An integrated micro–macro approach to robust railway timetabling

Nikola Bešinović, Rob M.P. Goverde, Egidio Quaglia, Roberto Roberti

A R T I C L E   I N F O

Article history:
Received 24 July 2015
Revised 22 February 2016
Accepted 23 February 2016
Available online 11 March 2016

Keywords:
Railway timetable
Feasibility
Robustness
Stability
Infrastructure occupation

A B S T R A C T

With the increasing demand for railway transportation infrastructure managers need improved automatic timetabling tools that provide feasible timetables with enhanced performance in short computation times. This paper proposes a hierarchical framework for timetable design which combines a microscopic and a macroscopic model of the network. The framework performs an iterative adjustment of train running and minimum headway times until a feasible and stable timetable has been generated at the macroscopic level. The macroscopic model optimizes a trade-off between minimal travel times and maximal robustness using an Integer Linear Programming formulation which includes a measure for delay recovery computed by an integrated delay propagation model in a Monte Carlo setting. The application to an area of the Dutch railway network shows the ability of the approach to automatically compute a feasible, stable and robust timetable. Practitioners can use this approach both for effective timetabling and post-evaluation of existing timetables.

1. Introduction

The recent growth in the demand for railway passengers and freight encourages infrastructure managers to improve efficiency of their networks in terms of higher infrastructure occupation and service quality (e.g., punctuality and travel times). Upgrading the infrastructure or the signaling system may help to achieve these objectives with the downside of massive financial investments. A cost-effective alternative is represented by designing effective timetables that can absorb everyday variations in running and dwell times while exploiting network capacity as much as possible. This means allocating as many trains as possible to the available infrastructure while guaranteeing sufficient time allowances (i.e., supplements and buffer times) to reduce delay propagation. In this context, timetable design must rely on accurate computations of realizable train paths and buffer times. Only a robust construction of train time-distance paths allows designing dense timetables which are operationally feasible, i.e., free from track conflicts including constraints imposed by the infrastructure layout and the safety and signaling systems.

In the literature, timetabling problems are most commonly modeled at a macroscopic level, usually referred to as the Periodic Event Scheduling Problem (PESP) (Serafini and Ukovich, 1989) or Train Timetabling Problem (TTP) (Caprara et al., 2002). Cacchiani and Toth (2012) gave an extensive review of variants of PESP and TTP, covering both nominal and robust approaches. Most of the models assume a macroscopic infrastructure, and as such neglect microscopic details.

* Corresponding author. Tel.: +31 152784914.
E-mail address: n.besinovic@tudelft.nl (N. Bešinović).

http://dx.doi.org/10.1016/j.trb.2016.02.004
0191-2615/© 2016 Elsevier Ltd. All rights reserved.
(e.g. signal position, switches) important for accurate timetabling. These models tend to use predefined norms for timetabling constraints such as default time values for train separation at stations and as such cannot guarantee timetable feasibility in practice. Therefore, macroscopic models must be upgraded or integrated with more detailed models if operational feasibility of the timetable must be ensured. To this end, different approaches have been proposed in the literature based on a hierarchical integration of timetabling models with different levels of detail. Schlechte et al. (2011) presented a bottom-up approach which first aggregates microscopic running and headway times to be used by a macroscopic model that identifies an optimized timetable for a given utility function, and then checks its feasibility by simulating it at a microscopic level. Middelkoop (2010) described the tool ROBERTO which uses a microscopic infrastructure model to compute accurate running and headway times which are then input to the macroscopic timetabling model DONS (Kroon et al., 2009). De Fabris et al. (2013) introduced a mesoscopic timetabling model which simplifies representation of station layouts to combine fast computation of macroscopic models with the accuracy of microscopic models. Caimi et al. (2011) extended PESP by proposing the flexible periodic event scheduling problem (FPESP), where intervals are used instead of fixed event times. By applying FPESP, the output does not define a final timetable but an input for finding a feasible timetable on a microscopic level (Caimi et al., 2011).

The main shortcomings with these approaches are that some do not perform any feasibility check of the timetable produced (Kroon et al., 2009; De Fabris et al., 2013), while others do not consider any iterative modification to the timetable if it is proved unfeasible at the microscopic level (Caimi et al., 2011; Schlechte et al., 2011). Another way of using a microscopic model to improve the outcome of a macroscopic model has been developed for the purpose of real-time railway traffic management. Kecman et al. (2013) tested the behavior of various macroscopic models and compared them with a microscopic one in order to determine the level of detail and operational constraints that are necessary to incorporate at the macroscopic level.

Within the European project ON-TIME (Optimal Network for Train Integration Management across Europe), we have developed a hierarchical iterative tool for the optimized design of railway timetables which combines microscopic, macroscopic and fine-tuning models (ON-TIME, 2015; Goverde et al., 2016). In this paper, we focus on the integration of a microscopic and a macroscopic model, which interact by iteratively updating macroscopic parameters that are re-computed at the microscopic level until the produced timetable is proved feasible, stable, and robust. This micro–macro timetabling approach has been applied to a real case study in the Netherlands. Experimental results show that our algorithms compute in short time a high quality timetable having an infrastructure occupation that satisfies UIC recommendations on capacity norms.

The main contributions presented in this paper are:

- A new timetabling approach that for the first time incorporates robustness in a micro–macro framework.
- An integrated iterative approach for computing a microscopically conflict-free and stable timetable that is optimized at a network level.
- An automatic procedure to adapt macroscopic input by constraint relaxation and tightening methods at the microscopic level.
- A macroscopic timetable optimization model with a post-processing Monte Carlo stochastic robustness evaluation of the generated timetables.

The remainder of the paper is organized as follows. The next section defines the problem statement of this paper and introduces the framework of the micro–macro approach. Section 3 describes the network and data modeling at different level of details and the automatic transformations between these networks. Sections 4 and 5 elaborate on the microscopic and macroscopic models of the approach, while Section 6 describes the applied constraint adaptations between successive iterations. The case study is presented in Section 7, while conclusions are provided in Section 8.

2. Problem description

2.1. Problem definitions

We adopt the definitions from Goverde et al. (2016). A line service is defined with origin and destination points, stopping pattern, i.e. served timetable points (stations, stops), and a corresponding rolling stock type. It also includes the service category, such as local or intercity, and the frequency represented in number of trains per hour. A train path is the infrastructure capacity needed to run a train between two places over a given time period (EC, 2001). A conflict is determined as an overlap (in time and space) between two train paths and entails that one train cannot use the railway infrastructure without interfering with another train. Timetable efficiency reflects the amount of time allowances in the scheduled travel times (running, dwell and transfer times) which must be as short as possible to provide short journey times and seamless connections. Timetable feasibility is the ability of all trains to adhere to their scheduled train paths. A timetable is feasible if (i) the individual processes are realizable within their scheduled process times, and (ii) the scheduled train paths are conflict free, i.e., all trains can proceed undisturbed by other traffic. Timetable stability is the ability of a timetable to absorb initial and primary delays so that delayed trains return to their scheduled train paths without rescheduling. This is directly connected with the infrastructure occupation rate. The higher this rate, the lower are the time allowances and hence the less stable is the timetable. Timetable robustness is the ability of a timetable to withstand design errors, parameter variations, and changing operational conditions.
We distinguish between a microscopic and macroscopic timetable. A *macroscopic timetable*, or MacroTT, includes scheduled running, dwell and transfer times, as well as event times such as arrivals, departures and passages between and at relevant timetable points (introduced in Section 3.1). A *microscopic timetable*, or MicroTT, includes scheduled running, dwell and transfer times as well as event times such as arrivals, departures and passages for microscopic timetable points (introduced in Section 3.1) and the corresponding train speed profiles defining the exact train trajectories. A MacroTT is the outcome of the macroscopic model and is analyzed by the microscopic one, while the MicroTT is the final output of the developed framework.

Given infrastructure and rolling stock characteristics, and the requested line services, the Train Timetabling Problem (TTP) consists of finding a feasible, efficient, stable and robust microscopic timetable.

### 2.2. The timetable planning framework

We propose a micro–macro framework to solve the TTP and design a railway timetable. The structure of such a framework is shown in Fig. 1, which indicates the interactions among the different models, their functions as well as the

![Fig. 1. SysML scheme of the hierarchical framework for timetable design.](image-url)
input-output data flow. The framework is implemented using a standardized RailML interface (RailML, 2015; Bosschaart et al., 2015). In particular, the RailML data required for the initialization of the models relate to characteristics of the infrastructure (i.e., gradients, speed limits, positions of stations, switches), rolling stock (i.e., mass, length, braking rate, tractive effort-speed curve), interlocking (i.e., alternative local routes), signaling system (e.g., position and type of signals, automatic train protection (ATP) behaviour), and routes/stopping pattern of the train services to be scheduled. Both input and output data of the framework are in RailML format, which is being developed with the goal of becoming a standard in Europe for communication among railway software tools.

Timetabling computation is an iterative process of two different models: a microscopic and a macroscopic model. The microscopic model computes reliable train running and headway times at a highly-detailed local level and checks for feasibility and stability of the timetable. The macroscopic model has an aggregated infrastructure representation and produces a timetable for the entire network by identifying arrival/departure times at/from stations or junctions which optimize a given objective function (e.g., minimize journey times). Such a macroscopic model includes methods for estimating delay propagation to assess produced timetables in terms of robustness versus stochastic operation disturbances.

In the first iteration a timetable is not available yet, so the microscopic model computes minimum running times and headways that are sent to the macroscopic model to calculate a timetable. The achieved macroscopic timetable is sent back to the microscopic model that, based on the operational running times (i.e. the running times including time supplements scheduled by the macroscopic timetable), computes train blocking times necessary for detecting track conflicts. If there are track conflicts, these are solved and new headways and running times are computed and transferred to the macro model again. The macroscopic model solves an optimization problem which incorporates heuristics with an integer linear programming problem minimizing a weighted sum of running, dwell and transfer times, and a robustness cost. The robustness cost is defined as the delay settling time obtained from a Monte Carlo simulation of the delay propagation for given candidate timetable solutions. This iterative process is repeated until no more track conflict is detected and the timetable is thus feasible at both the macroscopic and microscopic levels. Once feasibility is achieved, the microscopic model evaluates the stability of the timetable (i.e., the capability in absorbing delays). If the timetable is not stable, new operational running times are computed by e.g. increasing the value of time supplements and/or buffer times. This is performed until timetable stability is verified to have reached the required level (UIC, 2013). Transformations from the microscopic level to the macroscopic and vice versa require appropriate procedures that have been developed to aggregate/disaggregate input and output data. This interaction continues until the timetable produced by the macroscopic model is proved to be microscopically feasible and stable. As a result, the final output of the framework is a feasible and stable timetable with a suboptimal trade-off between efficiency and robustness.

3. Network and data modeling

In our approach, input and output data are exchanged between models with different levels of detail, so consistency in data flows must be guaranteed. This is achieved by automatic data transformation (aggregation/disaggregation) processes that we describe in this section.

3.1. Network representation

The hierarchical framework for timetable design is composed of two models that represent the same network but with a microscopic and a macroscopic level of detail.

The microscopic graph \( G = (X, B) \) represents the network at the level of homogeneous behavioral sections (i.e. sections with constant values of speed limit, gradient and curvature radius). Each infrastructure section (arc), \( b_j \in B \), is described with constant characteristics of speed limit \( v_i \), gradient \( g_i \) and radius \( \rho_i \) and given length \( l_i \), i.e., \( b_i = (v_i, g_i, \rho_i, l_i) \). Microscopic nodes \( x \in X \) represent both points in which these characteristics change and infrastructure elements like block section joints, switches, and station platforms. This detailed microscopic model is used to aggregate the homogeneous behavioral sections in block sections for open tracks and in track-free detection sections for interlocking areas which are required for considering sectional-release route-locking behaviour. This level of detail is important for computing blocking times, conflict detection, and infrastructure occupation.

On top of the microscopic network, a discrete set of microscopic timetable points \( K \subset X \) is defined. A microscopic timetable point \( k \in K \) represents an infrastructure point where an interaction exists between a train and passengers (boarding and alighting) or cargo (loading and unloading), or between two trains (converging and diverging tracks). These microscopic points therefore define important discrete events at stations, junctions, bridges or tunnels.

The macroscopic network is represented by a multi-graph \( N = (S, A \cup E) \), where the vertices represent a set \( S \subset K \) of macroscopic points corresponding to the microscopic timetable points that allow interaction between trains, i.e., meeting, overtaking, or connections. In the remainder, we refer to macroscopic points as timetable points. The macro tracks are defined as mono-directional \( a = (s_1, s_2) \) or bidirectional \( c = (s_1, s_2) \). The sets of arcs \( A \) and edges \( E \) represent mono-directional and bidirectional tracks between pairs of timetable points, respectively. For each timetable point \( s_i \in S \), the capacity \( cap_{s_i} \) for each station is known and corresponds to the number of tracks. For each pair of timetable points \( s_i, s_j \in S \), we consider that the number of directed arcs (mono-directional tracks) \( \alpha_{s_i s_j} \) from \( i \) to \( j \), the number of directed arcs \( \alpha_{s_j s_i} \) from \( j \) to \( i \), and the number of edges (bidirectional tracks) \( \beta_{s_i s_j} \) are known.
Inputs from the basic architecture consists of a set of data expressed in RailML format, namely: a) Microscopic infrastructure data, b) rolling stock data, including train formations, c) Interlocking, signaling and ATP, d) available routes, and e) train line requests. These data are converted to a suitable internal format of ASCII data that is used by the microscopic computation models. Additional parameters, such as connections, dwell times, timetable design parameters and quality norms are provided externally.

3.2. Trains, train lines and routes

Let $T$ be the set of all trains in the network and $L$ be the set of all lines of trains, i.e. $L = \{L_1, L_2, \ldots, L_l\}$, where $\bar{L} = |L|$, $L_j \subseteq T$, $j = 1, \ldots, \bar{L}$, $\bigcup_{j=1}^{\bar{L}} L_j = T$, and $L_j \cap L_k = \emptyset$, $1 \leq j, k \leq |L|$, $j \neq k$. For each line $L_j$, $j = 1, \ldots, \bar{L}$, the period time $per_j$ is given, representing the ideal interval time between the stops at each station of two consecutive trains from the same line. Moreover, let $S_j \subseteq S$ be the set of stations served by line $j \in L$. For each train $L_j \in T$, $l(j)$ indicates the corresponding line. We assume that the trains of a given family $L_j = \{t_1, t_2, \ldots, t_{T_j}\}$, $j = 1, \ldots, \bar{L}$, are ordered in increasing order of departure times.

Moreover, we assume that for each train, its route $\rho_t$ (i.e., the sequence of traversed tracks without the corresponding travel times) is provided. Finally, we differentiate between a microscopic route $\rho_t^{\text{micro}} = (b_1, b_2, \ldots, b_n)$, where $n_t$ is the number of microscopic tracks, and a macroscopic one $\rho_t^{\text{macro}} = (a_1, a_2, \ldots, a_m)$, where $m_t$ is the number of macroscopic tracks.

For each train $t \in T$ and each arc $a \in A$, (each edge $e \in E$, resp.) the minimum running time $L_a$, the nominal running time $t_a$, the maximum running time $T_a$, and the maximum running time $T_a$ are given. All running times are computed by microscopic algorithms, while the nominal and maximum ones are given as input to the macroscopic module. Also provided for each train $t \in T$ is the maximum journey time $\bar{f}$ from origin to destination.

The algorithms developed within the ON-TIME project, both microscopic and macroscopic, are suitable for both periodic and non-periodic timetabling. In the non-periodic case, each line contains exactly one train. Therefore, the microscopic tools may accept both trains and train lines without any adjustment of the algorithms.

3.3. Other parameters

For each train $t \in T$ and each station $s \in S_t$, the nominal dwell time $w_{ts}$ and maximum dwell time $\bar{w}_{ts}$ is provided. Let $Q$ be the set of all connections between pairs of trains at given stations. Then each connection $q = (t_1, t_2, s)$ with $t_1, t_2 \in T$, $s \in S$, is characterized by a nominal and a maximum connection time, $q_{ts}$ and $\bar{q}_{ts}$, respectively, which need to be respected by connection constraints. Since the aim of timetable planning is to provide an acceptable quality of service, certain design norms need to be predefined. The set of these parameters consists of minimum and maximum transfer times, turnaround times, minimum and maximum running time supplements (\%), and maximum allowed journey times of train lines (\%). The set of timetable design norms is named $\Lambda$.

3.4. Microscopic to macroscopic conversion

Microscopic data are migrated to the macroscopic level using the procedure described in Algorithm 1, which is comparable with the one implemented by Schlechte et al. (2011). The conversion from microscopic to macroscopic is instead done in two steps. First, the subset $S$ from $K$ is derived. The algorithm compares all pairs of train lines. The macroscopic nodes are chosen based on the interplay between train routes. The set $S$ includes only microscopic timetable points if two train routes are converging, diverging or crossing. This process of defining the set of timetable points is done automatically. Second, it consists of the aggregation of microscopic arcs $b_t$ to macroscopic arcs $a$ or edges $e$, $a = (b_1, b_2, \ldots, b_n)$ or $e = (b_1, b_2, \ldots, b_n)$. For each arc $a$ and edge $e$ the following data is determined: 1) the number of tracks $(\alpha a_i b_i)$ by identifying different routes between two nodes using function $\text{DetermineTracks}$, and 2) the orientation (mono- or bidirectional) for each of them by function $\text{DetermineDirection}$.

We will describe the inclusion into $S$ with an example. Consider two train lines that use the same successive microscopic timetable points $(k_1, k_2, k_3)$. Then the set $S$ will include $(k_1, k_3)$, since $k_1, k_3$ are the origin and destination of both lines, while $k_2 \in S$ as both trains use the same points before and after this. As another example, consider two train lines, where the first train line uses $(k_1, k_2, k_3)$ and the second one $(k_1, k_2, k_4)$, then $S$ will be equal to $(k_1, k_2, k_3, k_4)$ since the train lines diverge in $k_2$, which makes $k_2 \in S$.

After having initialized the macroscopic network, the microscopic model is activated to compute microscopic minimum running times (over homogeneous behavioral sections), blocking times and minimum headways. Headways are determined for all possible interactions between each two train routes (inbound–inbound, in–out, out–in, out–out) at every macroscopic timetable point $s$. The last two are executed on the block section level of the infrastructure network.

Once all process times are computed on the microscopic model, we carry out the aggregation of process times and the discretization of time. The function $\text{AggregateProcessTimes}$ is introduced to provide mitigation from microscopic running times (i.e., between any consecutive microscopic points) to aggregated process times between any consecutive timetable points in the macroscopic network. Since the macroscopic model uses a coarser time granularity, the time discretization of process times is performed as well. The incorporated function represents an innovative rounding method that has the objective to control the rounding error by combining rounding up and rounding down. By applying $\text{AggregateProcessTimes}$, we obtain all process times that are necessary for macroscopic computation.
Algorithm 1
Microscopic to macroscopic conversion.

**Input:** Microscopic network, microscopic timetable points, dwell times, timetable design norms, set of train lines

**Output:** \( N = (S, A \cup E) \), process times

- For all microscopic timetable points \( k \in K \)
- For all pairs of train lines
  - If \( k \) is origin or destination point OR lines converge OR lines diverge
    - OR lines cross
      - Include \( k \) in the set of macroscopic timetable points

End If
End For
End For

For all adjacent timetable points

- **DetermineTracks**
- **DetermineDirection**
- If track is used in both directions
  - Create edge \( e \)
- Else
  - Create arc \( a \)

End If
End For

Running times computation

Blocking times computation

- For all macroscopic timetable points \( s \in S \)
  - Minimum headway computation

End For

AggregateProcessTimes

Algorithm 2
Microscopic to macroscopic conversion.

**Input:** microscopic network, MacroTT

**Output:** Feasible, stable and robust timetable

- For all trains
  - Determine allocated running time supplements
  - Compute operational running times
  - Compute blocking times

End For

Conflict detection

- If timetable is not conflict-free
  - Constraint Tightening
  - Run Macroscopic optimization
- Else
  - Capacity evaluation
  - If capacity norms are not satisfied
    - Constraint Relaxation
    - Run Macroscopic optimization

End If
End If

3.5. Macroscopic to microscopic conversion

To convert data from the MacroTT to the MicroTT we use the process described in Algorithm 2. Substantially, from the scheduled event times for the macroscopic timetable points we have to reconstruct the corresponding train trajectories and scheduled times for all of the microscopic timetable points. The details of each module used in this description are given in Sections 4 and 5.

Starting with MacroTT, we determine the scheduled running time over each macroscopic edge (arc), as the difference between the arrival in one station and the departure from the preceding station. Further, we compute the corresponding allocated running time supplement \( \Psi_t \) as the difference between the scheduled and minimum running time for each train \( t \in T \), where \( T \) is the set of all trains. This defines a vector \( \Psi_t \) of the time supplements \( \Psi_t \) over each two macroscopic timetable points. For each train \( t \in T \) and the corresponding \( \Psi_t \) we compute the operational running time consisting of the detailed train trajectory and scheduled times at microscopic timetable points, which are used for further microscopic analyses. Consequently, the blocking times are computed for all trains.

Once blocking times are computed, conflict detection and capacity assessment are performed, and if one of those is not satisfied then train process times (headways and running times) are updated by applying constraint tightening and/or relaxation (cf. Section 6). Updated process times are sent again to the optimization model described in Section 5 to recompute MacroTT. After a new MacroTT has been obtained, Algorithm 2 starts from the beginning. Once MacroTT satisfies
an acceptable quality of service, the algorithm terminates. The final output of the model is a feasible, stable and robust timetable.

4. Microscopic timetabling

The microscopic module consist of the following computation functions: minimum and operational running times, blocking and headway times, conflict detection and resolution and capacity assessment. A description of each function is given in the next paragraphs.

4.1. Running times

The minimum running time is the time required for driving a train from one infrastructure point to another infrastructure point assuming conflict-free driving as fast as possible. In this section we will focus on the running time between two stations. Running time computation considers detailed characteristics of the infrastructure (i.e., gradients, speed limits, positions of stations, switches), rolling stock (i.e., mass, composition, braking rate, tractive effort–speed curve), signaling system (e.g., position and type of signals), and routes/stopping pattern of the train services to be scheduled. Running times are computed for every line service by means of dynamic Newton’s motion equations (Hansen and Pachl, 2014) according to the implementation described in Bešinović et al. (2013).

As already explained, MacroTT represents the scheduled running time between two macroscopic timetable points, which consists of the sum of the minimum running times and time supplements (usually 5–7% of the minimum running time) to recover from statistical variations during real operations. This means that from the scheduled running time given by MacroTT we must be able to retrieve the corresponding microscopic train trajectory (speed-distance, time-distance diagrams) in the MicroTT that satisfies that scheduled running time (described in Section 3.5). Such train trajectories incorporate the available time supplements. The operational running time represents the recomputed train trajectory that satisfies the scheduled running time between two timetable points. This trajectory will exploit associated running time supplements by applying cruising with a speed lower than the maximum speed, and to do so we implemented a customized bisection algorithm (Bešinović et al., 2015).

4.2. Blocking times

The blocking time of a section of track (block section or interlocked route) is the time duration that the section is exclusively allocated to a train and therefore blocked to other trains. Blocking times are computed in function Blocking times computation by applying the procedure described in Hansen and Pachl (2014).

The blocking time of a train for a given block section is composed of the following components: setup time \( t_{\text{setup},i} \) to set the route for the train approaching, sight and reaction time \( t_{\text{sight},i} \) of the train driver when approaching the approach signal, approach time \( t_{\text{approach},i} \) needed by the train to traverse the braking distance from the approach signal to the main signal, the running time \( t_{\text{block},i} \) of the train to traverse block section \( i \), the clearing time \( t_{\text{clear},i} \) to clear the block section over the train length, and the release time \( t_{\text{release},i} \) to release the route after the train clearance. After having provided all these terms the blocking time \( d_{ii} \) of the train \( t \) relative to block \( i \) is obtained as:

\[
d_{ii} = t_{\text{setup},i} + t_{\text{sight},i} + t_{\text{approach},i} + t_{\text{block},i} + t_{\text{clear},i} + t_{\text{release},i}.
\]

Each blocking time \( d_{ii} \) of section \( i \) by train \( t \) is specified from the start \( d^s \) to the end \( d^e \) of the blocking time. Hence, \( d_{ii} = (d_{ii}^s, d_{ii}^e) \).

4.3. Minimum headway times

The minimum headway time between two trains is the time separation that prevents the trains from having track conflicts with each other. Here we introduce the computation of one minimum headway at a timetable point \( s \in S \). Let \( B_{ijs} \) be the set of shared blocks associated to routes of both trains \( i \) and \( j \) in timetable point \( s \), \( d_{ij}^s \) be the end of blocking time \( d_{il} \) and \( d_{ij}^e \) the start of blocking time \( d_{jl} \). Let us assume that both trains have the same reference event (i.e., departure, arrival or passing) time at \( s \), e.g., equal to 0. Then the minimum headway \( h_{ijs} \) from train \( i \) to \( j \) in timetable point \( s \) is computed as

\[
h_{ijs} = \max_{e \in B_{ijs}} (d_{ij}^s - d_{ij}^e).
\]

For each pair of trains \( t_1, t_2 \in T \) and each timetable point \( s \in S \), we compute the nominal headway time \( h_{t_1t_2s}^{dd} \) (\( h_{t_1t_2s}^{da} \)) between the departure of train \( t_1 \) from timetable point \( s \) and the departure (arrival) of train \( t_2 \) from (at) timetable point \( s \), and the nominal headway time \( h_{t_1t_2s}^{dd} \) (\( h_{t_1t_2s}^{aa} \)) between the arrival of train \( t_1 \) at timetable point \( s \) and the departure (arrival) of train \( t_2 \) from (at) timetable point \( s \). Any headway time is equal to 0 whenever the two trains do not meet at a timetable point.
4.4. Conflict detection

The aim of conflict detection is to verify the feasibility of the macroscopic timetable by checking: a) the absence of track conflicts and b) the realizability of scheduled process times (i.e., running times, dwell times, turnaround times). Track conflicts are detected as partial or full overlaps of the blocking times provided by the BlockingTimes function. Therefore, conflict-freeness is tested comparing the interaction of scheduled blocking times for each pair of trains, i.e., checking the possible blocking times overlap between them. The blocking time overlap $c_{ij}^\varphi$ from train line $i$ to $j$ at corridor $\varphi$ is computed similarly as the minimum headway times:

$$c_{ij}^\varphi = \max_{\forall k \in \mathbb{B}_\varphi} (d_{ij}^k - d_{ij}^k),$$

where $\mathbb{B}_\varphi$ is the set of successive blocks at corridor $\varphi$, and the scheduled start and end of the blocking times are used. If $c_{ij}^\varphi > 0$ then a conflict exists. Usually, a corridor corresponds to a macroscopic arc (or edge). In this way, the whole network is considered by the conflict detection algorithm, and not only timetable points.

Realizability is tested by checking if the scheduled running and dwell times exceed the corresponding minimum technical values. Note that the macroscopic timetabling model is such that it always provides realizable scheduled times, so the realizability check can then be omitted.

4.5. Infrastructure occupation

Infrastructure occupation is defined as the time share needed to operate trains according to a given timetable pattern taking into account scheduled running and dwell times. The infrastructure occupation $\mu(\varphi)$ of corridor $\varphi$ can be obtained by:

$$\mu(\varphi) = \sum_{(i,j) \in W^\varphi} h_{ij}^\varphi,$$

with $W^\varphi$ the cyclic pattern of successive train pairs $(ij)$ in corridor $\varphi$, and $h_{ij}^\varphi$ the minimum line headway on this corridor. The latter is computed similarly as local minimum headway, where blocks at a corridor $\varphi$, instead of a timetable point $s$, are considered. A corridor may be equal to an arc (or edge) or comprise several adjacent arcs (edges), $\varphi = \cup_\varphi$. We compute the infrastructure occupation for each corridor $\varphi \in \Phi$, applying an algorithm based on max-plus automata theory (Gaubert and Mairesse, 1999; Bešinović et al., 2015).

5. Macroscopic timetabling

The macroscopic timetable optimization algorithm iteratively communicates with the microscopic module in order to achieve a timetable that is both macroscopically and microscopically feasible.

The macroscopic optimization module receives as input the following data from the microscopic calculation module: a) railway infrastructure aggregated at macroscopic level (including capacity of arcs/edges and macroscopic timetable points), b) a set of train lines to schedule with the corresponding routes, c) headway times between pairs of train lines meeting at timetable points, d) nominal and maximum running and dwelling times along the routes of each train line, and e) a set of connections (where a connection states that, at a given timetable point, the departing time of a train must be within a given time interval from the arrival time of the previous train).

The macroscopic timetable computation provides the microscopic module with a macroscopic timetable that consists of a set of paths (at most one for each train) and the indication of the trains that are cancelled – trains can be cancelled if all corresponding feasible paths violate some of the constraints defined on the network (e.g., headway times, capacity). A path of a train is an ordered sequence of tracks and provides, for each of the traversed tracks, the times where the trains enter and leave the track. Furthermore, Monte Carlo simulation is applied to obtain a timetable with improved robustness. The timetable provided by the macroscopic module is not only feasible, but also robust.

From an algorithmic point of view, the macroscopic timetable computation consists of a randomized multi-start greedy heuristic (hereby referred to as MacroHeu) that iteratively generates a set of feasible timetables and, among them, selects the one having the minimum cost. The cost of a timetable takes into account properly defined penalties plus a specific penalty that considers its robustness. To assess the robustness of a timetable, a number of different scenarios (each one featuring a randomly generated delay for each train) are considered and evaluated in terms of absorption of the delays.

The macroscopic timetable computation algorithm may be run several times in a loop with the microscopic module exchanging information based on tightening and/or relaxing constraints, in order to guarantee that the final macroscopic timetable is also feasible from a microscopic perspective. For this reason, MacroHeu needs to implement a relatively easy methodology that can provide a good solution within limited computing times (i.e., tens of seconds).

5.1. Optimization algorithm

The problem addressed by the macroscopic module can be formulated as an ILP model with four different types of binary and integer variables and an exponential number of constraints.
Let $P_l$ be set of all feasible paths of each train of line $l = 1, \ldots, \bar{L}$, where a path $p$ is an ordered sequence of arrival and departure times for each timetable point of the set $S_l$, defined as $(\tau_{t_{1}t_{2}p}, \tau_{t_{1}t_{2}p}, \tau_{t_{1}t_{2}p}, \ldots, \tau_{t_{1}t_{2}p})$, where $\tau_{t_{1}t_{2}p}$ represents the departure time at timetable point $s \in S_l \setminus [S_l]$ of train $l$ over path $p$ and $\tau_{t_{1}t_{2}p}$ is the arrival time at timetable point $s \in S_l \setminus [S_l]$ of train $t$. A path $p$ is feasible if it satisfies the following constraints:

- all timetable points of the set $S_l$ are visited according to the given route $\rho_{l}^{macro}$;
- the total maximum journey time $\tilde{t}_p$ is not exceeded;
- for each $i = 1, \ldots, |S_l| - 1$, the difference between the arrival time $\tau_{t_{i}t_{i+1}p}$ at timetable point $s_{i+1}$ and the departure time $\tau_{t_{i}t_{i}p}$ at timetable point $s_i$ is at least $\tau_{a}$ and does not exceed $\tilde{\tau}_{a}$, with $a = (s_i, s_{i+1})$, that is $\tau_{a} \leq \tau_{t_{i}t_{i+1}p} - \tau_{t_{i}t_{i}p} \leq \tilde{\tau}_{a}$;
- for each $i = 1, \ldots, |S_l| - 1$, the difference between the departure time $\tau_{t_{i}t_{i}p}$ and the arrival time $\tau_{t_{i}t_{i}p}$ at timetable point $s_i$ is at least $\tau_{s+$ and does not exceed $\tilde{\tau}_{s}$, that is $\tau_{s_+} \leq \tau_{t_{i}t_{i+1}p} - \tau_{t_{i}t_{i}p} \leq \tilde{\tau}_{s}$.

In addition, the following penalties are defined:

- $\pi_{l}^{conc}$: penalty paid for cancelling the trains of line $l \in L$;
- $\pi_{l}^{run}$: penalty for each unit time of running time exceeding the nominal one for trains of line $l \in L$;
- $\pi_{l}^{dwell}$: penalty for each unit of dwell time exceeding the nominal one for trains of line $l \in L$;
- $\pi_{Q}^{conn}$: penalty for the connection time exceeding $u_q$ for connection $q \in Q$;
- $\pi_{Q}^{conn}$: penalty for missing connection $q \in Q$.

The cost $c_p$ of path $p \in P_l$ that is assigned to the trains of line $l = 1, \ldots, \bar{L}$, is given by the running and dwell time exceeding the nominal ones penalized according to penalties $\pi_{l}^{run}$ and $\pi_{l}^{dwell}$, respectively, that is:

$$c_p = \pi_{l}^{run} \sum_{l = 1}^{\bar{L}} \sum_{p \in P_l} (|S_l| - 1) \left( \tau_{t_{i}t_{i+1}p} - \tau_{t_{i}t_{i}p} \right) - \pi_{l}^{dwell} \sum_{l = 1}^{\bar{L}} \sum_{p \in P_l} (|S_l| - 1) \left( \tau_{t_{i}t_{i}p} - \tau_{t_{i}t_{i}p} \right).$$

Let $\mathcal{U}$ be the set of all paths cliques, where each path clique $U \in \mathcal{U}$ is a subset of the paths of all lines (i.e., $U \subseteq \bigcup_{l \in L} P_l$) which may have conflicts with each other. This means that at most one of such paths can be in the solution simultaneously because any pair of those paths violate constraints on headway times and/or station/arc capacity.

By introducing the following sets of variables:

- Binary variable $x_p$ equal to 1 if path $p \in P_l$ of trains of line $l \in L$ is selected (0 otherwise),
- Binary variable $\xi_l$ equal to 1 if all trains of line $l \in L$ are cancelled (0 otherwise),
- Integer variable $y_q$ representing the connection time exceeding $u_q$ for connection $q \in Q$ if connection $q \in Q$ is not missed,
- Binary variable $\chi_q$ equal to 1 if connection $q \in Q$ is missed (0 otherwise),

the problem addressed by the macroscopic module can be formulated as the following ILP:

$$\begin{align*}
\min \quad & \sum_{l \in L} \sum_{p \in P_l} c_p x_p + \sum_{l \in L} \pi_{l}^{conc} \xi_l + \sum_{q \in Q} \pi_{Q}^{conn} y_q + \sum_{q \in Q} \pi_{Q}^{conn} \chi_q \\
\text{s.t.} \quad & \xi_l + \sum_{p \in P_l} x_p = 1 \quad l \in L \quad (5) \\
& \xi_{l(t_1)} + \xi_{l(t_2)} \leq 2 \chi_q \quad q = (t_1, t_2, s) \in Q \quad (6) \\
& \sum_{p \in P_{l(t_1)}} \tau_{t_{i}t_{i}p} x_p - \sum_{p \in P_{l(t_1)}} \tau_{t_{i}t_{i}p} x_p \geq u_q - M \chi_q \quad q = (t_1, t_2, s) \in Q \quad (7) \\
& \sum_{p \in P_{l(t_2)}} \tau_{t_{i}t_{i}p} x_p - \sum_{p \in P_{l(t_2)}} \tau_{t_{i}t_{i}p} x_p \leq u_q + M \chi_q \quad q = (t_1, t_2, s) \in Q \quad (8) \\
& \sum_{p \in U} x_p \leq 1 \quad U \in \mathcal{U} \quad (9) \\
& x_p \in \{0, 1\} \quad p \in P_l, l \in L \quad (10) \\
& \xi_l \in \{0, 1\} \quad l \in L \quad (11) \\
& y_q \in \{0, u_q - \bar{u}_q\} \quad q \in Q \quad (12) \\
& \chi_q \in \{0, 1\} \quad q \in Q \quad (13) 
\end{align*}$$

where $M$ is a large enough number.

The objective function (5) guarantees that the timetable achieved minimizes the total cost, given by the sum of: (a) the cost of the paths selected; (b) the cost for cancelling trains; (c) the cost for exceeding the nominal connection time; and
Algorithm 3
Step-by-step description of macroscopic optimization.

Input: macroscopic network, set of trains, routes, headway times, running times, dwell times, connections
Output: a feasible and robust macroscopic timetable MacroTT of cost \( c^{\text{MacroTT}} \)
Initialize MacroTT := \( \emptyset \) and \( c^{\text{MacroTT}} := \infty \)
For \( \text{iter} = 1, \ldots, \text{iter} \) Do
  Initialize CurrTT := \( \emptyset \), \( c^{\text{CurrTT}} := 0 \), and LeftLines := L
  While LeftLines \( \neq \emptyset \) Do
    Randomly select a line \( l \) from LeftLines
    Determine (see Appendix A) the min-cost path \( p \in P \) for line \( l \) that does not conflict with any of the paths of the set CurrTT
    If a path \( p \in P \) was found Then
      Update CurrTT := CurrTT \( \cup \) \( \{p\} \) and \( c^{\text{CurrTT}} := c^{\text{CurrTT}} + c_p \)
    Otherwise
      Update \( c^{\text{CurrTT}} := c^{\text{CurrTT}} + \pi^{\text{unc}} \)
    End If
    Update LeftLines := LeftLines \( \setminus \{l\} \)
  End While
  If \( c^{\text{CurrTT}} \leq c^{\text{MacroTT}} \) Then
    Compute (see Appendix B) the robust cost \( c^{\text{CurrTT}} \) of timetable CurrTT
    If \( c^{\text{MacroTT}} + c^{\text{CurrTT}} < c^{\text{MacroHeu}} \) Then
      Set MacroTT := CurrTT and \( c^{\text{MacroTT}} := c^{\text{MacroTT}} + c^{\text{CurrTT}} \)
    End If
  End If
End For

(d) the cost for missing connections. Constraints (6) impose on the model that each train is either cancelled or scheduled. Constraints (7) ensure that if one or both trains corresponding to a connection are cancelled, then a penalty for missing the connection is paid. Constraints (8) state that if both trains of a connection are scheduled, then the difference between the corresponding departure and arrival times at the station where the connection takes place must be not less than \( \delta_x \). Constraints (9) trigger the penalty for exceeding the nominal connection time for each connection having both trains scheduled. The term \( M \chi_q \) in constraints (8) and (9) is used to prevent counting the penalty of the exceeding connection time when the connection is missed. Constraints (10) are clique constraints that impose on the provided timetable to be conflict-free (i.e., no headway and capacity constraints are violated); notice that constraints (10) can also be used to model simultaneous arrival or departures of pairs of trains at given stations, if these have to be considered as hard constraints. Constraints (11)–(14) set the domains of the variables of the model.

5.2. The macroscopic heuristic

Algorithm MacroHeu is a randomized multi-start greedy heuristic that computes a number of heuristic solutions for formulation (5)–(14) and returns the “best” one found to the microscopic module. A step-by-step description of MacroHeu is provided in Algorithm 3.

The initialization phase of MacroHeu consists of setting MacroTT equal to the empty set and its cost \( c^{\text{MacroTT}} \) equal to 0, where MacroTT is the subset of macroscopically conflict-free paths (at most one for each line) that is returned as output of the procedure. Then an iterative procedure starts running \( \text{iter} \) times, where \( \text{iter} \) is a parameter that is set equal to 1000 in the computational evaluation reported in Section 7. At each of these \( \text{iter} \) iteration, CurrTT represents the incumbent solution (i.e., it is a subset of conflict-free paths), \( c^{\text{CurrTT}} \) is the cost of timetable CurrTT, and LeftLines is the subset of lines that have to be processed in the current iteration \( \text{iter} \). Each iteration \( \text{iter} \) consists of two main steps: first, an attempt to find a macroscopically feasible timetable is made by iteratively selecting a line and running a procedure that finds the least-cost feasible path compatible with the paths of the set CurrTT (where the cost of such a path does not consider only the cost of the path itself but also the cost deriving from penalties related to connections); second, timetable CurrTT is assessed in terms of robustness, compared with the best timetable MacroTT found so far, and the best timetable MacroTT is possibly updated accordingly. Therefore, at each iteration, algorithm MacroHeu makes an attempt to finding a macroscopic feasible timetable by iteratively fixing variables \( x_p \) to 1 (and variables \( \xi_i \) if no feasible path is found for line \( l \in L \)), while increasing the cost of timetable CurrTT as little as possible.

Two crucial sub-routines are used in each iteration of MacroHeu. The first sub-routine finds the least-cost (with respect to the current timetable CurrTT) path for a line \( l \in L \). The second sub-routine assesses the robust cost \( c^{\text{CurrTT}} \) of timetable CurrTT. These two sub-routines are described in Appendices A and B, respectively.

6. Constraint updating

When running our micro–macro framework, we sequentially adapt process times at micro level and return to the macroscopic model to re-compute the MacroTT again. In fact, process times of the micro model represent and define constraints for the macroscopic tabling model. If MacroTT has conflicts at the microscopic level, then macro constraints on train
process times are recomputed by the microscopic model according to two main processes, namely: 1) constraints tightening and 2) constraints relaxation. In this section we describe these procedures in detail.

6.1. Constraints tightening

Once the macroscopic timetable is obtained, the microscopic model runs the conflict detection algorithm that tests the feasibility of the produced timetable, i.e., the potential existence of conflicts. If a conflict is observed, then the time separation between two trains has to be increased to satisfy safety constraints. By doing so, the degree of freedom for scheduling a train path (i.e., its search space) gets smaller, since constraints on minimum headway times between two train paths are introduced. We call the procedure of increasing the constraints on minimum headway times constraint tightening.

For each corridor $\varphi \in \Phi$, the existing conflict for a train pair $i, j \in T$, $c_{ij} > 0$, is resolved by updating the headway time as:

$$ h_{ij} \leftarrow h_{ij} + c_{ij}, $$

where $s$ is a timetable point on corridor $\varphi$. Note that the process for detecting a conflict on a corridor and identifying the corresponding station headway to solve that conflict, is not trivial. Once a track conflict is detected, the algorithm identifies the location of the closest timetable point $s$ to the conflict, and computes the minimum headway time which solves that conflict. In addition, the minimum time separation to avoid conflicts between two trains depends on whether the trains run in the same or in the opposite direction, so we determine the corresponding interaction between conflicting trains that may be arrival-arrival or departure-departure for trains running in the same direction, as well as arrival-departure or departure-arrival for trains running in opposite directions. Consequently, the minimum headway time $h_{ij}$ can be correctly updated.

We here give an example for updating the minimum headway time between two trains is updated if they were detected to be conflicting. Trains $t_1$ and $t_2$ run in the same direction along corridor $\varphi$, from timetable point $s_1$ to timetable point $s_2$ and have a track conflict with overlap time $c_{t_1,t_2}\varphi$. If the conflict is geographically closer to location $s_1$ then we update the headway between the departures of our trains at location $s_1$, $h_{t_1,t_2}\varphi$. Alternatively, if the conflict is closer to location $s_2$ then we update the headway between the arrivals of our trains at $s_2$, $h_{t_1,t_2}^{\text{ad}}$. The minimum time headway $h_{ij}$ is so increased by $c_{ij}$ and thereby tightening the relative constraint in the macroscopic model.

6.2. Constraints relaxation

A timetable point (e.g. station, junction) in a railway network is considered to be a potential bottleneck if the corresponding infrastructure occupation rate exceeds the thresholds recommended by the UIC on capacity usage. In this case, enlarging the time separation between trains at that timetable point is necessary to reduce the infrastructure occupation rate and introduce additional buffer times beneficial for the mitigation of delay propagation. We propose two types of constraint relaxations to reduce unacceptable occupation rate: 1) train homogenization, and 2) journey time extension.

First, it is commonly known that homogenized traffic consumes the least infrastructure capacity (Hansen and Pachl, 2014). Driven by this logic, we can guide trains to have more unified behaviour, i.e., more similar macroscopic running times. This can be obtained twofold: first, by allowing fast (intercity) trains to run slower, and second, by increasing the operational speed of slow (regional) trains through the bottleneck area.

This relaxation procedure is implemented as follows. Parameter $\nu$ is the threshold recommended by the UIC for a good usage of infrastructure capacity, while $\Delta$ is the percentage increment in running time supplement in the timetable (e.g., +0.5%). If the infrastructure occupation rate for corridor $\varphi$ between two stations is $\mu(\varphi) > \nu$, then: i) the maximum running time supplement for fast trains $\tau_F$ on corridor $\varphi$ will be increased by $\Delta_1$ as in (15) and ii) the nominal running time supplement for slow trains $\tau_S$ on the same corridor will be decreased by $\Delta_2$ as in (16).

$$ \lambda_{\varphi}^{\text{max}}(\tau_F) \leftarrow (1 + \Delta_1)\lambda_{\varphi}^{\text{max}}(\tau_F) \quad (15) $$

$$ \lambda_{\varphi}^{\text{nom}}(\tau_S) \leftarrow (1 - \Delta_2)\lambda_{\varphi}^{\text{min}}(\tau_S) \quad (16) $$

As a further measure to reduce infrastructure occupation, we also increase the maximum allowed journey time of fast trains. Note that this relaxation goes in line with (15) as a necessary correction. For example, let us assume a scheduled fast train that has assigned the maximum journey time and runs on a corridor where the total capacity is higher than $\mu(\varphi)$. If we increase the maximum running time for one train only locally $\lambda_{\varphi}^{\text{max}}$, while maximum journey time $\bar{\tau}_F$ stays unchanged, we might not experience the benefit of the $\lambda_{\varphi}^{\text{max}}$ relaxation because $\bar{\tau}_F$ will be still the bounding constraint. Therefore, it is important to relax both $\lambda_{\varphi}^{\text{max}}$ and $\bar{\tau}_F$.

7. Computational experiments

This section considers the computed timetable for the Dutch case study, including the computational results, the achieved values for the performance measures and plots illustrating the timetable and its performance measures. We have tested our
iterative approach for timetable planning on a railway corridor in the Netherlands. The models and the framework are developed in Matlab and C++. Tests are made by using a dual core Intel E7 with 2.6 GHz processor and 8GB RAM. The microscopic and macroscopic modules used only one processor core. As input to construct the timetable, we considered the Dutch train service specification for the year 2012.

7.1. Case study

We have designed the timetable for a relevant part of the Dutch network which includes the corridors between the stations of Utrecht (Ut) and Eindhoven (Ehv), ’s Hertogenbosch (Ht) and Tilburg (Tb), and ’s Hertogenbosch and Nijmegen (Nm). Fig. 2a illustrates the geographical representation of the test case. The network in the microscopic model consists of 1500 homogenous behavioral sections, 950 block sections and 28 microscopic timetable points (e.g., stations, stops, junctions, bridges) of which five are IC stations: Ut, Ht, Ehv, Nm and Tb. Most of the corridors are double-track.

The timetable on this network is periodic with an hourly pattern composed of eight InterCity (IC) and 12 regional train lines all with two services per hour. So, a total of 40 train runs per hour are scheduled over the whole area. The corresponding line plan is given in Fig. 2b, where each color represents a single train line. In particular, two IC lines serve the corridor Ehv-Ht-Ut, one serves Tb-Ht-Nm and one Ehv-Tb. Regional train lines operate on the corridors: Tiel (Tl)-Geldermalsen (Gdm)-Ut, Tb-Ht, Ht-Gdm-Ut, Ht-Nm, Ehv-Ht and Ehv-Tb. This line plan results in 16 operating trains between Ut and Gdm, 4 between Gdm and Tl, 12 between Gdm and Ht, 8 between Ht and Nm, as well as Ht and Tb, and 20 between Ht and Ehv.

Table 1 gives the timetable design norms that have been used as an input for the timetabling process. These values are provided by the railway planners.

The developed framework computes the necessary process times and sets up the macroscopic network by applying Algorithm 1. The macroscopic model is built by aggregating the microscopic infrastructure model. Specifically, this process has aggregated the 28 microscopic points into 15 timetable points in the macroscopic model with 15 corresponding arcs between them. The macroscopic infrastructure model is illustrated in Fig. 3. Most of the lines are double-track unless the given number suggests differently. For example, the corridor between Boxtel (Btl) and Ehv consists of a four-track line. Note that not all microscopic timetable points are considered in the macroscopic model as explained in Section 3.1. In total, the function AggregateProcessTimes produces 76 macroscopic running times for any pair of consecutive timetable points for all of the 20 train lines, and 2027 minimum headway time computations. The computation time depends on the size of the network and the number of train runs. For the considered case study, the execution of Algorithm 1 took under 30 s to generate the macroscopic network model and compute the aggregated process times. The average computation time per iteration

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timetable design norms.</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Maximum journey time extension</td>
</tr>
<tr>
<td>Minimum running time supplement</td>
</tr>
<tr>
<td>Maximum running time supplement</td>
</tr>
<tr>
<td>Dwell time at stops</td>
</tr>
<tr>
<td>Dwell time at macro points</td>
</tr>
<tr>
<td>Minimum transfer times</td>
</tr>
</tbody>
</table>

Fig. 2. a) Dutch railway network with highlighted case study area and b) train line plan (case study). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
is about 40 s for the microscopic model, and about 80 s for the macroscopic one (with \( \text{iter} = 1000 \) macroscopic iterations). Thus, the average time per micro–macro iteration was in total 120 s.

Fig. 4 shows the computational results of the micro–macro iterations for obtaining a conflict-free, robust and stable timetable. After nine iterations the algorithm converged to a feasible solution which is both microscopically conflict-free and stable and macroscopically optimized. During the iterations a decreasing trend can be observed for the number of conflicts (blue solid line) and the total overlap time of conflicting blocking times (green dashed line), with some iterations leading to an increased number of conflicts and overlap time when the timetable structure (train orders) changes significantly from one iteration to the next in face of new minimum headway times provided to resolve the conflicts. The total computation time, from the microscopic input computation to the produced conflict-free, stable and robust timetable was about 20 min.

Fig. 5 shows the time-distance diagram, while the associated blocking time diagram is given in Fig. 6. The vertical axis shows time in minutes downwards. The horizontal axis shows distance with the station positions indicated. The blue lines are IC trains, while magenta lines are local trains. Note that Fig. 6 considers only the infrastructure route used by the intercity train line 3500, running form Ut to Ehv. This means that blocking times of other trains will be represented in the picture only if these trains traverse block sections which are on the route of train line 3500.

The optimized timetable shows periodic passenger trains with regular 15 min services of both IC and local trains where two similar train lines follow the same route. Hence, effectively 15 min train services are realized instead of two separate
30 min train lines. On the main corridor Utrecht–Eindhoven, the ICs overtake local trains at Geldermalsen (Gdm), but this overtaking does not take place for trains in the opposite direction.

Table 2 shows the infrastructure occupation on the main corridors. All the infrastructure occupation rates are below the threshold recommended by the UIC of 75% defined for mixed traffic corridors at peak hours. The two heaviest used corridors are Utrecht – ’s-Hertogenbosch in both directions, with a maximum infrastructure occupation rates of 54.7% for one direction and 53.4% for the opposite direction. The other corridors have an infrastructure occupation rate below 41%.

7.2. Additional computational analyses

To demonstrate the applicability of the micro–macro timetabling model, we utilized additional computations. We randomly generated line plans for a different number of train lines in the train service specification that ranged from 16 to 25. Table 3 shows the computational results for all given scenarios. For every scenario it is reported the number of train lines in the line plan, the number of macroscopic running arcs and headway arcs, the number of micro–macro iterations, the infrastructure occupation rate of the most used corridor, the average time supplements allocated in the corresponding timetable, the time for microscopic to macroscopic conversions in the initialization, and the total computation times. The scenario previously analyzed in Section 7.1 is reported as scenario basic while the other scenarios were randomly generated from the set of lines of the basic scenario by varying the chosen lines.

In general, the number of micro–macro iterations grows when the number of requested train lines increases. The timetabling model needed at least three iterations to find a solution (sc5) when the number of train lines was 16. On the other hand, at least 10 iterations were needed to obtain the conflict-free solutions for scenarios with more than 20
Table 3
Computational results for all scenarios.

| Scenario # lines # trains # macro running arcs # headway arcs # iterations Max infra occupation rate (%) Average time supplements (%) Initialization (s) Computation time (s) |
|-----------------------------------------|-----------------|-------------------|----------------|----------------|----------------|-----------------|-----------------|-----------------|-----------------|
| Basic                                   | 20              | 40                | 76             | 1712           | 9              | 53.44           | 8.80            | 29              | 1054            |
| sc1                                     | 16              | 32                | 62             | 1304           | 6              | 42.46           | 8.79            | 23              | 611             |
| sc2                                     | 17              | 32                | 56             | 966            | 3              | 46.89           | 8.78            | 23              | 353             |
| sc3                                     | 22              | 44                | 88             | 1423           | 7              | 57.94           | 8.89            | 24              | 719             |
| sc4                                     | 23              | 44                | 60             | 1049           | 5              | 46.89           | 8.88            | 24              | 539             |
| sc5                                     | 18              | 36                | 70             | 1564           | 8              | 53.44           | 9.28            | 26              | 814             |
| sc6                                     | 18              | 36                | 64             | 1154           | 4              | 51.33           | 9.27            | 26              | 502             |
| sc7                                     | 19              | 38                | 70             | 1414           | 4              | 52.56           | 8.87            | 28              | 514             |
| sc8                                     | 21              | 42                | 82             | 2030           | 11             | 59.06           | 8.93            | 30              | 1165            |
| sc9                                     | 21              | 42                | 79             | 1846           | 5              | 57.94           | 8.84            | 30              | 640             |
| sc10                                    | 22              | 44                | 88             | 2386           | 9              | 63.67           | 8.70            | 32              | 1089            |
| sc11                                    | 22              | 44                | 82             | 2000           | 10             | 57.94           | 8.59            | 32              | 1207            |
| sc12                                    | 22              | 44                | 84             | 2060           | 10             | 57.94           | 9.16            | 34              | 1304            |
| sc13                                    | 23              | 46                | 91             | 2540           | 14             | 60.67           | 9.04            | 34              | 1795            |
| sc14                                    | 23              | 46                | 85             | 2095           | 13             | 57.94           | 8.53            | 34              | 1694            |
| sc15                                    | 24              | 48                | 90             | 2264           | 14             | 57.94           | 8.94            | 36              | 1809            |
| sc16                                    | 24              | 48                | 90             | 2384           | 11             | 57.94           | 9.29            | 36              | 1456            |
| sc17                                    | 25              | 50                | 96             | 2758           | 15             | 64.39           | 9.35            | 37              | 1885            |

Fig. 7. Time-distance diagram corridor Utrecht – Eindhoven for scenario sc17.

This could be expected as the larger the number of train lines, the less is the freedom to schedule trains on the given infrastructure. Consequently, the computation times varied between 353 and 1885 s.

The number of macroscopic running arcs also depends on the number of train lines in the train service specification and we observed this number varied in the range from 56 to 96. A similar situation is observed for the number of headways which ranged from 996 to 2758 and the initialization time which varied between 23 and 37 s. The average computation time for calculating minimum and operational running times for a single train line was 1 s and 5 s, respectively. The average time supplement for all scenarios varied between 8.78% and 9.27%.

We also observed that the infrastructure occupation rate does not explicitly grow when increasing the number of train lines. Thus, changing the number of trains does not necessarily mean a change in infrastructure occupation. For example, this was observed between scenarios basic and sc5, where the maximum infrastructure occupation rate remained the same although the number of train lines was different, 20 and 18, respectively. Thus, we may say that the infrastructure occupation rate does not depend only on the number of train lines, but also on the characteristics of the line plan such as the type of line service (heterogeneity), the origin and destination stations, the line routes and the scheduled connections.

Fig. 7 gives the time-distance diagram for the scenario with the biggest line plan sc17 which included 25 train lines.

8. Conclusions

This paper presented an integrated automatic timetable planning framework that produces timetables that are microscopically feasible, stable and robust. The developed approach incorporated the strengths and advantages of macroscopic and macroscopic algorithms to provide overall efficient and satisfactory solution. Network transformation algorithms were introduced to automatically convert data from the microscopic level to macroscopic one and vice versa. The macroscopic model is used for computing a robust network timetable which is afterwards converted and thoroughly analyzed at the microscopic level. The analysis includes conflict detection and capacity assessment. If track conflicts are detected and/or
capacity norms are violated, necessary adjustments to train process times were undertaken by applying a procedure of constraints tightening and relaxation. This iterative micro–macro process automatically terminates once the timetable is also microscopically feasible and stable.

A practical application to an area of the Dutch railway network showed the ability of this framework in ensuring the feasibility of the macroscopic timetables at the level of track detection sections. A high quality timetable was produced in 3 to 15 iterations, depending on the given number of train lines plan, while the computing times were between 353 and 1885 s. In addition, the UIC norms on infrastructure occupation rates were satisfied, so that for all the scenarios we obtain a maximum occupation rate below 65%.

The proposed framework and integrated models are suitable for developing both periodic and non-periodic timetables. Practitioners and timetable designers can use this framework for timetable design and for the evaluation of existing timetables. Future research will be addressed to generate and evaluate timetables that also include scheduling of short-term freight train paths. Also a specific study will be dedicated to define performance measures that evaluate the resilience of the timetable, i.e., the ability of restoring scheduled operations when real-time rescheduling is applied during perturbed traffic. The use of the proposed micro–macro approach is indicated for practitioners as a tool for generating timetables that are operationally feasible and robust to daily perturbations of the scheduled train operations.

Acknowledgments

This research is funded by the European FP7 project Optimal Networks for Train Integration Management across Europe (ON-TIME) under Grant Agreement number FP7-SCP1-GA-2011-285243. We thank the Dutch infrastructure manager ProRail and the Netherlands Railways (NS) for the data provided.

Appendix A. Finding the least-cost path for a given line and timetable

The sub-routine for finding the least-cost path for a given line and a given partial timetable works as follows. The insertion of a line into the incumbent timetable corresponds to fixing a variable \( x \) into the model (5)–(14) and is performed by running an exact dynamic programming recursion, a simplified version of the one used in Cacchiani et al. (2010), that identifies a feasible min-cost path that is compatible with the paths assigned to the previously processed trains/lines, where the cost takes into account not only the cost of the paths but also the cost for connections. Such a dynamic programming recursion computes functions \( f(t, \sigma, e) \) with three state-variables, where \( t \) is the instant of time at which the event \( e \) (be it an arrival, a departure, or a pass at a given station along the route of the train) takes places, and \( \sigma \) represents the total stretch of the path (computed as the running and dwell time excess over the nominal ones).

This sub-routine receives as input the index \( l \) of a line of the set \( \text{LeftLines} \) that must be scheduled, and a timetable \( \text{CurrTT} \) that contains at most a path for each of the lines of the set \( \text{LeftLines} \). The output of this sub-routine is either a path of the set \( P_l \) that minimizes the increase in the cost of timetable \( \text{CurrTT} \) while maintaining its feasibility, or the indication that all of the paths of the set \( P_l \) are incompatible with the paths of the set \( \text{CurrTT} \), which means that at least one of the constraints (10) is violated if such a path of line \( l \) is added to \( \text{CurrTT} \).

The sub-routine is a dynamic programming recursion that computes functions \( f(t, \sigma, e) \), for each instant of time \( t = 0, \ldots, \bar{t} \) (where \( \bar{t} \) is the length of the planning horizon over which the timetabling computation is performed), each possible total stretch \( \sigma = 0, \ldots, \bar{\sigma}_l \) with respect to nominal running and dwell times of line \( l \), and each event \( e \) (being it a departure and/or an arrival to each of the timetable points belonging to the route of line \( l \)). Function \( f(t, \sigma, e) \) represents the cost of the min-cost partial path from the departure from station \( s_l \) of \( S_l \) up to event \( e \) among all partial paths scheduling event \( e \) at time \( t \) with a total stretch equal to \( \sigma \).

Functions \( f(t, \sigma, e) \) are computed according to the following recursive propagation rules:

- If event \( e = \text{dep}_i \) corresponds to the departure from timetable point \( s_l \in S_l, i = 1, \ldots, |S_l| - 1 \), then
  \[
  f(t, \sigma, \text{dep}_i) = \min_{t = \tilde{t} + t - w_{l, s_l}} \left\{ f\left(t' \% \bar{t}, \max\left\{ \sigma + t' - \tilde{t} - t + w_{l, s_l}, 0\right\}, \text{arr}_i\right) + \pi^\text{dwell}_{l, s_l} \left(\max\left\{t' - \tilde{t} - t + w_{l, s_l}, 0\right\}\right)\right\},
  \]
  
  (17)

  for each instant of time \( t = 0, \ldots, \bar{t} \) and each possible stretch \( \sigma = 0, \ldots, \bar{\sigma}_l \);

- If event \( e = \text{arr}_i \) corresponds to the arrival at timetable point \( S_l \in S_l, i = 2, \ldots, |S_l| \), then
  \[
  f(t, \sigma, \text{arr}_i) = \min_{t = \tilde{t} + t - r_{l, s_l}} \left\{ f\left(t' \% \bar{t}, \max\left\{ \sigma + t' - \tilde{t} - t + r_{l, (s_l-1)}, 0\right\}, \text{dep}_{i-1}\right) + \pi^\text{arr}_{l, s_l} \left(\max\left\{t' - \tilde{t} - t + r_{l, (s_l-1)}, 0\right\}\right)\right\}
  \]
  
  (18)

  for each instant of time \( t = 0, \ldots, \bar{t} \) and each possible stretch \( \sigma = 0, \ldots, \bar{\sigma}_l \).

  Here, the term \( \% \) represents the modulo operation, that is the remainder of the Euclidean division, \( t' \% \bar{t} = t' - \bar{t} \cdot t'/\bar{t} \).

  To explain the previous recursive propagation rules, consider the following examples:
Given $i = 100$, $t = 50$, and $\sigma = 10$, and considering the departure from station $i$ (i.e., considering event $dep_i$), we want to compute functions $f(50, 10, dep_i)$, knowing that the nominal dwell time $w_{i,s_i}$ is 5 and the maximum dwell time $\bar{w}_{i,s_i}$ is 10. All functions $f(50, 10, dep_i)$ recursively originate from functions $f(t^\prime, \sigma^\prime, arr_i)$, where $t^\prime$ is in between $40 = (\bar{t} + t - w_{i,s_i})\%\bar{t}$ and $45 = (\bar{t} + t - w_{i,s_i})\%\bar{t}$ and $\sigma^\prime = 10$ when $t^\prime = 45$, $\sigma^\prime = 9$ when $t^\prime = 44$, ..., $\sigma^\prime = 5$ when $t^\prime = 40$, where $\sigma^\prime = t - \bar{t} + \bar{w}_{i,s_i}$.

Given $i = 100$, $t = 50$, and $\sigma = 10$, and considering the arrival at station $i$ (i.e., considering event $arr_i$), we want to compute functions $f(50, 10, arr_i)$, knowing that the nominal running time $r_{i,(s_{i-1}, s_i)}$ from station $s_{i-1}$ to station $s_i$ is 5 and the maximum running time $\bar{r}_{i,(s_{i-1}, s_i)}$ from station $s_{i-1}$ to station $s_i$ is 10. All functions $f(50, 10, arr_i)$ recursively originate from functions $f(t^\prime, \sigma^\prime, dep_{i-1})$, where $t^\prime$ is in between $40 = (\bar{t} + t - \bar{r}_{i,(s_{i-1}, s_i)})\%\bar{t}$ and $45 = (\bar{t} + t - \bar{r}_{i,(s_{i-1}, s_i)})\%\bar{t}$ and $\sigma^\prime = 10$ when $t^\prime = 45$, $\sigma^\prime = 9$ when $t^\prime = 44$, ..., $\sigma^\prime = 5$ when $t^\prime = 40$, where $\sigma^\prime = t - \bar{t} + \bar{r}_{i,(s_{i-1}, s_i)}$.

In order to compute functions $f(t, \sigma, e)$ recursively, the following initialization is required:

$$f(t, 0, dep_{s_i}) = 0$$

for each instant of time $t = 0, \ldots, \bar{t}$, which means that the cost for departing from the first station of line $l$ with no stretch is 0 no matter the departing time, and

$$f(t, \sigma, dep_{s_i}) = \infty$$

for each instant of time $t = 0, \ldots, \bar{t}$ and each possible stretch $\sigma = 1, \ldots, \bar{\sigma}_i$.

Moreover, whenever event $e$ cannot take place at time $t$ because this corresponding path when added to $CurrTT$ would violate some constraints (e.g., headway times, capacity constraints, connections involving lines already scheduled in $CurrTT$, etc.), we set $f(t, s, e) = \infty$ for each possible stretch $\sigma = 0, \ldots, \bar{\sigma}_i$.

It is clear that, by computing functions $f(t, \sigma, e)$ as described above, the only penalties that are taken into account are the ones for exceeding nominal dwell times (i.e., $\pi^d_{dwell}$) and for exceeding nominal running times (i.e., $\pi^d_{run}$). So far, penalties related to connections (i.e., $\pi^c_{conn}$ and $\pi^c_{conn}$) have not been considered. Nonetheless, it is easy to observe that, given the subset of paths of the set $CurrTT$, the penalties that must be paid if a path for line $l$ is selected depend uniquely on the time each of the events of line $l$ take place. This means that before computing functions $f(t, \sigma, e)$, the penalty $\pi^c_{conn}(t, e)$ for adding to $CurrTT$ a path of line $l$ where event $e$ takes place at time $t$ can be computed.

Penalties $\pi^c_{conn}(t, e)$, for each instant of time $t = 0, \ldots, \bar{t}$ and each event $e$ are computed as follows:

- If event $e = dep_i$ corresponds to the departure from timetabling point $s_i \in S_i$, $i = 1, \ldots, |S_i| - 1$, then

$$\pi^c_{conn}(t, dep_i) = \begin{cases} 
\infty & \text{if } \exists \ell' \in L \setminus \text{LeftLines} \land \exists q = (l', l, s_i) \in Q : \\
\sum_{q = (l', l, s_i) \in Q} \pi^c_{conn}(t - \pi^d_{p(l')s_i}) & \text{otherwise,}
\end{cases}$$

where $\pi^d_{p(l')s_i}$ indicates the arrival time of line $l'$ at timetabling point $s_i$ in path $p(l')$, which is the path assigned to line $l'$ in $CurrTT$. This means that, whenever there exists a line $l'$ that has already been scheduled (i.e., $l' \in L \setminus \text{LeftLines}$) and a connection between the arrival of line $l'$ and the departure of line $l$ from timetabling point $s_i$ that cannot be met if the departure of line $l$ from $s_i$ is scheduled at time $t$, then any path of line $l$ scheduling such a departure at time $t$ is infeasible; otherwise (if such a line and such a connection do not exist), then the penalty for scheduling the departure of line $l$ from timetabling point $s_i$ at time $t$ is given by the sum of the differences between $t$ and the arrival time at $s_i$ of all lines $l'$ for which a connection $(l', l, s_i) \in Q$ exists;

- If event $e = arr_i$ corresponds to the departure from timetabling point $s_i \in S_i$, $i = 2, \ldots, |S_i|$, then

$$\pi^c_{conn}(t, arr_i) = \begin{cases} 
\infty & \text{if } \exists \ell' \in L \setminus \text{LeftLines} \land \exists q = (l, l', s_i) \in Q : \\
\sum_{q = (l, l', s_i) \in Q} \pi^c_{conn}(t - \pi^d_{p(l')s_i}) & \text{otherwise,}
\end{cases}$$

where $\pi^d_{p(l')s_i}$ indicates the departure time of line $l'$ at timetabling point $s_i$ in path $p(l')$, which is the path assigned to line $l'$ in $CurrTT$. This means that, whenever there exists a line $l'$ that has already been scheduled (i.e., $l' \in L \setminus \text{LeftLines}$) and a connection between the departure of line $l'$ and the arrival of line $l$ at timetabling point $s_i$ that cannot be met if the arrival of line $l$ at $s_i$ is scheduled at time $t$, then any path of line $l$ scheduling such an arrival at time $t$ is infeasible; otherwise (if such a line and such a connection do not exist), then the penalty for scheduling the arrival of line $l$ at timetabling point $s_i$ at time $t$ is given by the sum of the differences between the departure time from $s_i$ of all lines $l'$ for which a connection $(l, l', s_i) \in Q$ exists and $t$. 
Therefore, in order to keep account of penalties related to connections, the recursive Eqs. (17) and (18) have to be modified as follows:
\[
f(t, \sigma, dep_t) = \min_{t' = t + \text{w}_{i,j} \ldots t + \text{w}_{i,j}} \left\{ f(t' \% \tilde{t}, \max \{\sigma + t' - \tilde{t} - t + \text{w}_{i,j}, 0\}, \text{arr}_t) \right. \\
\left. + n^\text{dwell}_i (\max \{t' - \tilde{t} - t + \text{w}_{i,j}, 0\}) \right\} + f \pi^\text{conf} (t, \text{dep}_t)
\]

and
\[
f(t, \sigma, \text{arr}_t) = \min_{t' = t + \text{r}_{1, (i, j)} \ldots t + \text{r}_{1, (i, j)}} \left\{ f(t' \% \tilde{t}, \max \{s + t' - \tilde{t} - t + \text{r}_{1, (i, j)}, 0\}, \text{dep}_{t-1}) \right. \\
\left. + n^\text{run}_i \left(\max \{t' - \tilde{t} - t + \text{r}_{1, (i, j)}, 0\}\right) \right\} + \pi^\text{conf} (t, \text{arr}_t).
\]

Then, the path for line \( l \) that implies the minimum increase when added to the set \( \text{CurrTT} \), corresponds to the one generating the function \( f(t^*, \sigma^*, \text{arr}_{\text{CurrTT}}) \), where
\[
(t^*, \sigma^*) = \arg \min_{t = 0, \ldots, l \sigma = 1, \ldots, \tilde{t}_l} \{ f(t, \sigma, \text{arr}_{\text{curr}}) \}.
\]

Appendix B. Robustness assessment of a given timetable

This procedure is aimed at assessing the robustness of a given timetable. The input of the procedure is timetable \( \text{CurrTT} \) generated in a given iteration of MacroHeu, and the output is its robust cost \( c^{\text{CurrTT}} \), which is defined in the following. The main idea about assessing the robustness of the timetable is to generate a number of different scenarios, each one characterized by a random delay for each train, and run on each of these scenarios a local search procedure that tries to eliminate conflicts by retiming trains. The robust cost \( c^{\text{CurrTT}} \) is then determined by the weighted sum of the unresolved conflicts plus the time to absorb the delays on each of the scenarios considered. A step-by-step description of the robustness assessment procedure is provided in Algorithm 4.

Step 1 consists of an initialization phase. Timetable \( \text{TTDelay} \) is the timetable that will be used in the following steps. It is the same as \( \text{CurrTT} \) except that the trains of each line are replicated. This means that, for example, if in \( \text{CurrTT} \) there is a line with periodicity 30 min that is scheduled to depart from station \( s \) at 10:00, then in \( \text{TTDelay} \) such a line corresponds to \( n \) trains (where \( n \) is a parameter, e.g., \( n = 4 \)); the first one scheduled to depart from \( s \) at 10:00, the second at 10:30, the third at 11:00, and the fourth at 11:30. In other words, from the periodic timetable \( \text{CurrTT} \) the corresponding non-periodic timetable \( \text{TTDelay} \) is generated.

Step 2 consists of the generation of the delays for each train, in particular, delays are generated for the first train of each line of the non-periodic timetable \( \text{TTDelay} \) only. The delays are generated according to a standard normal distribution and normalized over the periodicity of the timetable. In particular, for each train \( t \) a random value \( \text{rand} \) is generated and an event \( e \) (i.e., a departure or an arrival at one of the timetable points traversed) is randomly selected. The time of event \( e \) in \( \text{TTDelay} \) is then postponed by \( \text{rand} \times \tilde{\theta}/3 \) units of time, where \( \tilde{\theta} \) is the maximum delay considered. Function \( \text{rand} \) generates a number randomly based on a truncated Normal distribution \( N(0, 1) \) that allows only positive values that are not greater than three.

Having generated a delay for the first train of each line, the resulting timetable \( \text{TTDelay} \) may no longer be feasible from a macroscopic point of view: there may be headway times, capacity constraints, and/or connection times violated. In order to recover such a feasibility, all conflicts are iteratively processed, one at a time, by considering first the conflict occurring the last in time (i.e., conflicts are resolved starting from the latest ones in the time horizon), and then causing the conflict are retimed to resolve the conflict. In particular, the retiming operation consists of simply increasing the dwell times and/or the running times of the trains involved in the conflict; no rerouting nor rerouting of the trains are allowed. The procedure ends as soon as all conflicts are resolved or the resolution of the conflicts would imply that more conflicts would be generated.

Step 4 consists of updating the robust cost of timetable \( \text{CurrTT} \). Let \( \gamma \) be the number of unresolved conflicts in \( \text{TTDelay} \) (i.e., the number of unsatisfied connections, headway times, capacity constraints, etc.). Moreover, let \( \hat{D}_p(t) \in \mathbb{R} \) be

\begin{algorithm}

\begin{itemize}
  \item **Input:** macroscopic timetable \( \text{CurrTT} \) of cost \( c^{\text{CurrTT}} \)
  \item **Output:** robust cost \( c^{\text{robust}} \) of timetable \( \text{TTDelay} \)
  \item 0. Initialize \( c^{\text{CurrTT}} := 0 \)
  \item For \( n \text{scenario} = 1, \ldots, n \text{scenario} \) Do
    \item 1. Initialize \( \text{TTDelay} \) by starting from \( \text{CurrTT} \)
    \item 2. Generate a random delay for each train of timetable \( \text{TTDelay} \)
    \item 3. Run a local search procedure that retimes trains of \( \text{TTDelay} \) to resolve conflicts
    \item 4. Update \( c^{\text{robust}} \)
  \item End For
\end{itemize}

\end{algorithm}
the departure time from timetable point \( s_i \in S_i, \ i = 1, \ldots, |S_i| - 1 \), in the path \( p(t) \) selected for any of the trains \( t \) of line \( l \in L \), and let \( \pi_{p(t)_{s_i}}^A \) be the arrival time at timetable point \( s_i \in S_i, \ i = 2, \ldots, |S_i| \), in the path \( p(t) \) selected for any of the trains \( t \) of line \( l \in L \) in timetable \( TTDelay \) obtained after Step 3. The robust cost \( C^{CurrTT} \) is updated by adding

\[
\pi^{\text{uns}} \sum_{l \in L} \sum_{t \in T(t)} \sum_{s \in S} (\hat{\pi}^D_{p(t)_s} - \pi_{p(t)_s}^D + \hat{\pi}^A_{p(t)_s} - \pi_{p(t)_s}^A)
\]

where \( \pi^{\text{uns}} \) and \( \pi^{\text{delay}} \) are parameters representing the penalties for each unsolved conflict and each unit of delay, respectively.

References


