**CASPT 2018**

# Maximum Robust Train Path for an Additional Train Inserted in an Existing Railway Timetable

Fredrik Ljunggren · Kristian Persson ·
Anders Peterson · Christiane Schmidt

**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. 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

## 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 tonne-kilometres), respectively, see [14, 18, 19]. 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 [5], the European commission sees the potential to revitalize rail freight [3]: road congestion and the high oil price make road transport

Fredrik Ljunggren
Communications and Transport Systems, ITN, Linköping University, Norrköping, Sweden.
Current affiliation: Trafikverket, Stockholm, Sweden, fredrik.ljunggren@trafikverket.se

Kristian Persson
Communications and Transport Systems, ITN, Linköping University, Norrköping, Sweden.
Current affiliation: Sweco, Stockholm, Sweden, kristian.persson@sweco.se

Anders Peterson, Christiane Schmidt
Communications and Transport Systems, ITN, Linköping University, Norrköping, Sweden,
{anders.peterson, christiane.schmidt}@liu.se

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. Already with the current traffic, railway infrastructure is often overloaded. This is particularly true for marshalling yards: trains that are already completed occupy highly demanded space until their departure. To free this capacity, the freight operator and the infrastructure manager (IM) often agree in their goal to depart early. Today, such a request is typically answered manually by looking a few stations ahead, and if the completed 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.

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.

In this paper, we propose an algorithm that computes a maximum robust train path for inserting an additional train (at a time). 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 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.

### 1.1 Roadmap

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

### 1.2 Related Work

Timetabling is a problem that has been extensively studied, in the majority a new time table, or a larger part of it, is constructed from scratch, see e.g. Hansen and Pachl [9], Liebchen [13], or Törnquist [17] for an overview.

Various authors also considered adding a new train to an existing timetable, amongst others Burdett and Kozan [1]. Flier et al. [7] (see also [6]) 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. [10] consider adding new trains to a heterogeneous, heavily loaded railway network, and aim to minimize the traversal time for each additional train.

Cacchiani et al. [2] 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).

Robustness might be defined in various ways, cp. Kroon et al. [12]. If we assume delays to be uniformly distributed, our objective function is valid, see Goerigk and Schöbel [8].

## 2 Notation and Preliminaries

**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 $\mathcal{S}_\varphi = (s_0, s_1, \ldots, s_M)$, when clear from content, we only refer to the stations by $0, \ldots, M$; 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, stations $s \in \mathcal{S}_\varphi$ ; the train-specific running times $t_{i,i+1}$ for train $\varphi$ from station $i$ to $i+1$ $\forall i \in \{0, \ldots, M-1\}$; 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 $t_{i,i+1}$ $\forall i \in \{0, \ldots, M-1\}$, and optimizes $\mathcal{F}$.
In this paper, we have $\mathcal{F}$ = robustness.

## 3 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 Section 3.1. We then show that we can use a variant of Dijkstra's algorithm to compute the maximum robust train path, see Section 3.2.

### 3.1 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 $\big(d_{\tau_{k+1},s} - d_{\tau_k,s}\big) \geq \big(c_{\tau_{k+1},\varphi,s} + c_{\tau_k,\varphi,s}\big)$, if $\big((a_{\tau_{k+1},s+1} - c_{\tau_{k+1},\varphi,s} - t_{s,s+1}) \geq (a_{\tau_k,s+1} + c_{\tau_k,\varphi,s} - t_{s,s+1})\big)$, and if this gap falls in the feasible time window at station $s$, see Figure 1(a). That is, in case the time gap between the two trains is large enough to accommodate a departure for train $\varphi$. The earliest departure time for $\varphi$ for this gap, $D^e_{s,\tau_k,\tau_{k+1}}$, is given by

$$D^e_{s,\tau_k,\tau_{k+1}} = \max\{d_{\tau_k,s} + c_{\tau_k,\varphi,s}, a_{\tau_k,s+1} + c_{\tau_k,\varphi,s} - t_{s,s+1}, w^a_s\}, \qquad (1)$$

the latest departure time for $\varphi$ for this gap, $D^\ell_{s,\tau_k,\tau_{k+1}}$, is given by

$$D^\ell_{s,\tau_k,\tau_{k+1}} = \min\{d_{\tau_{k+1},s} - c_{\tau_{k+1},\varphi,s}, a_{\tau_{k+1},s+1} - c_{\tau_{k+1},\varphi,s+1} - t_{s,s+1}, w^d_s\}. \quad (2)$$

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 Figure 1(b) 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 pink interval in Figure 1(b); however, departing in-between trains from station $s-1$ we might not actually arrive

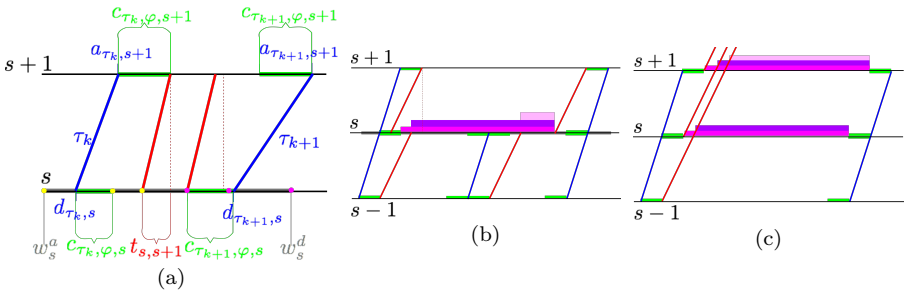



**Fig. 1** (a) Two stations, $s$ and $s+1$, are shown as horizontal black lines, time is depicted along the x-axis. Two existing, consecutive trains, $\tau_k$ and $\tau_{k+1}$, are shown in blue, the safety distances (depending on the station and the trains) in green, and the feasible time window on station $s$ in gray. The earliest departure and the latest departure time for $\varphi$ (shown in red) on $s$ is the maximum of the three points in time marked in yellow and the minimum of the three points in time marked in magenta, respectively. (b)/(c) 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. (b) Possible departure intervals at station $s$ as defined by the arrival times from station $s-1$. (c) Earliest departure times from different stations imply different earliest departure times from consecutive stations.

early enough at station $s$ to allow for a departure in any point in the pink interval. If we depart between the first and second blue train from station $s-1$, only departures in the violet interval are possible, if we depart between the second and third blue train from station $s-1$, only departures in the light pink interval are possible—if we depart before the blue train to the right. On the other hand, we need to keep the pink interval, as an earlier departure is possible, in case our train $\varphi$ arrives at station $s$ already before the left blue train, 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 Figure 1(c), 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 $A^e_{v,s} = D^e_{s-1,\tau_k,\tau_{k+1}} + t_{s-1,s}$, and a new earliest departure time from station $s$ as $D^{e'}_{s,\tau_k,\tau_{k+1}} = \max\{A^e_{v,s}, d_{\tau_k,s} + c_{\tau_k,\varphi,s}, a_{\tau_k,s+1} + c_{\tau_k,\varphi,s} - t_{s,s+1}, w^a_s\}$. 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):

$E = \emptyset$
**IF** $D^{e'}_{s,\tau_k,\tau_{k+1}} \leq D^{\ell}_{s,\tau_k,\tau_{k+1}}$
  **IF** there exists a vertex $w \in V_s$ with departure interval $[D^{e'}_{s,\tau_k,\tau_{k+1}}, D^{\ell}_{s,\tau_k,\tau_{k+1}}]$

Add an edge $e$ from $v$ to $w$ to $E$.
Set the robustness of $e$: $r_e = D^{\ell}_{s-1,\tau_k,\tau_{k+1}} - D^{e}_{s-1,\tau_k,\tau_{k+1}}$.

**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 $e$: $r_e = D^{\ell}_{s-1,\tau_k,\tau_{k+1}} - D^{e}_{s-1,\tau_k,\tau_{k+1}}$.
The interval departure times for $w_n$ are $D^{e'}_{s,\tau_k,\tau_{k+1}}, D^{\ell}_{s,\tau_k,\tau_{k+1}}$.

**ELSE**

**IF** There is a sidetrack available at $s-1$ in $[D^{e}_{s-1,\tau_k,\tau_{k+1}}, D^{\ell}_{s-1,\tau_k,\tau_{k+1}}]$
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 $e$: $r_e = D^{\ell}_{s-1,\tau_k,\tau_{k+1}} - D^{e}_{s-1,\tau_k,\tau_{k+1}}$.

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 $s_M$.

### 3.2 Algorithm for Bottleneck Train Path

Given the graph defined in Subsection 3.1, 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 $\mathcal{F}$ = 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 [15], or the *widest path problem*, and can be solved by a variant of Dijkstra's shortest path algorithm [4]; the pseudocode is given in Algorithm 1.

## 4 Experimental Study: Malmö–Hallsberg

We test our method on the Swedish railway stretch between Malmö and Hallsberg, see Figure 2. It has a length of 447 km and covers 76 stations. Between

**Algorithm 1:** Maximum Bottleneck Path

**INPUT :** Directed graph $G$, edge weights $c : E(G) \to \mathbb{R}_+$, start vertex $s \in V(G)$.
**OUTPUT:** Maximum bottleneck path from $s$ to all vertices $v \in V(G)$ and their value.

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

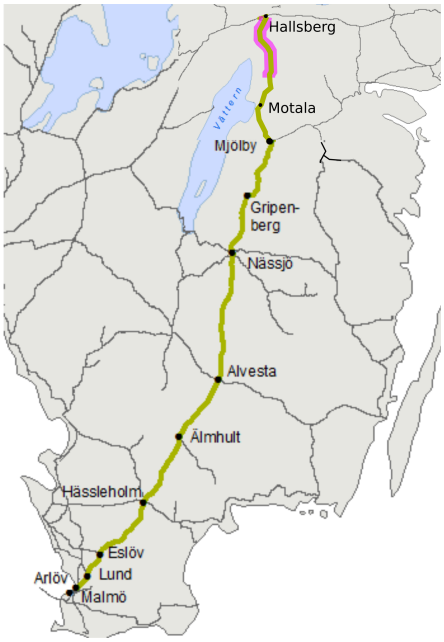



**Fig. 2** Map of the Swedish railway stretch, with a selection of all stations marked, between Malmö and Hallsberg, single-track is marked in pink. (Source of figures: trafikverket.se)

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

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

| | double-track station | single-track staion |
|---|---|---|
| no-wait station | 49 | 4 |
| single sidetrack | 8 | 4 |
| multiple sidetrack | 9 | 2 |

marshalling yard in Scandinavia; substretches are, for example, considered in Khoshniyat et al. [11] and Solinen et al. [16], 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, 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-hour and the most congested 2-hour period during the day). Here, Trafikverket uses the capacity model suggested by UIC [20].

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 stop) or to be overtaken by other trains, we mark it as a **no-wait**-station. Moreover, we distinguish 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 Subsection 4.1, experiments with varied time windows in Subsection 4.2, experiments with different train types in Subsection 4.3, and experiments with different number of total train departures in Subsection 4.4.

### 4.1 Base Scenario

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

- Historical train data used: Tuesday, February 24, 2015 (a representative weekday)
- 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: 25200 seconds, 07:00 AM
- Latest allowed departure time from Malmö freight terminal: 47763 seconds, 01:16 PM
- Latest allowed arrival time to Hallsberg marshalling yard: 64800 seconds, 06:00 PM
- Critical distance: $c_{\tau,\varphi,s} = 180$ seconds $\forall \tau \in \mathcal{T}, \forall s \in \mathcal{S}_\varphi$

The uninterrupted travel time for our chosen train type from Malmö to Hallsberg is 4 hours 44 minutes. 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 Figure 3: 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 seconds. The train $\varphi$ departs Malmö at 10:57:58 and arrives in Hallsberg at 17:53:24. This results in a travel time of 6 hours and 55 minutes.

### 4.2 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 Section 4.1) 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 seconds until no feasible train path can be found. The result is shown in Figure 4: a latest arrival time in Hallsberg between 17:52 and 24:00

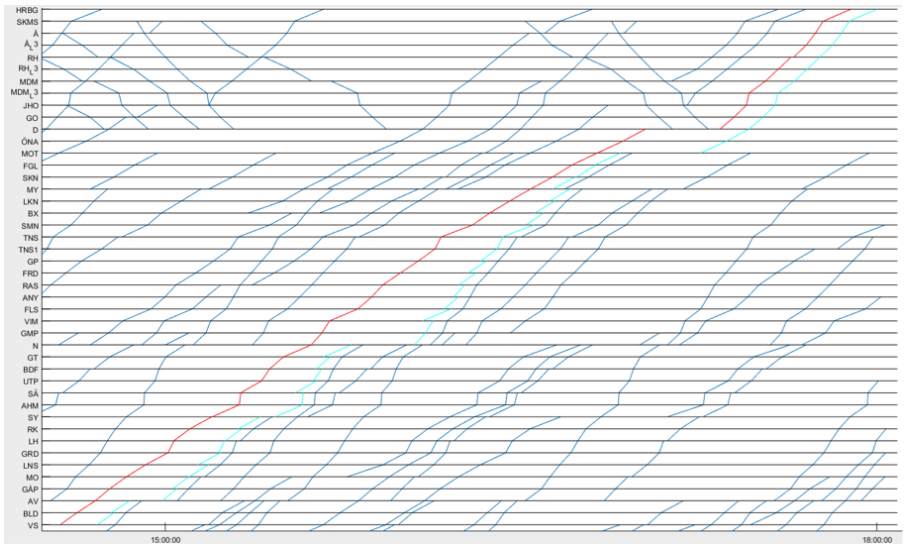
**Table 2** Station number, station code, and number of train departures for all stations on the stretch from Malmö to Hallsberg on February 24, 2015. S=station#, C=station code, D=# departures

| 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 | | | |

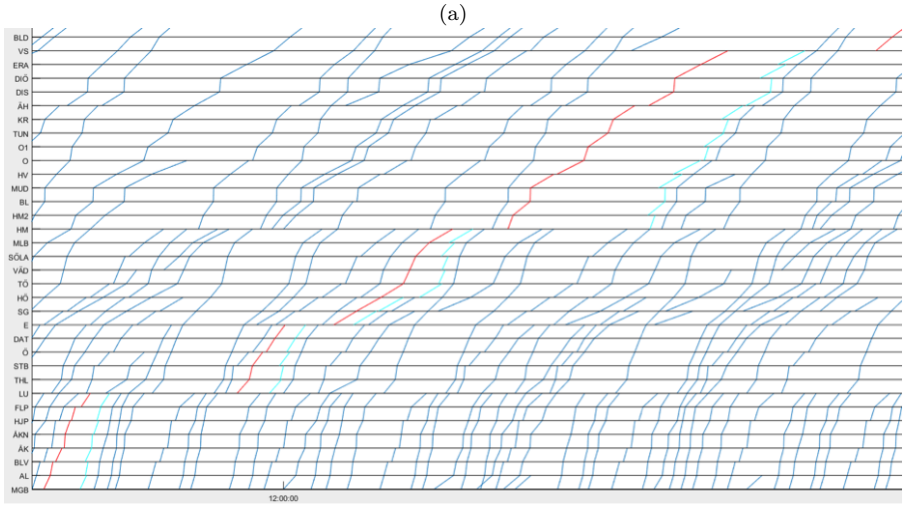
allows to insert a train path with robustness of 300 seconds, a latest arrival time between 16:04 and 17:50 allows to insert a train path with robustness of 147 seconds. The earliest feasible solution, which we obtained by running our algorithm with a resolution up to seconds, has an arrival time of 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 hours 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 minutes (which allow to keep the critical distance of 3 minutes 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




(a)




(b)

**Fig. 3** 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).

120 seconds until no feasible train path can be found. The result is shown in Figure 5: 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 seconds (ca. 23 minutes) 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


**Fig. 4** Robustness of the computed train path depending on the latest possible arrival time in Hallsberg.

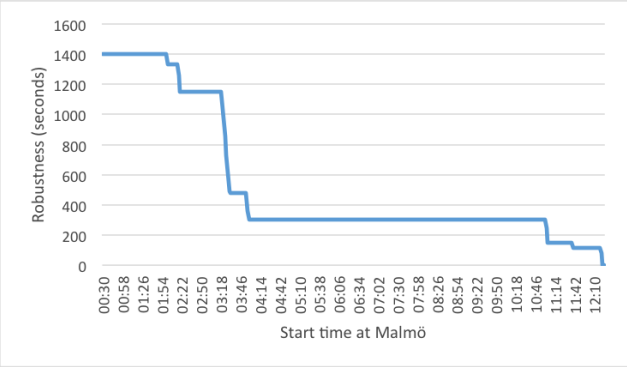



**Fig. 5** Robustness of the computed train path depending on the earliest possible departure time in Malmö.

reduces dramatically for a start time between 03:00 and 04:00: from 1149 seconds at 03:20 to 300 seconds 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.

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 Section 6.

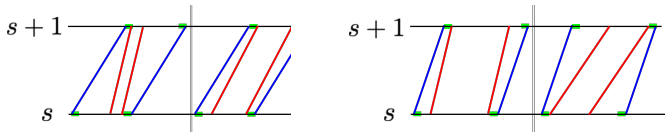



**Fig. 6** Existing trains between station $s$ and station $s+1$ are shown in blue, the earliest and latest possible departure for an inserted train is shown in red (limited by the safety distance to the existing trains shown in green). The distance between these two red lines defines the robustness 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).

Another observation, from both Figure 4 and Figure 5, 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

### 4.3 Variation of Train Type

In our base scenario in Subsection 4.1 we used a train of type GR421410, a freight train with a maximum weight of 1400 tons, a maximum speed of 100km/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 Figure 6. 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.

For each train type, we allowed a latest arrival time at 18:00, and then reduced this time stepwise by intervals of 45 minutes. 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 seconds longer runtime between Malmö and Hallsberg, train GT421209 has a 2300 seconds longer runtime, and train GR401608 has a 4360 seconds longer runtime. On the other hand, trains GR400710 and GR401409 take 22 seconds less than GR421410, for the two

**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 | - |

**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 |

passenger train types the difference is even larger: PX2-2000 and PX610016 need 6753 and 5546 seconds, 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 an 277, while freight trains yield a train path with 300 seconds robustness.

We take a more detailed look into this difference in robustness: in Figure 7 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.

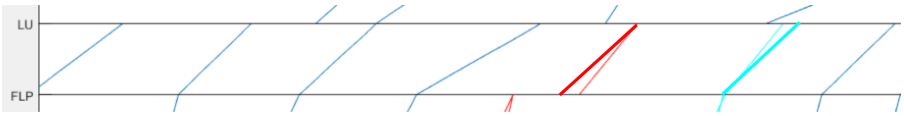



**Fig. 7** Earliest and latest possible train path of the passenger train PX2-2000 and the freight train GR421410 (bold lines) for the bottleneck shown in red and light blue, respectively. All existing trains in the time table are shown in dark blue.

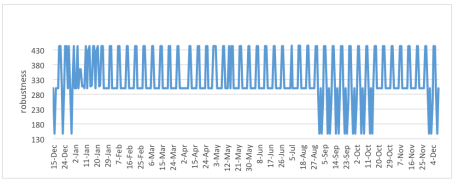
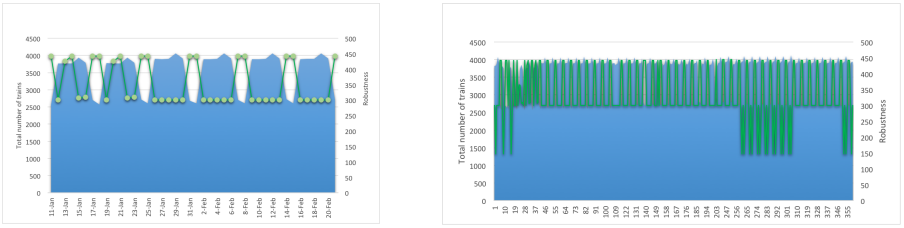



**Fig. 8** Maximum bottleneck robustness of the freight train obtained for all days fo the 2015 annual time table, except for April 3, 2015 - April 6, 2015 and July 6, 2015 - August 9, 2015.




(a) (b)

**Fig. 9** Number of trains per day and robustness of the inserted freight train path shown in blue and green, respectively. (a) In the interval January 11-February 21, 2015, (b) for the complete period of the 2015 annual time table.

## 4.4 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 number 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 to December 09, 2015) with the standard time window of 06:00 to 18:00. Figure 8 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).

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 Figure 9: 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 Figure 9(a) 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

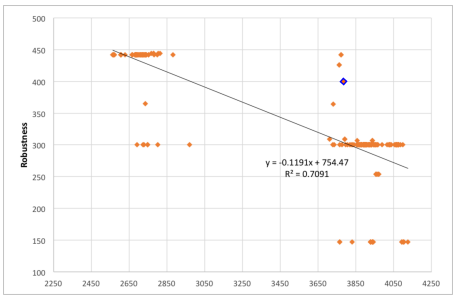
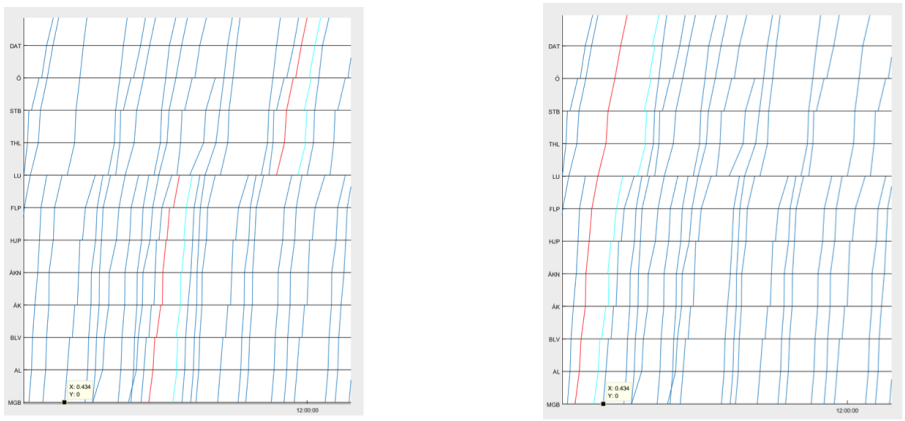



**Fig. 10** Robustness over number of trains in the time table. Actual data points, given by days from the 2015 annual time table, are shown in orange, the linear function obtained by regression analysis is shown in black.

closely, we performed a linear regression analysis; for the plot of both all single data points and the resulting regression line see Figure 10. 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 Figure 10 (the two data points overlap): January 14 and January 21, 2015, both days are Wednesdays. We compared these two days with the Wednesday of the succeeding week, January 28; see Figure 9(a) 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 Figure 11. 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.

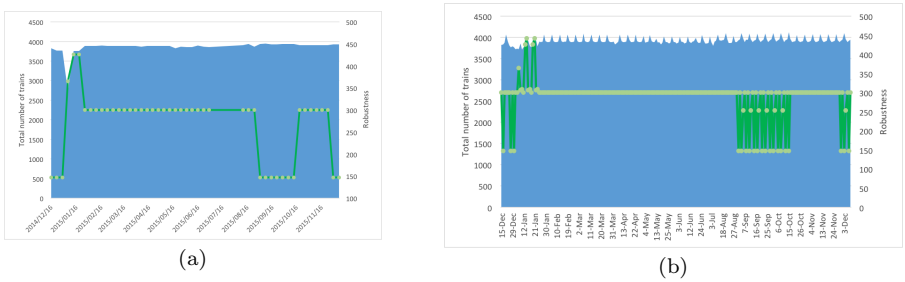
Finally, we consider all weekdays and all Tuesdays, to justify our choice of February 24, 2015 as a representative day of the 2015 annual time table. Figure 12(a) shows the number of existing trains per day and the robustness of the train path obtained by our algorithm for all weekdays (with all public holidays omitted), Figure 12(b) 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 robustness. We observe February 24 to be representative for the robustness range on the majority of weekdays in the 2015 annual time table.




(a) (b)

**Fig. 11** Analysis of the bottleneck of the inserted train path on January 28. The robustness on that day is 300 seconds, and the bottleneck occurs at station FLP (the last station before Lund). (a) and (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)

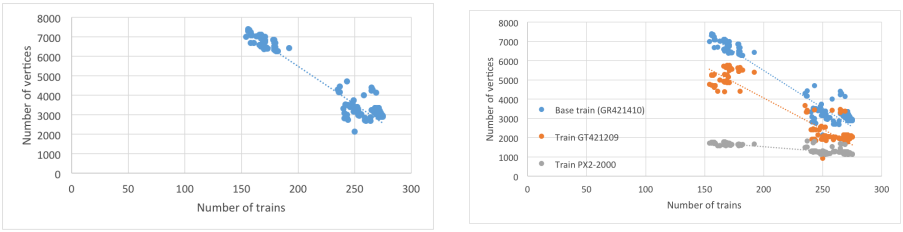
**Fig. 12** Number of trains per day and robustness of the inserted freight train path shown in blue and green, respectively. (a) For all tuesdays and (b) for all weekdays of the 2015 annual time table.

## 5 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:

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

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 $|E| \leq 2|V|$.




(a) (b)

**Fig. 13** Number of vertices over number of trains: (a) data points for different days, (b) comparison for different train types.

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 Subsection 3.1 significantly reduces the number of vertices in our experiments. For our base scenario with $|\mathcal{S}_\varphi| = 76$ and $|\mathcal{T}| = 251$ Equation (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, 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 Figure 13: the number of vertices actually decreases with the number of trains, this holds for different train types as shown in Figure 13(b).

## 6 Note on an Improved Algorithm for the "Best" Bottleneck Train Path

The algorithm presented in Section 3.2, and analyzed in detail in Section 4, has one obvious 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. Obviously, when we actually want to insert a freight train that influences the existing trains as little as possible, we would prefer the latter.

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.

**Algorithm 2:** Best Maximum Bottleneck Path

**INPUT :** Directed graph $G$, edge weights $c : E(G) \to \mathbb{R}_+$, 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.

More precisely: $\forall v \in V(G)$:
$r(v)$ - the value of a maximum bottleneck $s$-$v$-path.
$p(v)$ - the predecessor of $v$ on a maximum bottleneck $s$-$v$-path.
$R(v) = (r_{v,1}, \ldots, r_{v,M})$ - a vector of the robustness at all stations along a maximum bottleneck $s$-$v$-path (except for station 0).
1. Set $r(s) := \infty, r(v) := 0 \ \ \forall v \in V(G) \setminus \{s\}$, $R := \emptyset$.
$r_{v,i} = \infty \forall v \in V(G) \setminus \{s\}, \forall i \in \{1, \ldots, M\}$.
2. Find a vertex $v \in V(G) \setminus R$ with $r(v) = \max_{w \in V(G) \setminus R} r(w)$.
3. Set $R := R \cup \{v\}$.
4. **for** $\forall w \in V(G) \setminus R$ *with* $(v, w) \in E(G)$ **do**
    **if** $(r(w) < \min\{r(v), c((v, w))\}$ **then**
        Set $r(w) = \min\{r(v), c((v, w))\}$.
        $p(w) = v$.
        Set $R(w)$ to $R(v)$ extended by $c(v, w)$.
    **else if** $(r(w) = \min\{r(v), c((v, w))\}$ **then**
        **if** $(R(v) >_{lex} R(w))$ **then**
            $p(w) = v$.
            Set $R(w)$ to $R(v)$ extended by $c(v, w)$.
5. **if** $R \neq V(G)$ **then**
    THEN GOTO 2

## 7 Conclusions and Outlook

We presented an algorithm that can be implemented to insert one additional train in an existing timetable. The algorithms 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.

## Acknowledgments

This research is a result of a collaboration between Linköping University and Trafikverket, and part of the EU H2020 project Shift2Rail/ARCC (grant number 730813), and partially funded by Trafikverket (Dnr TRV 2016/75881). The authors are grateful to Magnus Wahlborg (Trafikverket) for fruitful discussions and Martin Aronsson (SICS RISE) for timetable data.

## References

1. R. Burdett and E. Kozan. Techniques for inserting additional trains into existing timetables. *Transportation Research Part B: Methodological*, 43(8):821 – 836, 2009.
2. V. Cacchiani, A. Caprara, and P. Toth. Scheduling extra freight trains on railway networks. *Transportation Research Part B: Methodological*, 44(2):215 – 231, 2010.
3. Commission of the European communities. Towards a rail network giving priority to freight, 2007.
4. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
5. Eurostat. Railway transport - goods transported, by type of transport, 2017.
6. H. Flier. Optimization of railway operations: Algorithms, complexity, and models, 2011.
7. H. Flier, T. Graffagnino, and M. Nunkesser. Scheduling additional trains on dense corridors. In *8th International Symposium on Experimental Algorithms (SEA 2009), Dortmund, Germany, June 4-6, 2009*, pages 149 –160, 2009.
8. M. Goerigk and A. Schöbel. Recovery-to-optimality: A new tow-stage approach to robustness with an application to aperiodic timetabling. *Computers & Operations Research*, 52:1–15, 2014.
9. I. A. Hansen and J. Pachl. *Railway Timetable & Traffic: Analysis - Modelling - Simulation*. Eurailpress in DVV Media Group, 2nd edition edition, 2014.
10. L. Ingolotti, F. Barber, P. Tormos, A. Lova, M. A. Salido, and M. Abril. *An Efficient Method to Schedule New Trains on a Heavily Loaded Railway Network*, pages 164–173. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
11. F. Khoshniyat and A. Peterson. Improving train service reliability by applying an effective timetable robustness strategy. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 2017.
12. L. Kroon, D. Huisman, and G. Maróti. Optimisation models for railway timetabling. In I. Hansen and J. Pachl, editors, *Railway Timetable & Traffic*, pages 135–154. Eurailpress: Hamburg, Germany, 2008.
13. C. Liebchen. The first optimized railway timetable in practice. *Transportation Science*, 42(4):420–435, 2008.
14. M.Grimm. The analysis of congested infrastructure and capacity utilisation at trafikverket. *WIT Transactions on the Built Environment*, 127, 2012.
15. M. Pollack. Letter to the editor—the maximum capacity through a network. *Operations Research*, 8(5):733–73, 1960.
16. E. Solinen, G. Nicholson, and A. Peterson. A microscopic evaluation of railway timetable robustness and critical points. *Journal of Rail Transport Planning & Management 5*, 2017.
17. J. Törnquist. Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms. *OASIcs-OpenAccess Series in Informatics*, 2006.
18. Trafikanalys. Rail traffic 2016, 2017.
19. Trafikanalys. Railway transport 2017 quarter 3, 2017.
20. UIC. UIC code 406: Capacity. Technical Report 1st edition, International Union of Railways, 2004.