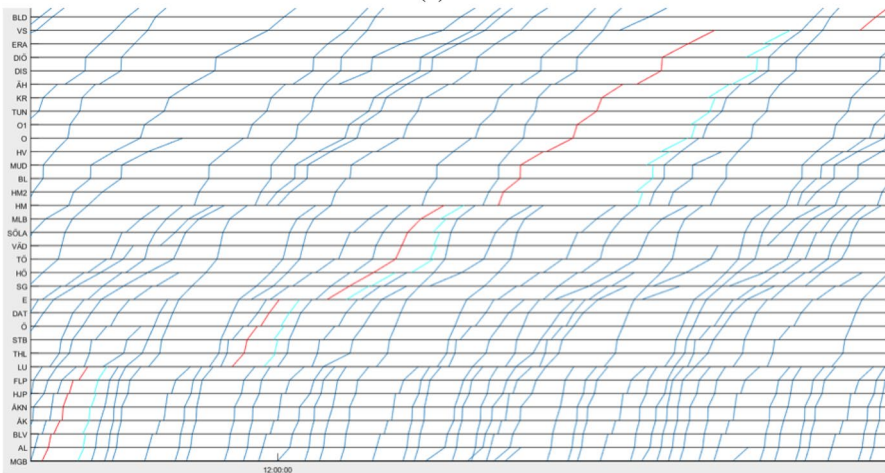
In the first set of experiments we fix the earliest departure time from Malmö (to 07:00, as in Sect. 5.1) and vary the latest possible arrival time in Hallsberg. We start with the latest possible arrival time at 24:00 and reduce it in steps of 120 s until no feasible train path can be found. The result is shown in Fig. 6: a latest arrival time in Hallsberg between 17:52 and 24:00 allows to insert a train path with robustness of 300 seconds, a latest arrival time between 16:04 and 17:50 allows to insert a train path with robustness of 147 s. The earliest feasible solution, which we obtained by running our algorithm with a resolution up to seconds, has an arrival time of 16:00:56 at Hallsberg.

If we could simply insert the train at 7 am and it could run uninterrupted, it would arrive in Hallsberg at 11:44. Thus, the earliest feasible train path arrives more than 4 h after this theoretical earliest arrival. This gap is caused by two factors: morning rush hour in the urban region around Malmö, and network congestion during daytime (due to which the train often needs to wait on sidings). During the morning rush hour between 7 and 9 there exist only five feasible time gaps of 6 min (which allow to keep the critical distance of 3 min to both the preceding and the succeeding train) between the two stations Arlöv and Burlöv just North of Malmö. In fact, when we consider stop pattern and speed of the existing trains, also these five gaps disappear: there is no feasible path that leaves Malmö before 9:00.

In the second set of experiments we fix the latest arrival time at Hallsberg (to 19:00) and vary the earliest possible departure time from Malmö. We start with the earliest possible departure time at 00:30 and increase it in steps of 120 s until no feasible train path can be found. The result is shown in Fig. 7: the robustness associated with a possible departure time varies significantly more than with the latest arrival time in Hallsberg, for a departure at 00:30 a train path with robustness of 1400 s (ca.



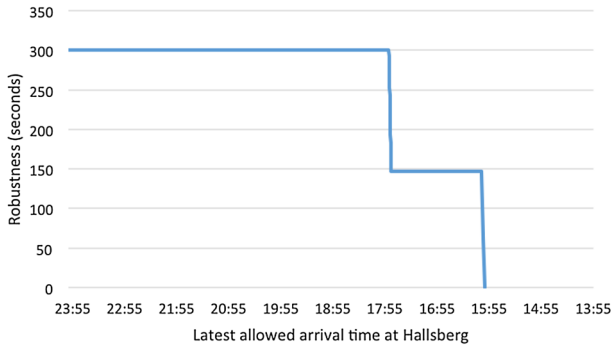
(a)



(b)

**Fig. 5** Space/time representation of the railway timetable: existing trains are shown in blue, red and light blue denote the interval borders for  $\varphi$ . To obtain maximum robustness  $\varphi$  runs in the center of the so defined corridor. **b** depicts the stretch from Malmö (MGB) to BLD, **a** depicts the stretch from VS (followed by BLD) to Hallsberg (HRBG)

23 min) can be obtained. For the first set of departure times,  $\varphi$  may run during the night, mostly undisturbed by other trains. The robustness of the best found path is reduced dramatically for a start time between 03:00 and 04:00: from 1149 s at 03:20 to 300 s at 04:00. This is, again, caused by congestion on the stretch between Malmö and Lund: between 01:00 and 04:00 one to three other trains travel on the stretch, in the hour from 04:00 to 05:00 the traffic increases to 11 trains per hour.



**Fig. 6** Robustness of the computed train path depending on the latest possible arrival time in Hallsberg

If we consider the variation of the arrival time in Hallsberg, all runs with a latest arrival time between 17:50 and 24:00 yield feasible solutions with identical robustness; however, they do not share the same path. The bottleneck is the same for all solutions, but the remainder of the paths differs. Several paths with different arrival times at Hallsberg between 18:00 and 20:00 all use the same bottleneck section. As our algorithm only accounts for the bottleneck, all these paths (with a feasible arrival time) are equally good solutions. By reducing the latest possible arrival time, some of the paths become infeasible, and the algorithm outputs a different path. We consider the problem of choosing the “best” path out of several feasible paths with the same bottleneck in Sect. 7.

Another observation, from both Figs. 6 and 7, is that there are large plateaus in the step function. That is, increasing the time window size does not necessarily—and will in fact often not—result in a train path with increased robustness.

### 5.3 Variation of train type

In our base scenario in Sect. 5.1 we used a train of type GR421410, a freight train with a maximum weight of 1400 tons, a maximum speed of 100 km/h, and multiple locomotives of type Rc4. Using different train types, with different maximum speed and maximum weight restrictions, results in different runtimes for the train from station to station. Both a slower and a faster train might lead to train paths with better robustness, depending on the speed pattern of the already existing trains, see Fig. 8. Recall that in our model all trains always run with the maximum allowed speed. Using a faster train might lead to an earlier arrival at a station, which will in turn open up train path departure possibilities from that station that were not an option for a later arriving, slower train. Moreover, if the freight train is not restricted by other existing trains, a faster train will automatically obtain a shorter travel time and an earlier arrival time at the final station.

To investigate the sensitivity of a train path’s robustness, we analyze nine train types, see Table 3 for an overview of their attributes. All of these types operate currently on the considered stretch, and we consider them a good representation of all the train types operating between Malmö and Hallsberg.



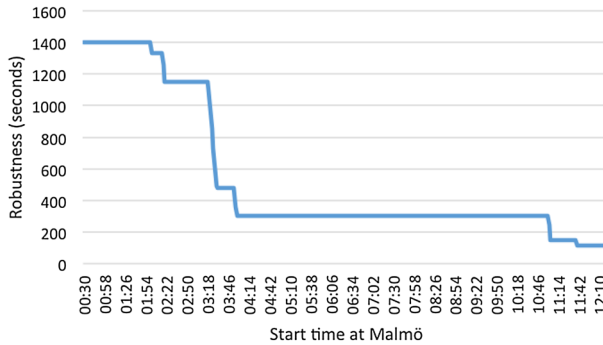


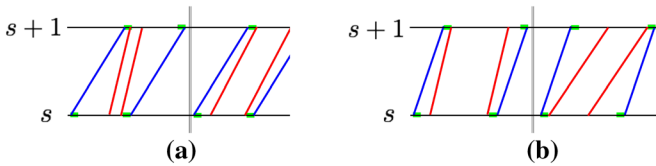
Fig. 7 Robustness of the computed train path depending on the earliest possible departure time in Malmö

For each train type, we allowed a latest arrival time at 18:00, and then reduced this time stepwise by intervals of 45 min. The resulting robustness of train paths found by our algorithm is given in Table 4.

The first five trains obtain paths with quite similar robustness, if we compare the travel time to our train from the base scenario (GR421410), trains GB201310 and GB402010 have a 139 s longer runtime between Malmö and Hallsberg, train GT421209 has a 2300 s longer runtime, and train GR401608 has a 4360 s longer runtime. On the other hand, trains GR400710 and GR401409 take 22 s less than GR421410, for the two passenger train types the difference is even larger: PX2-2000 and PX610016 need 6753 and 5546 s, respectively, less than the train from the base scenario.

The faster passenger trains obtain feasible train paths also for a latest arrival time at 15:00 and 15:45; on the other hand, the robustness of the obtained passenger train paths is not consistently better than that of freight trains: for arrival at 18:00 the best train path for passenger trains is 272 and 277, while freight trains yield a train path with 300 s robustness.

We take a more detailed look into this difference in robustness: in Fig. 9 we compare the earliest and latest possible train paths for the PX2-2000 and the GR421410 at the bottleneck, which for both occurs at the same station, FLP, between the same preceding and succeeding train. The main difference in robustness is not caused by the latest possible departure time (the light blue lines nearly coincide at station FLP), but by the earliest possible departure time. The earliest possible departure time is defined by the preceding train. This train is slower than both considered trains, thus, the slower of the two trains, GR421410, may depart earlier to keep the safety distance to the preceding train than the faster passenger train. Thus, the resulting robustness at the bottleneck is higher for the freight train. Hence, we can conclude that the algorithm yields higher robustness for trains with a speed similar to that of adjacent traffic.



**Fig. 8** Existing trains between station  $s$  and station  $s + 1$  are shown in blue, the earliest and latest possible departure for an inserted train is shown in red (limited by the safety distance to the existing trains shown in green). The distance between these two red lines defines the robustness of this section of the train path. **a** A slower train (right) obtains a higher robustness than a faster train (left). **b** A faster train (left) obtains a higher robustness than a slower train (right)

**Table 3** Train types, maximum speed, and maximum weight

train type ID	Train type	Speed (km/h)	Cargo weight (t)
GB201310	Freight	100	1300
GB402010	Freight	100	2000
GR400710	Freight	100	700
GR401409	Freight	90	1400
GR421410	Freight	100	1400
GT421209	Freight	90	1200
GR401608	Freight	80	1600
PX2-2000	Passenger	200	-
PX610016	Passenger	160	-

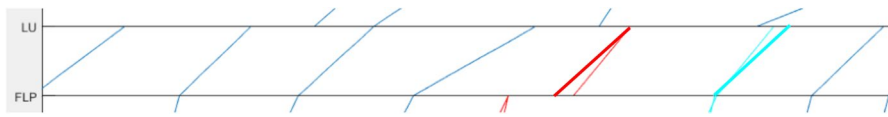
#### 5.4 Variation of number of train departures in the network

So far we used the time table of Tuesday, February 24, 2015 for our experiments. In this subsection, we analyze the impact of different days with different numbers of trains and time tables on the robustness of the train path of our added freight train. We analyze all days in the 2015 annual time table (December 15, 2014–December 09, 2015) with the standard time window of 06:00–18:00. Figure 10 shows the maximum bottleneck robustness for our inserted freight train for all days of the 2015 annual time table [some dates (38 days) were excluded due to incomplete input data]. We can observe only a few distinct bottleneck robustness values, but these values repeat frequently over the considered time period. In particular, the maximum bottleneck robustness follows an alternating high-low value trend for weekdays/weekends.

When we compare the number of existing trains in the time table on a specific day and the robustness obtained for the inserted freight train, we can observe a clear correlation, see Fig. 11: on days with a higher number of existing trains (usually weekdays) the maximum bottleneck robust path is less robust than on days with fewer existing trains (usually weekends). In Fig. 11 we can observe some exceptions to this rule in the two first weeks, thus, the general statement that more existing trains in the time table will always lead to a lower bottleneck value is not correct. To study the relation more closely, we performed a linear regression analysis; for the

**Table 4** Train type and achieved robustness for different latest arrival times at Hallsberg

	15:00	15:45	16:30	17:15	18:00
GB201310	–	–	146	146	300
GB402010	–	–	146	146	300
GR400710	–	–	147	147	300
GR401409	–	–	147	147	300
GR421410	–	–	147	147	300
GT421209	–	–	97	97	228
GR401608	–	–	–	15	26
PX2-2000	107	107	230	230	272
PX610016	107	107	238	238	277

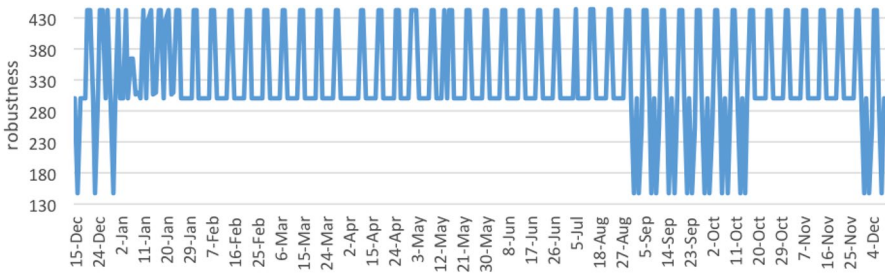


**Fig. 9** Earliest and latest possible train path of the passenger train PX2-2000 and the freight train GR421410 (bold lines) for the bottleneck shown in red and light blue, respectively. All existing trains in the time table are shown in dark blue

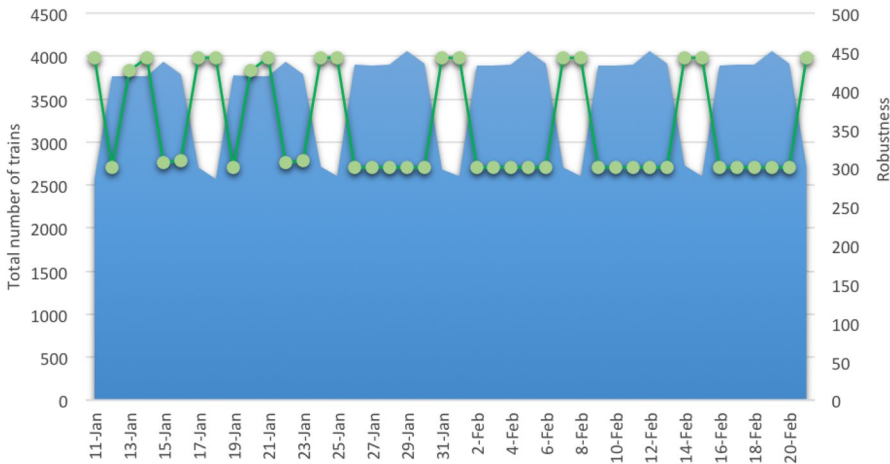
plot of both, all single data points and the resulting regression line see Fig. 12. We obtained a linear function with negative slope, which underlines the aforementioned trend of more existing trains in the time table leading to a lower bottleneck robustness; and an  $R^2$  value of 0.7091. To study the existing variance, we focussed on two outlier data points, shown with a blue frame in Fig. 12 (the two data points overlap): January 14 and January 21, 2015, both days are Wednesdays. We compared these 2 days with the Wednesday of the succeeding week, January 28; see Fig. 11 for the number of existing trains and the resulting robustness of the inserted freight train on these three dates. When comparing the train path our algorithm obtains for the inserted freight trains on January 14, 21 and 28, we observed that the output train path of January 14 and 21 is blocked by another train on January 28. Thus, the algorithm outputs another (in this scenario best) train path, with the bottleneck located at station FLP (Flackarp), see Fig. 13. That is, the existence of a single additional train reduces the robustness of the inserted train path significantly. This explains the relatively low  $R^2$  value we obtained. The general trend of more existing trains often yielding a train path with lower robustness holds.

Finally, we consider all weekdays and all Tuesdays, to justify our choice of February 24, 2015 as a representative day of the 2015 annual time table. Figure 14a shows the number of existing trains per day and the robustness of the train path obtained by our algorithm for all weekdays (with all public holidays omitted), Fig. 14b shows the same information for all Tuesdays only of the same time period.

As described, the robustness for the inserted train path varies significantly—and the existence of one additional train may have a large impact on the



**Fig. 10** Maximum bottleneck robustness of the freight train obtained for all days for the 2015 annual time table, except for April 3, 2015–April 6, 2015 and July 6, 2015–August 9, 2015

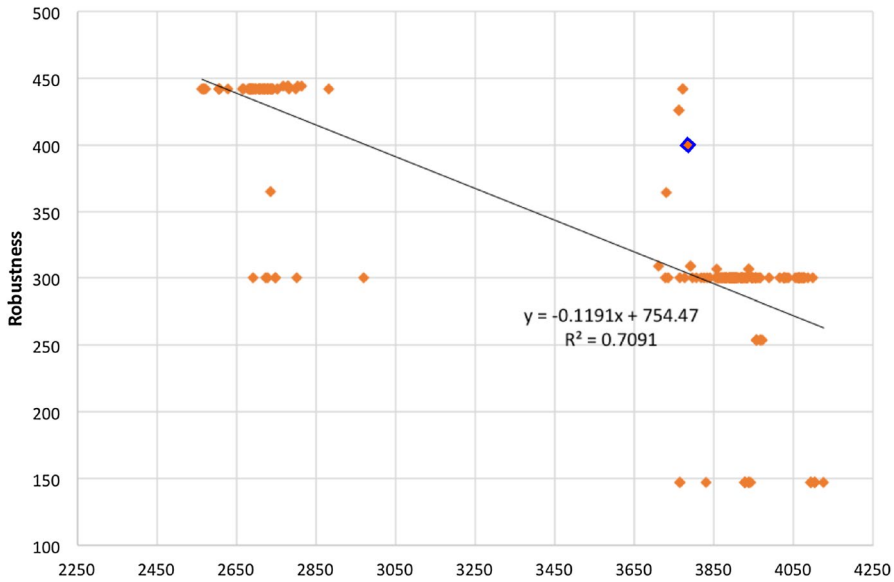


**Fig. 11** Number of trains per day and robustness of the inserted freight train path shown in blue and green, respectively, selected time interval January 11–February 21, 2015

robustness. We observe February 24 to be representative for the robustness range on the majority of weekdays in the 2015 annual time table.

### 6 Analysis: runtime and graph size

Obviously, our algorithm is a variant of Dijkstra’s algorithm and, thus, runs in  $\mathcal{O}(|V| \log |V| + |E|)$  on a given graph  $G = (V, E)$ . To analyze the runtime of our algorithm we have to determine the number of vertices and edges in our constructed graph. These values depend on the number of stations,  $|\mathcal{S}_\varphi|$ , and on the number of existing trains in the time table,  $|\mathcal{T}|$ . We yield:



**Fig. 12** Robustness over number of trains in the time table. Actual data points, given by days from the 2015 annual time table, are shown in orange, the linear function obtained by regression analysis is shown in black

$$|V| \leq |\mathcal{S}_\varphi| + |\mathcal{T}| \cdot \sum_{k=1}^{|\mathcal{S}_\varphi|} k = |\mathcal{S}_\varphi| + \frac{|\mathcal{T}| \cdot |\mathcal{S}_\varphi| \cdot (|\mathcal{S}_\varphi| + 1)}{2} \tag{3}$$

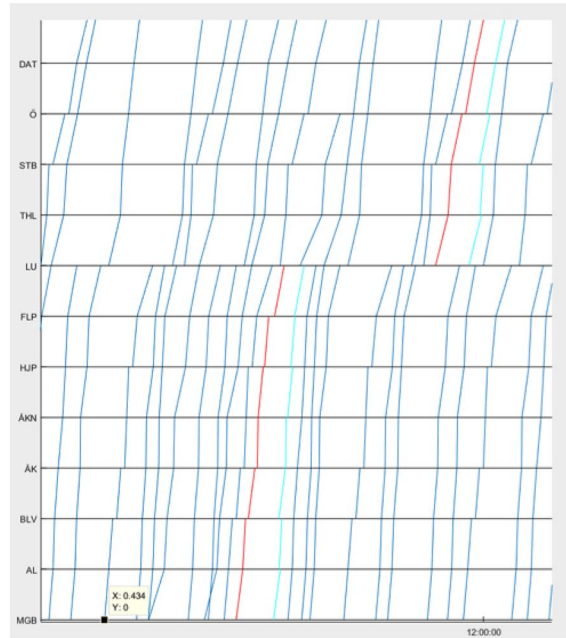
Because each vertex at station  $s$  has at most one directed edge to a vertex on  $s + 1$  and one directed edge to a succeeding vertex on  $s$ , we have  $|E| \leq 2|V|$ . Consequently

$$(|V| \log |V| + |E|) \leq (|V| \cdot (2 + \log |V|)) \tag{4}$$

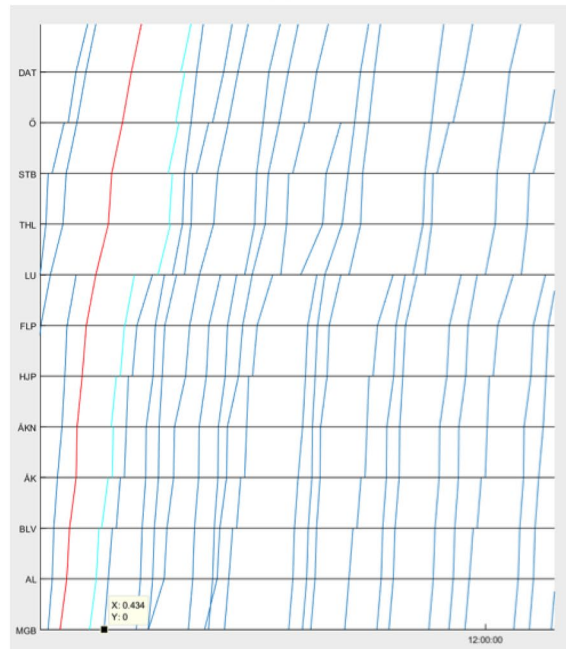
which yields a runtime of  $\mathcal{O}(|\mathcal{S}_\varphi|^2 |\mathcal{T}| \log(|\mathcal{S}_\varphi|^2 |\mathcal{T}|))$ .

While this reflects the worst case upper bound, this bound is not tight and the postprocessing step described in Sect. 4.1 significantly reduces the number of vertices in our experiments. For our base scenario with  $|\mathcal{S}_\varphi| = 76$  and  $|\mathcal{T}| = 251$  Eq. (3) yields an upper bound of 734502 on the number of vertices. However, we actually only have  $|V| = 6948$ , that is, the inequality really includes a lot of slack or hides very small constants, and after the postprocessing step, we are left with only  $|V| = 3259$  (less than half of the original vertices), which is significantly smaller than the given upper bound. This reduction is highlighted in Fig. 15: the number of vertices actually decreases with the number of trains, this holds for different train types as shown in Fig. 15b.

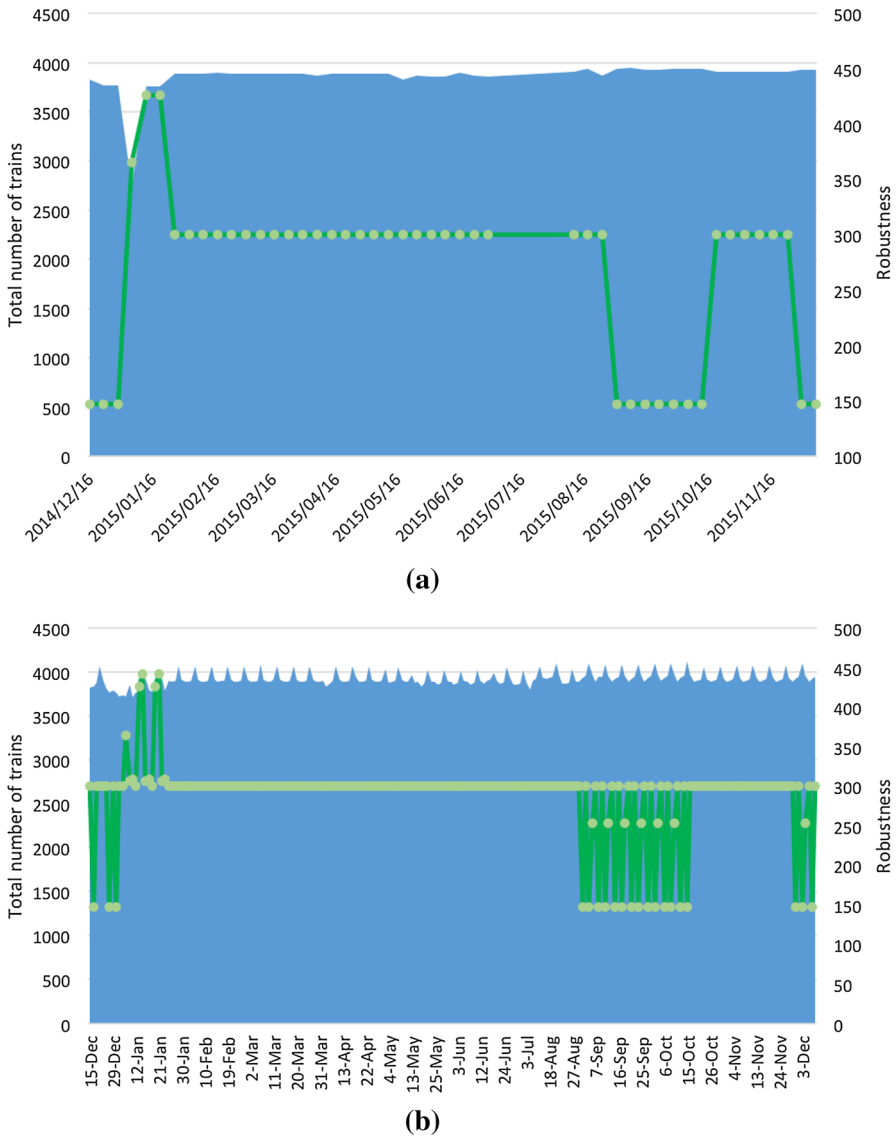
**Fig. 13** Analysis of the bottleneck of the inserted train path on January 28. The robustness on that day is 300 s, and the bottleneck occurs at station FLP (the last station before Lund). **a, b** Show the earliest and latest possible train paths for the inserted freight train (in red and light blue) on January 28 and January 21, respectively. The path we obtain for January 21 is blocked by another, existing train on January 28, thus, we cannot insert the train in the same gap



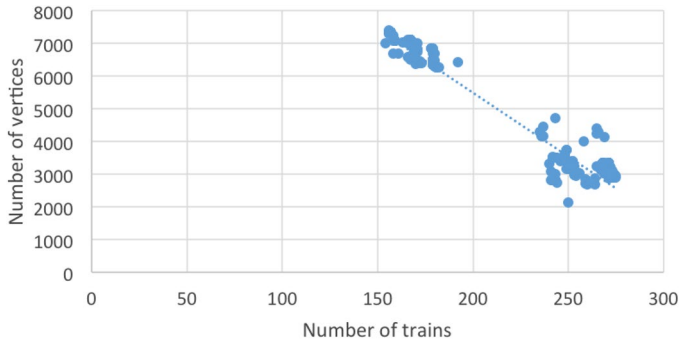
(a)



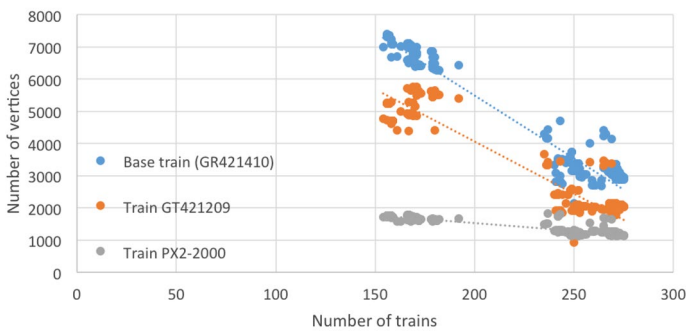
(b)



**Fig. 14** Number of trains per day and robustness of the inserted freight train path shown in blue and green, respectively. **a** For all tuesdays and **b** for all weekdays of the 2015 annual time table



(a)



(b)

**Fig. 15** Number of vertices over number of trains: **a** data points for different days, **b** comparison for different train types

## 7 Note on an improved algorithm for the “Best” bottleneck train path

The algorithm presented in Sect. 4.2, and analyzed in detail in Sect. 5, has one drawback: We only care about the robustness at the bottleneck, the rest of the train path is not optimized, we only know that nowhere along the path the bottleneck robustness is undercut. Thus, two train paths over five stations with robustness of 150, 150, 150, 150, 150 and 350, 300, 150, 350, 400 at the stations are equally good, and our algorithm might output the former. When we actually want to insert a freight train that influences the existing trains as little as possible, we would prefer the latter. However, this might, e.g., lead to longer total travel times for the inserted train.

We can easily adapt our algorithm to reflect this choice by increasing our storage complexity: We store a vector,  $R$ , with the robustness at all stations along the path. We still update the path in case it allows for a higher bottleneck robustness, but, if we found another path with the same bottleneck robustness for which the vector  $R$  is lexicographical larger than the vector  $R$  for the old path, we update our chosen path



as well. See Algorithm 2 for the pseudocode of this improved algorithm. We leave an implementation of this algorithm to future work.

---

**Algorithm 2: Best Maximum Bottleneck Path**

---

```

INPUT : Directed graph  $G$ , edge weights  $c : E(G) \rightarrow \mathbb{R}_+$ , start vertex  $s \in V(G)$ .
OUTPUT: Maximum bottleneck path from  $s$  to all vertices  $v \in V(G)$ , which maximizes the second smallest robustness, third smallest robustness etc., and their value.

1  $\forall v \in V(G)$ :
2  $r(v)$  - the value of a maximum bottleneck  $s$ - $v$ -path.
3  $p(v)$  - the predecessor of  $v$  on a maximum bottleneck  $s$ - $v$ -path.
4  $R(v) = (r_{v,1}, \dots, r_{v,M})$  - a vector of the robustness at all stations along a maximum bottleneck  $s$ - $v$ -path (except for station 0).
5 1. Set  $r(s) := \infty, r(v) := 0 \ \forall v \in V(G) \setminus \{s\}, R := \emptyset$ .
6  $r_{v,i} = \infty \forall v \in V(G) \setminus \{s\}, \forall i \in \{1, \dots, M\}$ .
7 2. Find a vertex  $v \in V(G) \setminus R$  with  $r(v) = \max_{w \in V(G) \setminus R} r(w)$ .
8 3. Set  $R := R \cup \{v\}$ .
9 4. for  $\forall w \in V(G) \setminus R$  with  $(v, w) \in E(G)$  do
10 |   if  $(r(w) < \min\{r(v), c((v, w))\})$  then
11 |   |   Set  $r(w) = \min\{r(v), c((v, w))\}$ .
12 |   |    $p(w) = v$ .
13 |   |   Set  $R(w)$  to  $R(v)$  extended by  $c(v, w)$ .
14 |   else if  $(r(w) = \min\{r(v), c((v, w))\})$  then
15 |   |   if  $(R(v) >_{lex} R(w))$  then
16 |   |   |    $p(w) = v$ .
17 |   |   |   Set  $R(w)$  to  $R(v)$  extended by  $c(v, w)$ .
18 5. if  $R \neq V(G)$  then
19 |   THEN GOTO 2

```

---

## 8 Conclusions and outlook

We presented an algorithm that can be implemented to insert one additional train in an existing timetable. The algorithm is fast and gives a satisfying result in reasonable time for operational use.

In our test set of experiments, we solely used the maximum bottleneck as objective. This is well-motivated from a robustness perspective, but can easily be extended to account for other goodness measures; in particular, we can easily account for the robustness on the path to and from the bottleneck section by increasing our storage complexity. Moreover, preprocessing may be applied to omit train paths with unwanted properties such as a travel time exceeding the train driver’s maximum allowed working time. The study of further objectives for FTI is left as future work.

As a heuristic for inserting several trains, the algorithm may be called repeatedly. To obtain better algorithms for this scenario, we might study either the offline or the online problem in which multiple additional train paths need to be inserted within a time interval. Moreover, our approach allows for a specific, defined speed of the inserted train, and if we allow a larger number of allowed

speeds (still a discrete set), this will increase the size of our graph. Hence, it might be interesting to study other types of algorithms that allow a continuous spectrum of train speed.

**Funding** Open access funding provided by Linköping University.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Andersson EV, Peterson A, Krasemann JT (2013) Quantifying railway timetable robustness in critical points. *J Rail Transp Plan Manag* 3(3):95–110. <https://doi.org/10.1016/j.jrtpm.2013.12.002>
- Burdett R, Kozan E (2009) Techniques for inserting additional trains into existing timetables. *Transp Res Part B Methodol* 43(8):821–836
- Cacchiani V, Caprara A, Toth P (2010) Scheduling extra freight trains on railway networks. *Transp Res Part B Methodol* 44(2):215–231
- Cacchiani V, Huisman D, Kidd M, Kroon L, Toth P, Veelenturf L, Wagenaar J (2014) An overview of recovery models and algorithms for real-time railway rescheduling. *Transp Res Part B Methodol* 63:15–37
- Commission of the European communities (2007) Towards a rail network giving priority to freight. Communication from the commission to the council and the European parliament. <https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:52007DC0608>
- Dijkstra EW (1959) A note on two problems in connexion with graphs. *Numer Math* 1:269–271
- Eurostat (2017) Railway transport-goods transported, by type of transport. [http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=rail\\_go\\_typeall&lang=en](http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=rail_go_typeall&lang=en)
- Flier H (2011) Optimization of railway operations: algorithms, complexity, and models. PhD thesis, ETH. <https://www.research-collection.ethz.ch/handle/20.500.11850/72891>
- Flier H, Graffagnino T, Nunkesser M (2009) Scheduling additional trains on dense corridors. In: 8th International Symposium on experimental algorithms (SEA 2009), Dortmund, Germany, June 4–6, 2009, pp 149–160
- Goerigk M, Schöbel A (2014) Recovery-to-optimality: a new two-stage approach to robustness with an application to aperiodic timetabling. *Comput Oper Res* 52:1–15
- Goverde RMP, Hansen IA (2013) Performance indicators for railway timetables. In: 2013 IEEE International Conference on intelligent rail transportation proceedings, pp. 301–306
- Grimm M (2012) The analysis of congested infrastructure and capacity utilisation at trafikverket. *WIT Trans Built Environ* 127:359–367
- Hansen IA, Pachl J (2014) Railway timetable & traffic: analysis-modelling-simulation, 2nd edn. Eurailpress in DVV Media Group, Hamburg
- Harrod SS (2012) A tutorial on fundamental model structures for railway timetable optimization. *Surv Oper Res Manag Sci* 17(2):85–96
- Ingolotti L, Barber F, Tormos P, Lova A, Salido MA, Abril M (2004) An efficient method to schedule new trains on a heavily loaded railway network. Springer Berlin Heidelberg, Berlin, pp 164–173
- Jiang F, Cacchiani V, Toth P (2017) Train timetabling by skip-stop planning in highly congested lines. *Transp Res Part B Methodol* 104:149–174

- Khoshniyat F, Peterson A (2017) Improving train service reliability by applying an effective timetable robustness strategy. *J Intell Transp Syst Technol Plan Oper* 21(6):525–543. <https://doi.org/10.1080/15472450.2017.1326114>
- Kroon L, Huisman D, Maróti G (2008) Optimisation models for railway timetabling. In: Hansen I, Pacht J (eds) *Railway timetable and traffic*. Eurailpress, Hamburg, pp 135–154
- Liebchen C (2008) The first optimized railway timetable in practice. *Transp Sci* 42(4):420–435
- Ljunggren F, Persson K, Peterson A, Schmidt C (2018) Maximum robust train path for an additional train inserted in a railway timetable close to operation. In: *Conference on advanced systems in public transport and transitdata 2018*, Brisbane
- Pollack M (1960) Letter to the editor—the maximum capacity through a network. *Oper Res* 8(5):733–736
- Solinen E, Nicholson G, Peterson A (2017) A microscopic evaluation of railway timetable robustness and critical points. *J Railw Transp Plann Manag* 5:207–223
- Törnquist J (2006) Computer-based decision support for railway traffic scheduling and dispatching: a review of models and algorithms. *OASICs-OpenAccess Ser Inf*. <https://doi.org/10.4230/OASICs.ATMOS.2005.659>
- Trafikanalys (2017a) Rail traffic 2016. <https://www.trafa.se/globalassets/statistik/bantrafik/jarnvagstransporter/2017/jarnvagstransporter-2017-kvartal-3.pdf?>
- Trafikanalys (2017b) Railway transport 2017 quarter 3. [https://www.trafa.se/globalassets/statistik/bantrafik/bantrafik/2016/bantrafik-2016\\_okt.pdf?](https://www.trafa.se/globalassets/statistik/bantrafik/bantrafik/2016/bantrafik-2016_okt.pdf?)
- UIC (2004) Uic code 406: capacity. Technical report, International Union of Railways. 1st edition
- Vromans MJ, Dekker R, Kroon LG (2006) Reliability and heterogeneity of railway services. *Eur J Oper Res* 172(2):647–665
- Zhang J (2015) Analysis on line capacity usage for China high speed railway with optimization approach. *Transp Res Part A Policy Pract* 77:336–349

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.