A cycle time optimization model for generating stable periodic railway timetables

Sparing, Daniel; Goverde, Rob M.P.

DOI
10.1016/j.trb.2016.12.020

Publication date
2017

Document Version
Final published version

Published in
Transportation Research Part B: Methodological

Transportation Research Part B 98 (2017) 198–223

Contents lists available at ScienceDirect

Transportation Research Part B

journal homepage: www.elsevier.com/locate/trb

A cycle time optimization model for generating stable periodic railway timetables[☆]

Daniel Sparing, Rob M.P. Goverde[*]

Department of Transport and Planning, Delft University of Technology, P.O. Box 5048, 2600 GA Delft, The Netherlands

ARTICLE INFO

Article history:
Received 31 July 2016
Accepted 31 December 2016
Available online 12 January 2017

Keywords:
Railway timetabling
Robustness
Stability

ABSTRACT

As train passengers expect a high degree of reliability from a railway network with minimal delays, during the timetabling process planners need to balance the goals of maximizing the offered capacity and delay resistance. This is often done in a two-step process where first a feasible timetable is found for a given line structure, and consecutively the stability of this timetable is evaluated and local modifications are performed to the timetable. This paper describes an optimization method to find a feasible periodic timetable that also ensures maximum stability for heterogeneous railway networks. The model is capable to handle flexible train orders, running and dwell times, and overtaking locations. We use the minimum cycle time of the periodic timetable as an indicator for stability, and define an optimization problem with this minimum cycle time as the objective function to be minimized. We also present dimension reduction methods and an iterative optimization approach to improve the mathematical optimization process. We show the applicability of the approach with case studies on the central part of the Dutch railway network.

1. Introduction

In this paper we introduce a railway timetable optimization model that focuses on maximizing the stability of the timetable already during the optimization. A large number of approaches exist estimating railway network capacity, generating a robust timetable, and assessing timetable stability. In particular, existing robust timetable generation methods not only focus on the feasibility but also the stability and robustness of the timetable, either by ex post stability analysis or maximizing buffer times between trains during the optimization. Hardly any paper takes into account, however, the *network-level stability* during the optimization process. We therefore present a model to handle this shortcoming.

In this paper we adopt the definitions of stability and robustness from Goverde and Hansen (2013). Stability is the ability of a timetable to absorb delays so that delayed trains return to their scheduled train paths without rescheduling. A timetable that can return to the schedule faster from the same primary delays is considered more stable. Robustness is the ability of a timetable to withstand deviations from the scheduled time–distance paths due to design errors, parameter variations, and changing operational conditions. A timetable that generates less delays from disturbances in the process times is considered more robust. Hence, stability refers to sufficient amount of time allowances in the timetable, while robustness refers to the

☆ This research was funded by the research programme *Sustainable Accessibility of the Randstad* of the Netherlands Organisation for Scientific Research (NWO), Grant no. 434-09-035.

* Corresponding author.
E-mail address: r.m.p.goverde@tudelft.nl (R.M.P. Goverde).

http://dx.doi.org/10.1016/j.trb.2016.12.020
0191-2615/© 2017 Elsevier Ltd. All rights reserved.

allocation of the time allowances to process times of trains (time supplements) or between train paths (buffer times). In this paper, we focus on stability at the network level for periodic timetables. The stability of a periodic timetable for a railway network depends on the minimal cycle time (period length) that is possible given all precedence constraints in the event-activity network (or timed event graph). This minimum cycle time is defined by the critical circuit in the event-activity network which has the least mean time allowance, where a circuit is a closed path of events and activities (Goverde, 2007; Goverde, 2010). The minimum cycle time can be computed efficiently in max-plus algebra for a given timetable structure in which the event orders are fixed and is independent of the actual allocation of time allowances over the timetable associated with the scheduled cycle time (Goverde, 2007; Heidergott et al., 2005). However, changing the event orders may also affect the critical circuit and this leads to the important problem of finding a timetable structure with the optimal event orders such that the resulting minimum cycle time is minimized. This timetable structure is characterized as having the maximal mean time allowance on the critical circuit and is thus the most stable among all timetables satisfying all precedence constraints. This is the problem we will tackle in this paper.

We propose an innovative stability-focused periodic timetable optimization model and solution method for densely used railway networks based on an extension of the Periodic Event Scheduling Problem (PESP) (Serafini and Ukovich, 1989). This approach allows for a quick evaluation of whether a required train line pattern is feasible on given infrastructure, and provides an optimized timetable structure if possible. Such a model can be used for a variety of timetabling purposes: in the development of the actual railway timetable; in experimenting with new, innovative stop patterns of train lines and evaluating their feasibility; and the evaluation of infrastructure bottlenecks to identify which infrastructure improvement could yield the best results in increasing capacity.

The main innovation of this paper is the idea to directly use timetable stability as the main objective in the timetable optimization of heterogeneous train networks with flexible train running times and train orders. This approach was first introduced in Sparing and Goverde (2013) of which this paper is an extension. Stability of a periodic timetable is quantified in terms of the minimum cycle time of the timetable: if this minimum cycle time is lower than the nominal timetable period (such as one hour), then and only then the timetable is stable, and the bigger this gap is the more stable at the expense of unused capacity (Goverde, 2007). This key idea allows for the integration of previously separate timetable planning activities: the setting of train orders and the evaluation of timetable stability; therefore avoiding the need for a feedback loop of several iterations of planning.

This timetable optimization model is also highly flexible: while it does assume a fixed line pattern and routing, the run and dwell times are flexible and only constrained from above by sensible business rules of what is considered an acceptable running and dwell time supplement, and likewise train orders and overtaking locations are flexible. The railway infrastructure restrictions, however, are explicitly modeled by headway times separating trains and also overtaking limitations are taken into account. When defining overtaking constraints at stations, we introduce a new method to work only with headway constraints and dummy nodes, in order to avoid the definition of a large number of new train order constraints that would otherwise be necessary for flexible train order models. Our main contributions toward reducing the solution time of the optimization problem are twofold. First, we use a number of reduction techniques to reduce the problem definition, taking advantage of the symmetry of the periodic timetable and using so-called symmetry-breaking constraints. Second, we develop an iterative solution method using a flexible range of the cycle time that adaptively re-adjusts the cycle time range to ensure that intermediate solutions are found fast with feedback in the solution process to speed up the calculation time to the optimal solution.

The main contributions of this paper are the following:

- An optimization model has been developed that generates feasible periodic railway timetables with maximized network stability.
- The model finds optimal overtaking locations and train orders that minimizes the cycle time while respecting timetable and infrastructure constraints.
- The periodic event scheduling problem has been extended to a variable cycle time formulation to find periodic schedules with a minimum cycle time.
- An effective iterative solution method has been developed to solve large instances.

The macroscopic periodic timetable optimization model described in this paper focuses on high-frequency, high capacity-utilization networks. It takes as input the desired stop patterns and frequencies of train lines, as well as the structure of the railway network and the minimum process times, and generates a feasible timetable optimized for stability, if it exists. The running and dwell times of trains are flexible up to predefined boundaries, and the train orders are fully flexible. The overtaking locations are also flexible, given a priori within a predefined list of stations with enough capacity for overtaking. While the model is designed for periodic timetables, an aperiodic timetable can also be calculated using a single period. The model takes into account limited railway infrastructure by respecting predefined minimum headways between trains at stations and junctions. The capacity limits of small stations are modeled by forbidding overtakings. Large stations, however, are assumed to have enough capacity. Train orders on the open track are always preserved despite flexible running times. We do not consider re-routing and assume that in case of multiple tracks in a direction, the train lines are pre-assigned to one of the tracks.

The optimization model presented solves the timetable feasibility problem according to the given line structure and process duration bounds. In case a feasible timetable exists, it also generates stable timetables, and returns one optimized for stability. Even in case of infeasibility, a timetable with a cycle time larger than the nominal cycle time is generated, which is useful for finding bottlenecks.

The line patterns and activity duration bounds can be represented by a graph called a periodic event network (PEN) (Nachtigall, 1996, Großmann et al., 2012). The limitations of railway infrastructure capacity at stations or junctions can be modeled by infrastructure arcs in the PEN. On the other hand, the infrastructure limitations not occurring at a station or junction, but *between* two such points can be modeled implicitly by the duration bounds, or where necessary by using additional nodes. Based on the PEN, an optimization model can be formulated with the cycle time as the objective function to be minimized. This model can be reformulated as a Mixed-Integer Linear Programming (MILP) instance and can therefore be efficiently solved by available MILP solvers. Note that minimizing the cycle time does not mean that the train running and dwell times are also minimized, therefore a secondary objective can be added to ensure minimal running and dwell times for a given cycle time. Once the MILP formulation is solved, the results can be interpreted as follows. A feasible set of the optimization model with the cycle time not larger than the requested scheduled cycle time corresponds to a stable timetable, while the optimal solution of the optimization model corresponds to a stable timetable with minimal cycle time.

The minimum headway times from the infrastructure constraints can be based on normative headways, but the macroscopic model can also be connected to a microscopic model which generates the minimum headway time. In turn, the results of the model can be verified by microscopic means, and the model input parameters (e.g. headways) can be corrected if necessary (Besinovic et al., 2016; Goverde et al., 2016). The model can also be seen as a train order and overtaking location optimizer, that can be used as the 1st stage in a two-stage optimization model, where the second optimization model for instance optimizes robustness for fixed train orders and overtaking locations.

The remainder of this paper is organized as follows. Section 2 gives a literature review of related work. In Section 3 we define the optimization model that generates a timetable with maximum stability using flexible train orders and running and dwell times. We here also give an interpretation of different results returned by an optimization solver. Section 4 describes techniques to improve the running time of the solution process, including treating the multiplication of variables, dimension reduction and dynamic frequencies. Section 5 extends the solution approach to an iterative optimization method. Section 6 reports computational results on real-life networks proving the applicability of the iterative method on large networks. Finally, Section 7 gives conclusions.

2. Literature review

In this section we discuss literature in timetable generation and timetable evaluation, the two areas or timetable research that our approach covers in a unified manner. We close highlighing the literature gaps that this paper addresses.

2.1. Timetable generation

Cacchiani and Toth (2012) present a survey of timetabling methods, classifying them into *nominal* and robust methods, based on whether they take into account timetable stability or not. The methods are further classified, among others, whether they are used for periodic (regular interval) or aperiodic timetables, valid only on a corridor or on a network level, and different objective functions. Another classification aspect can be whether a timetabling model has a level of detail of the infrastructure including all switches, signals and block sections, a *microscopic* model, or only models the network as a set of stations and important junctions and the connecting lines, a *macroscopic* model. Using a microscopic model for a large network is impractical, and while a microscopic model is necessary to construct a conflict-free timetable for a large station or a complex junction. The approaches are therefore split into local microscopic models and macroscopic models capable of handling large networks with a lower level of detail. A further distinction can be made between models that consider different routing options and models that only focus on the timing and ordering of fixed-route trains. Note that a routing model is not necessarily microscopic. Using these concepts, the contribution of this paper can be described as a macroscopic fixed-route network-level periodic timetable generation and evaluation method with timetable stability the objective function – therefore in the literature research we also focus on related results.

One way timetabling trains differs from general scheduling problems in industrial engineering is that in many countries the train timetable is periodic, repeating usually every hour. Therefore the Periodic Event Scheduling Problem (PESP), defined by Serafini and Ukovich (1989) building on the *Program Evaluation and Review Technique* (PERT) (Malcolm et al., 1959), was applied to railways on a macroscopic level by numerous approaches. Schrijver and Steenbeek (1993) developed the algorithm named CADANS for solving the network timetable problem. Nachtigall (1996) models the train timetable with a periodic event network where nodes are train arrivals and departures and arcs are dwell, run or change activities. Based on the CADANS network timetable design model, the constraint programming system DONS was introduced (Schrijver, 1998; Kroon et al., 2009). DONS can find feasible solutions to the railway network of the Netherlands if it exists under the given initial parameters, or points to the critical constraints if a feasible solution does not exist.

Goverde (1999) reformulated PESP with buffer times as decision variables and exploiting the graph structure of the network to reduce the number of variables. Lindner (2000) extends the PESP formulation to include operational costs in the objective. Liebchen (2008) optimized a homogeneous, high-frequency metro network, implemented in practice. Kroon and Peeters (2003) extend the infrastructure constraints to variable running times, by inserting further dummy nodes into the graph where necessary. Another problem type successfully applied to timetable planning is the Quadratic Semi-Assignment

Problem (QSAP), e.g. by Schuele et al. (2009). A new MILP model scheduling extra freight trains into an existing passenger train timetable is presented by Cacchiani et al. (2010), with a Lagrangian heuristic that enables finding a feasible timetable in a few hours for a large instance. These models focus on timing with fixed train routing. On the other hand, Caprara et al. (2011) includes flexible routing solving the platform assignment problem with only small deviations allowed related to an existing macroscopic timetable. While applied for delay management and not initial timetable generation, the timetabling model used by Dollevoet et al. (2012) also takes into account passenger flows.

A common characteristic of the above approaches is that they treat the period of the timetable as a fixed parameter and they minimize a certain sum of travel time, waiting time or transfer waiting time. For example, Nachtigall and Voget (1996); 1997), Wong et al. (2008), Schuele et al., (2009), and Liebchen et al. (2010) minimize transfer waiting times, Nachtigall (1996) minimize the total travel time, and Kroon and Peeters (2003) define a general objective function dependent on any of the variables, but still using fixed period length. Lindner and Zimmermann (2005), however, optimizes for cost. Heydar et al. (2013), based on Bergmann (1975), use the cycle time as the objective in the timetable optimization. The authors assess the capacity of a single track, unidirectional railway line with passing loops by calculating the minimum period for given number of local and express trains using Mixed Integer Linear Programming (MILP). The dwell times at intermediate stations are variable and therefore the passing locations for trains are flexible. The main limitation of this model is that the train speeds are constant during the optimization and equal for each train class. Petering et al. (2016) extended the model of Heydar et al. (2013) to different train speeds and more than two station platform tracks, with now also the platforming problem as part of the optimization.

Robust timetable generation consists of timetable design methods where the reliability of the timetable is taken into account already during the design process. Looking at macroscopic network-level models, Kroon et al. (2008) expanded the PESP approach to a stochastic case. Fischetti et al. (2009) compare four different methods, including the *light robustness* method proposed in Fischetti and Monaci (2009), to improve the robustness of a timetable where robustness is represented by the average cumulative delay for a set of minor disruption scenarios, assuming no train cancellation or reordering. Cacchiani et al. (2012) proposed a method based on Lagrangian optimization that can produce a Pareto set of solutions with different weights on robustness. Liebchen et al. (2010) extend the light robustness approach concentrating on connection management during the distribution of timetable slack. Liebchen et al. (2009) defined the related concept of *recoverable robustness*, focusing on disruption scenarios and recovery from them not only using up buffer time, but also by breaking connections. All these methods assume an existing nominal timetable and the train orders are fixed during the robust optimization.

Besides macroscopic modeling on the network level, microscopic timetabling is necessary for complex station areas and junctions where routes are explicitly taken into account to check the feasibility of the network timetable and further optimize it, see e.g. Caimi et al. (2008) and Lusby et al. (2011) for large station areas and large junctions, respectively.

2.1.1. Railway capacity estimation and timetable evaluation

The most widely accepted standard to estimate railway capacity is defined in the Capacity leaflet of the International Union of Railways (UIC) (2004, 2013). This compression method based on blocking time theory counts as a microscopic model of a single line section, given the train orders and speeds: the *infrastructure utilization*, the ratio of the time period needed to execute a schedule and the nominal time period it is scheduled in.

A closely related concept applied to periodic timetables is the *minimum cycle time* as introduced by Goverde (2007). In a periodic timetable, the planned cycle time needs to exceed the maximum cycle mean over all circuits in the periodic event-activity network. In other words, for a given planned cycle time, a timetable is possible exactly if its minimum cycle time is not larger than the planned cycle time. The ratio of the minimum cycle time and the planned cycle time can be seen as a generalization of infrastructure utilization for periodic timetables to networks.

The minimum cycle time of these macroscopic models is therefore conceptually similar to the microscopic compression method. The macroscopic models use minimal departure and arrival time headways as input parameters, and these values can be obtained using the microscopic compression method. The macroscopic models, on the other hand, can include other time constrains between event pairs besides the limited infrastructure capacity, such as passenger connections, crew and rolling stock-related constraints, and any constraints linking multiple corridors together.

While there are several other methods that aim at estimating capacity independent of the timetable (De Kort et al., 2003; Lindner, 2011; Mussone and Wolfler Calvo, 2013), in this paper we focus on timetable-dependent capacity and stability.

The key concept in timetable evaluation is the timetable *stability*. The macroscopic network-level stability of a periodic timetable can be defined using timetable cycle times: the relationship between the nominal and the minimum cycle time describes the capacity utilization of the timetable (Goverde, 2007, 2014): the timetable is stable exactly if the minimum cycle time T is less than the nominal cycle time T_0, i.e. $T < T_0$, and the larger $T_0 - T$ is, the more time reserve there is available. This ratio $\frac{T}{T_0}$ is defined in Goverde (2007) as *network throughput*. Therefore there is a strong relationship between the capacity of the physical network and the stability of the timetable: infrastructure capacity determines the pace at which the timetable can be executed, therefore the minimum cycle time T, and the stability of the timetable can be described as the relationship between T and T_0. The max-plus algebra technique has been successfully applied for timetable evaluation using the cycle time (Braker, 1993; Goverde, 2007; Heidergott et al., 2005). For a more detailed analysis of stability, delay propagation and max-plus algebra, see also Goverde (2010).

2.2. Literature gap

The key contribution to our approach is the combination of network timetable generation and stability evaluation in a single optimization method, independent of simulation or delay scenarios. We have seen that the timetable generation methods typically used travel times, transfer waiting times, or cost as their objective functions. The only known periodic timetable generation models using the cycle time as the objective are Heydar et al. (2013) and Petering et al. (2016) for a unidirectional single track railway line with multiple passing loops. Timetable evaluation methods, on the other hand, need an existing timetable with predefined run and dwell times and train orders to analyse. Our approach optimizes the train order, run and dwell times, and overtaking locations of a line plan to generate a network-optimal timetable, including its evaluation for stability and bottlenecks in a single optimization problem.

3. A railway timetable optimization model focusing on stability

3.1. Initial assumptions

Without loss of generality we further assume that the planned cycle time is 60 min: the approach can be easily generalized for other periods such as 30 or 120 min. While the following model can also be extended to single track lines, for simplicity we assume that all railway tracks are one directional. This limitation still allows the modeling of large high capacity utilization railway networks of 2–4 track lines, such as the central part of the Dutch railway network.

3.2. The periodic event-activity network

Following the notation in Schöbel (2006) for event-activity networks and applying it for the periodic case just as in case of a PEN (Nachtigall, 1996), we model a railway system as the *periodic event-activity network* $\mathcal{N} = (\mathcal{E}, \mathcal{A}, T)$, where T is the common cycle time for all events. An event $i \in \mathcal{E}$ is a tuple

$$i = (Line_i, Station_i, EventType_i), \tag{1}$$

where *Line* is the train line identifier, unique for each one directional train run per one cycle time period: i.e. in case of the cycle time of one hour, if a bidirectional train service runs twice per hour, that corresponds to four lines. While in general, the cycle time can be different from one hour, all events use the same cycle time. *Station* is a train station, junction or other timetable point; with a binary flag assigned to each station describing whether overtaking is possible at the given station. *EventType* can take values from the set {*dep, arr, thr*}, representing departure, arrival, and through events, respectively. Furthermore, for each station, a binary variable is set whether overtakings between two trains are possible or not. Additional information on the station locations and line types can be given for visualization purposes.

We also define the subsets $\mathcal{E} = \mathcal{E}_{\text{dep}} \cup \mathcal{E}_{\text{arr}} \cup \mathcal{E}_{\text{thr}}$ corresponding to each *EventType*, to simplify notation.

An activity $a_{ij} \in \mathcal{A}$ is a tuple

$$a_{ij} = (i, j, ActivityType_{ij}, L_{ij}, U_{ij}), \tag{2}$$

where $\{i, j\} \subset \mathcal{E}$ are respectively the start and end events,

$$ActivityType_{ij} \in \{run, dwell, infra\}$$

where *infra* stands for minimum infrastructure headways, and the allowed range of the activity duration is

$$(L_{ij}, U_{ij}).$$

Again, we define the subsets $\mathcal{A} = \mathcal{A}_{\text{dwell}} \cup \mathcal{A}_{\text{run}} \cup \mathcal{A}_{\text{infra}}$ corresponding to the *ActivityType* values to simplify notation. A run activity connects a departure or through event to an arrival or through event, a dwell activity always connects an arrival event to a departure event, while an infrastructure activity can connect any two activities.

Then, for a given cycle time t and ensuring that for all processes a_{ij}, $0 \leq L_{ij} \leq U_{ij} < t$ (see below), a timetable is an assignment $i \to \tau_i$ of periodic scheduled times to each event $i \in \mathcal{E}$ so that for each $a_{ij} \in \mathcal{A}$,

$$L_{ij} \leq (\tau_j - \tau_i \mod t) \leq U_{ij}. \tag{3}$$

Note the ordering of events in an infrastructure activity does not symbolize an ordering of events, as an infrastructure activity with lower and upper bounds of l, u, respectively, is equivalent to an edge between the same two events but directed in the inverse direction and with lower and upper bound $T - u$, $T - l$, respectively. In other words, in the graph representation of the periodic event-activity network, an infrastructure edge can be traversed in both forward and backward direction.

3.2.1. Modeling a single train line

A single train line run can be modeled in an intuitive way as a time-ordered directed chain of departure, arrival, and optionally through events, connected by run and dwell activities. The departure and arrival events are unambiguously defined by the stations where the train stops at for boarding or alighting passengers, while the through events need to be defined at

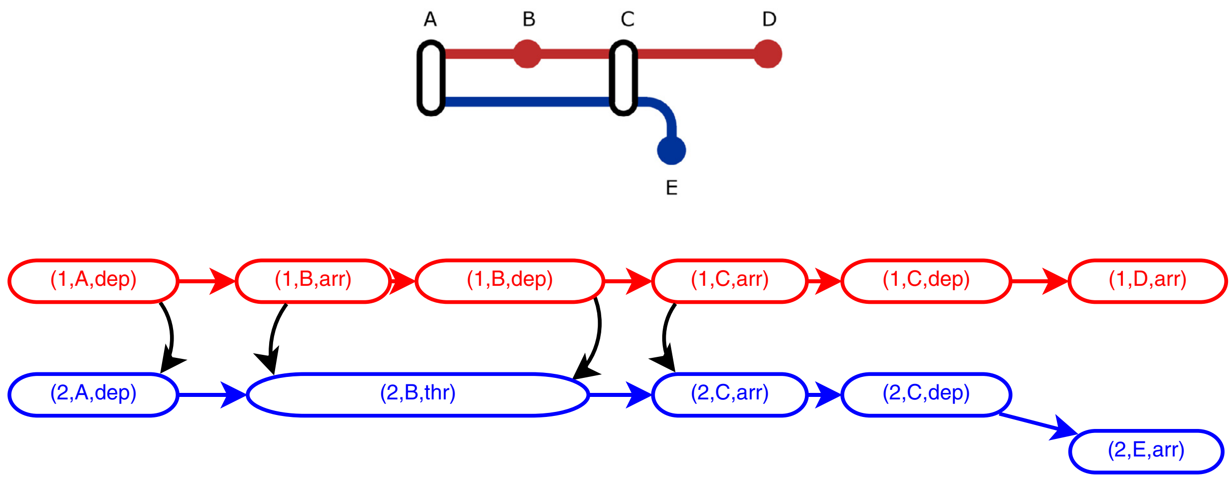


Fig. 1. Schematic line graph and periodic event-activity network of two train lines (Line 1 in red and line 2 in blue) with infrastructure activities (in black). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

any intermediate station, junction, or other timetable point where infrastructure activities need to be defined, as described in the next section.

The run and dwell activity durations are defined as follows. The minimum run and dwell times can be set based on predefined norms, measurements or simulations. For example, minimum dwell times might be defined by a train operator for each train type and station type; while minimum running times can be a lower percentile value of a series of running time measurements, optionally with an extra proportional slack time added as required.

The maximum run and dwell times, on the other hand, can be much more freely set: after all, a large run or dwell duration in this model can be directly translated to a large running or dwell time on the real train network. However, there might be business norms in effect, that require that the train run meets expectations by maintaining a limited deviation from its minimum possible running and dwell times. These norms or expectations can then either directly be translated into the maximum durations of running and dwell times, or, as we will see later, defined as a higher level constraint for the whole train line, allowing for larger local deviations and therefore more flexibility.

3.2.2. Modeling limited infrastructure capacity

Following Odijk (1996), we also introduce infrastructure activities representing minimum headway constraints between consecutive trains because of shared infrastructure resources. On the macroscopic level, most headway constraints can be modeled as a minimum time separation between two events of two trains, and this minimum duration can be measured or simulated with microscopic tools. This type of infrastructure constraint can conveniently be modeled in our periodic event-activity network with infrastructure activities connecting train events. In case a minimum time separation is necessary at a location where previously no train events were defined, it is necessary to define through events for all affected trains by splitting the run activities into two, and consequently the infrastructure activities can be defined between all these events.

We define a *timetable point* as a station, a junction or other location where trains can have scheduled times. In case there are more than two trains using the same timetable point sharing some infrastructure, then in general a full graph between all these events needs to be defined. It is possible, however, to avoid pairs of activities between each two events and just define one activity, according to Goverde (1999), as follows. Let the activities be i and j, and the directed minimum headway times L_{ij} and L_{ji}, for which $L_{ij} + L_{ji} \leq T$ holds. Then a single infrastructure activity a_{ij} can be defined with the bounds $(L_{ij}, T - L_{ji})$ periodic with cycle time T according to (3).

See Fig. 1 for an example periodic event-activity network of two train lines, where "(1, A, dep)" is the *dep* (departure) event of Line 1 at station *A*, etc. Train line 1 is a local train stopping at stations A, B, C, and D. Train line 2 is an express train line stopping at station A, running through station B without stopping, and stopping at stations C and E. Each train line is represented by departure, arrival and through nodes, and run and dwell activities in between. Infrastructure constraints at the shared resources, leaving station A, entering and leaving station B, and entering station C, are represented by infrastructure activity arcs. In case of flexible run and dwell times, however, the model above is not sufficient anymore to ensure the separation of trains using shared infrastructure, as pointed out in Kroon and Peeters (2003). In particular, the following two scenarios are valid according to the above model, but impossible in reality:

1. On the open track, one train overtakes another one, while still observing the relevant departure–departure and arrival–arrival headways.
2. At a station without overtaking facility, one train overtakes another one, while still observing the relevant arrival–arrival and departure–departure headways.

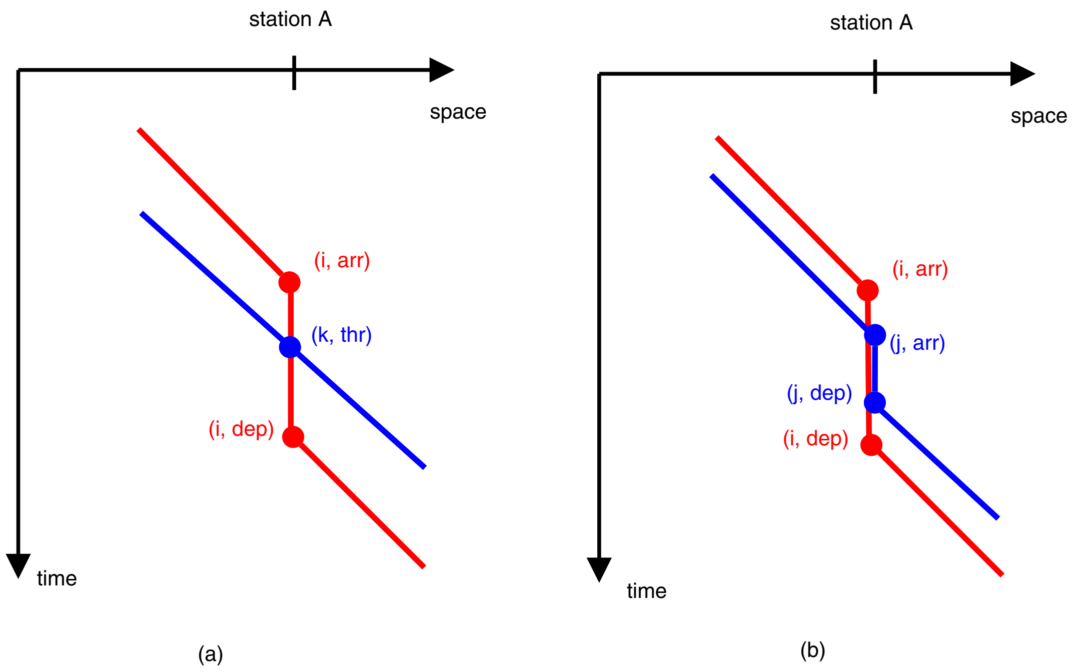


Fig. 2. Exemplary time–space diagram for an overtake at a station where the slower train is stopping, with (a) overtaking train not stopping and (b) overtaking train stopping.

Kroon and Peeters (2003) give the solution to the first problem by introducing further dummy nodes and infrastructure activities where necessary. We use the same approach, and extend it in a similar fashion for stations where train overtaking is not possible.

Let us consider such an "illegal" station overtaking as the following: at station A, train line i is overtaken by train line j running in the same direction, while both trains stop at the station. In another case, again at station A, train line i is overtaken by train line k running in the same direction, but in this case while train i stops at station A, train k does not. The following approach can be iteratively extended for further overtaking trains that may either stop of not. In case of train k, for events to take place in the violated order of (i, arr), (k, thr), (i, dep), where we omitted the station index for clarity, the following have to be true (see Fig. 2(a)):

$$L_{(i,arr),(k,thr)} + L_{(k,thr),(i,dep)} \le U_{(i,arr),(i,dep)}. \tag{4}$$

Similarly, for train j, the following needs to be true for an invalid overtaking in the form of the event sequence (i, arr), (j, arr), (j, dep), (i, dep) (see Fig. 2(b)):

$$L_{(i,arr),(j,arr)} + L_{(j,arr),(j,dep)} + L_{(j,dep),(i,dep)} \le U_{(i,arr),(i,dep)}. \tag{5}$$

Therefore the following preprocessing is necessary to ensure that no invalid overtakings take place: for each station where overtaking is not possible, all trains stopping need to be checked for the above inequalities if any other train can overtake them and in case of violation, dummy node(s) need to be inserted in the dwell process of the overtaken train to ensure that the overtaking is not permitted by the constraints.

The detailed method to insert these dummy dwell nodes is the following: for every such dwell activity of a potentially slower train where an illegal overtaking would be possible according to one of the above two inequalities, among all potentially overtaking trains k we take one where $L_{(i,arr),(k,thr)} + L_{(k,thr),(i,dep)}$ or $L_{(i,arr),(k,arr)} + L_{(k,arr),(k,dep)} + L_{(k,dep),(i,dep)}$ is minimal, and let this minimal sum of lower bounds be L_{fast} for the faster train. We also denote $U_{slow} = U_{(i,arr),(i,dep)}$ for brevity. Then this dwell process needs to be split by $n = \lfloor U_{slow}/L_{fast} \rfloor$ additional nodes into $n+1$ sections.

We now introduce *dummy* as a new possible value of *EventType* and insert the dummy event nodes ($Line_i$, $dummy_1$), ... , ($Line_i$, $dummy_n$) (omitting the station ID in the notation for clarity as before). To simplify notation, let us rename (i, arr) to (i, $dummy_0$) and (i, dep) to $(i, dummy_{n+1})$ so that we can refer to the dwell activities with dummy nodes as $((i, dummy_x), (i, dummy_{x+1}))$ regardless if they include the arrival or departure node or not. Then replace the activity

$$\left\{(i, arr), (i, dep), dwell, L_{(i,arr),(i,dep)}, U_{(i,arr),(i,dep)}\right\}$$

(where L and U stand for the initial bounds of this activity) with the ordered chain of activities

$$\left\{(i, dummy_x), (i, dummy_{x+1}), dwell, L_{(i,arr),(i,dep)}/(n+1), U_{(i,arr),(i,dep)}/(n+1)\right\}, \tag{6}$$

for $0 \le x \le n$.

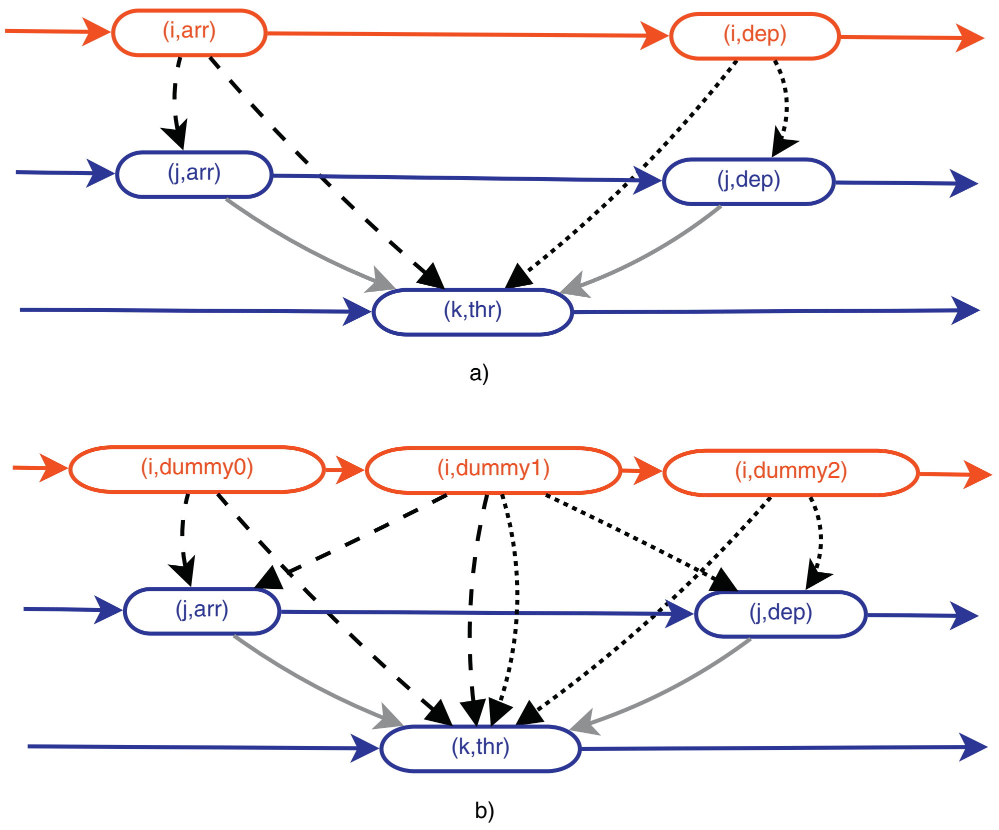


Fig. 3. Example events and activities before (a) and after (b) inserting a single dummy node for a station dwell with one faster train, not stopping (dashed black and dotted black arrows are the infrastructure events related to the original arrival and departure nodes, respectively; gray arrows are infrastructure arcs not related to the dummy node extension).

Finally, we create n copies of any infrastructure constraint involving events $(i, arr) = (i, dummy_0)$ and $(i, dep) = (i, dummy_{n+1})$ and in the replicated versions we replace the applicable original event with each of the dummy nodes (i, $dummy_x$) for $1 \leq x \leq n$ (excluding, thus, the first and last dummy node, i.e. the original arrival and departure nodes, $(i, arr) \equiv (i, dummy_0)$ and $(i, dep) \equiv (i, dummy_{n+1})$). An example is shown in Fig. 3(b) for a faster stopping and a faster not stopping train and $n = 1$, the situation is analogous for more faster trains, and multiple dummy nodes. If this results in multiple infrastructure activities between the same two events, the multiple arcs can be merged into a simple arc using the maximum of the required infrastructure headways across the multiple arcs, such as in the case of the parallel dashed and dotted arcs in Fig. 3(b) between nodes (i, A, $dummy1$), and (k, A, thr). Having inserted the dummy node(s), we show why an illegal overtaking is now disabled for the case of $n = 1$ and the overtaking train not stopping, the same can be shown similarly for the other cases. An overtaking would mean an order of events

$$(i, dummy_x),\ (k, thr),\ (i, dummy_{x+1}),$$

for some x, as in Fig. 4. (Recall that as train orders are flexible and infrastructure headways are modeled by a single one-directional arc with lower and upper bounds between two points, the direction of the infrastructure arc does not symbolize the ordering of trains within the hour. Therefore a path of consecutive events within the hour may traverse an infrastructure arc backwards.) As the constraints are satisfied, for the sequence of events $(i, dummy_x)$, (k, thr), $(i, dummy_{x+1})$, the upper bound of the duration of activity $((i, dummy_x), (i, dummy_{x+1}))$ has to be at least the sum of the lower bounds of the durations of activities ((i, $dummy_x$), (k, thr)) and $((k, thr), (i, dummy_{x+1}))$:

$$L_{(i,dummy_x),(k,thr)} + L_{(k,thr),(i,dummy_{x+1})} \leq U_{(i,dummy_x),(i,dummy_{x+1})}.$$

On the one hand, $U_{(i,dummy_x),(i,dummy_{x+1})}$ was defined in (6) a $U_{(i,arr),(i,dep)}/(n+1)$. On the other hand, as activities ((i, $dummy_x$), (k, thr)) and $((k, thr), (i, dummy_{x+1}))$ are copies of ((i, arr), (k, thr)) and ((k, thr), (i, dep)) respectively, their lower bounds are equal to $L_{(i,\ arr),\ (k,\ thr)}$ and $L_{(k,\ thr),\ (i,\ dep)}$ respectively. Therefore it follows that

$$L_{(i,arr),(k,thr)} + L_{(k,thr),(i,dep)} \leq U_{(i,arr),(i,dep)}/(n+1).$$

Rearranging for n, using the definition of n and the earlier notations of U_{slow} and L_{fast},

$$U_{slow}/L_{fast} - 1 \geq n = \lfloor U_{slow}/L_{fast} \rfloor$$

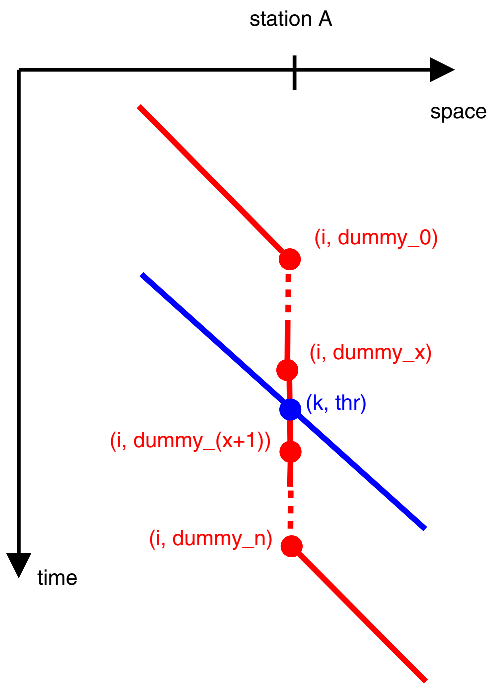


Fig. 4. Time–space diagram for an overtaking without stopping including a station dummy node.

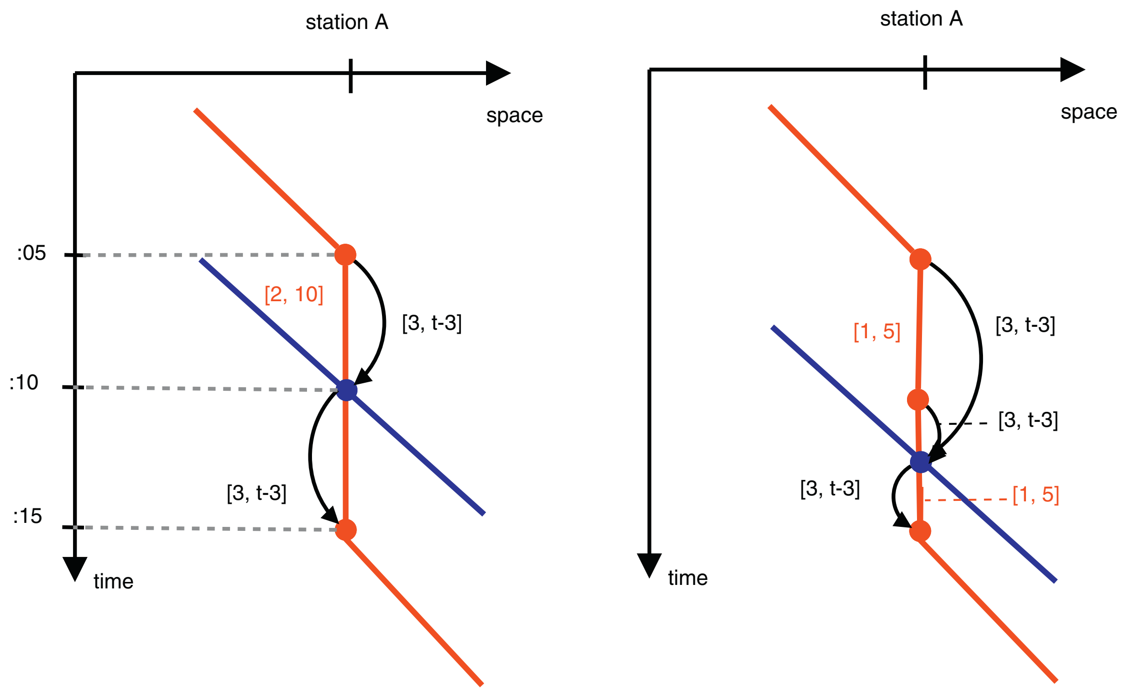


Fig. 5. Numerical example for illegal overtaking (left) and added dummy node (right).

which is not possible, therefore we proved that the illegal overtaking now is disabled by the constraints including the dummy node.

To give a numerical example, the left side of Fig. 5 shows an illegal overtaking. This is in line with the original constraints, as the maximum dwell time is 10, while both minimum infrastructure headways are 3, therefore a timetable of 5, 10, and 15 for the arrival, pass, and departure, respectively, satisfies the activity bounds. In case that this station does not allow for overtakings, $\lfloor 10/(3+3) \rfloor = 1$ dummy node is added to split the dwell process into two, and the two infrastructure arcs are duplicated to connect to the dummy node as well. After this, an overtaking is not possible anymore, as follows. Let us assume without loss of generality that it happens between the dummy and the departure node (the argument is identical for between the arrival and dummy node). This on the one hand would mean that the dummy node and the overtaking point is separated by at least 3 min and the overtaking and the departure node are separated by at least 3 min, therefore the dummy and departure nodes are separated by at least 6 min. On the other hand, the maximum separation between the dummy and the departure node is at most $10/(n+1) = 5$ min, leading to an infeasible solution.

The usefulness of the above approach lies in the fact that in practice, there are relatively few locations in the original graph where the assumptions above are violated and dummy nodes need to be inserted, and even within these cases the vast majority needs only one dummy node per case.

3.2.3. Modeling processes longer than the cycle time

Above we assumed that all activities have duration bound ranges smaller than the cycle time. It can happen though, that this is not immediately the case, e.g. in case of long running times. This would lead to a modeling issue as we will require that for all activities, $U_{ij} < T$ in order to be able to define constraints including the binary variables $z_{ij} \in \{0, 1\}$, as we will see later. Therefore in case of processes initially longer than the cycle time, these need to be split into shorter sections with dummy events connecting them until this condition is satisfied. Concretely, if for an activity (i, j) it holds that $U_{ij} \geq T$, then let $n = \lfloor U_{ij}/T \rfloor + 1$ and then we split (i, j) to n segments by introducing dummy events $dummy_1, \dots, dummy_{n-1}$ and replacing activity (i, j) with activities $(i, dummy_1)$, $(dummy_1, dummy_2)$, ... , $(dummy_{n-1}, j)$ where the lower and upper bounds of each are L_{ij}/n and U_{ij}/n, respectively. For the bounds L, U of these new activities it will all hold true that $L \leq U < T$.

Later the cycle time will become a variable: then the exact process is to define a lower bound L for the variable cycle time $t \geq L$ and ensure that all duration bounds are lower than this cycle time lower bound, i.e., $L_{ij} \leq U_{ij} < L, \forall (i, j) \in \mathcal{A}$. This can be achieved in an identical manner as above for the case of T.

3.3. The optimization model

Building on the periodic event-activity network as defined earlier, finding a periodic timetable with given cycle time T can be defined as the periodic event scheduling problem

$$L_{ij} \leq \tau_j - \tau_i + z_{ij}T \leq U_{ij} \qquad \text{for all } (i, j) \in \mathcal{A} \tag{7}$$

$$0 \leq \tau_i < T \qquad \text{for all } i \in \mathcal{E} \tag{8}$$

$$z_{ij} \in \{0, 1\} \qquad \text{for all } (i, j) \in \mathcal{A}, \tag{9}$$

where for the bounds it holds that $0 \leq L_{ij} \leq U_{ij} < T$ for all $(i, j) \in \mathcal{A}$. Note that for $(i, j) \in \mathcal{A}_{\text{infra}} \subset \mathcal{A}$ it holds that $U_{ij} = T - L_{ji}$ with $L_{ij} + L_{ji} \leq T$.

Consider a solution to the above feasibility model, with $\tau_i\ \forall i \in \mathcal{E}$ being the event time of each event, and the binary $z_{ij}\ \forall (i, j) \in \mathcal{A}$ variables describing for each activity the order of events within the timetable period. We now state the earlier definition (see Section 3.1) of the minimal cycle time more concretely: the minimal cycle time of this timetable, more precisely of this set of event orders, is the minimal value of T for which a solution with the same values of $z_{ij}\ \forall (i, j) \in \mathcal{A}$ exists (but possibly with different values of $\tau_i\ \forall i \in \mathcal{E}$) .

Intuitively, the minimal cycle time is the following. Take a railway network in a periodic timetable, and modify the dispatching rules such that the trains do not need to wait for their scheduled departure and arrival events, however, they do respect all minimal process times defined between any event pairs, and therefore also the order or trains is not changed. Make the trains depart and arrive, within these rules, as soon as possible. In this system, any periodic event will occur with a period not longer than the minimal cycle time.

Recall that stability is the ability of a timetable to absorb delays so that delayed trains return to their scheduled train paths without rescheduling: a timetable that can return to schedule faster from the same delay disturbance is considered more stable. Therefore the minimum cycle time of a periodic timetable is a valuable estimator of its stability (Braker, 1993; Heidergott et al., 2005; Goverde, 2007): the difference between the period and the minimal cycle time is the minimum mean hourly delay reduction that is certainly possible given any initial delay and no further disturbance, with no train re-ordering or cancellation, regardless of which event or how many events are delayed.

We argued that for otherwise comparable timetables of identical train stopping patterns and frequencies, the one with lower minimum cycle time is the more stable. Therefore we transform the above feasibility problem into an optimization problem with variable cycle time t that is to be minimized:

$$\text{Minimize } t \tag{10}$$

subject to

$$L_{ij} \leq \tau_j - \tau_i + z_{ij}t \leq U_{ij} \qquad \text{for all } (i, j) \in \mathcal{A} \tag{11}$$

$$0 \leq \tau_i < t \qquad \text{for all } i \in \mathcal{E} \tag{12}$$

$$L \le t \tag{13}$$

$$z_{ij} \in \{0, 1\} \qquad \text{for all } (i, j) \in \mathcal{A}, \tag{14}$$

where $L > 0$ is a lower limit on t, and $0 \le L_{ij} \le U_{ij} < L$ for all $(i, j) \in \mathcal{A}$. Note that the variable cycle time t and the event times τ_i are real numbers, so we do not assume that these values are in full minutes but in particular support a precision of 1 s as well. Note that an internal working timetable can have a precision of 1 s, while the timetable published to passengers is rounded down to full minutes, without leading to either missed trains or a perception of too much delay.

Because the cycle time is a decision variable, constraint (11) is now nonlinear as the product $z_{ij}t$ of two variables. However, we can reformulate this constraint to obtain a MILP model. Let U be a suitably large upper bound for the objective value t. A product zt of a binary variable z and a bounded continuous variable $t \in [0, U]$ can be reformulated into linear inequalities using the new variable $y = zt$ with (Williams, 1990)

$$0 \le y \le Uz \tag{15}$$

$$t - U(1 - z) \le y \le t. \tag{16}$$

Note that if $z = 0$ then (15) and (16) gives $y = 0$ and $t \in [0, U]$, while if $z = 1$ it becomes $y = t$ and $t \in [0, U]$. Then the nonlinear problem (10)–(14) can be formulated into the MILP:

$$\text{Minimize } t \tag{17}$$

subject to

$$L_{ij} \le \tau_j - \tau_i + y_{ij} \le U_{ij} \qquad \text{for all} (i, j) \in \mathcal{A} \tag{18}$$

$$\tau_i < t \qquad \text{for all } i \in \mathcal{E} \tag{19}$$

$$y_{ij} \le t \qquad \text{for all } (i, j) \in \mathcal{A} \tag{20}$$

$$y_{ij} - Uz_{ij} \le 0 \qquad \text{for all } (i, j) \in \mathcal{A} \tag{21}$$

$$y_{ij} - t + U\left(1 - z_{ij}\right) \ge 0 \qquad \text{for all } (i, j) \in \mathcal{A} \tag{22}$$

$$L \le t \le U \tag{23}$$

$$\tau_i \ge 0 \qquad \text{for all } i \in \mathcal{E} \tag{24}$$

$$y_{ij} \ge 0 \qquad \text{for all} (i, j) \in \mathcal{A} \tag{25}$$

$$z_{ij} \in \{0, 1\} \qquad \text{for all } (i, j) \in \mathcal{A}. \tag{26}$$

Furthermore, the strict inequality (19) can be replaced by the inequality $\tau_i - t \le \delta$ using a suitable small δ, such as 1 s. The MILP (17)–(26) is the extension of the PESP (7)–(9) to a variable cycle time which must be minimized.

3.4. Interpreting the optimization model results

Although the optimality of the generated timetable can only be guaranteed if the solver reached the optimal solution, information about the timetable can be inferred in several of the other cases as well. Table 1 shows how the different solver statuses and related variable value ranges can be interpreted, including t_{LP}, the objective value of the LP relaxation of the model, which is therefore a lower bound on t. On the one hand, if an intermediate feasible solution exists with $t \le T$, we can already conclude that a stable timetable exists and it is generated by the model. On the other hand, irrespective of whether a feasible solution is known or not, if we know that $T < t_{LP}$ then we already know that a stable timetable is not possible. This of course is also the case if the MILP model is infeasible or if for the optimal solution, $T < t$.

While the core idea of the model is to keep the cycle time flexible, we have seen above that the model definition restricts the range of the cycle time to $[L, U]$. This in practice does not restrict the flexibility of the cycle time variable if the optimization is executed in an iterative fashion. In case that the calculation returns a cycle time equal to this predefined lower bound, the solver can be restarted with a decreased lower bound. Likewise, in case of infeasibility a higher upper bound can be used in a subsequent run. Furthermore, if the sole goal of an optimization run is to decide feasibility, both cycle time bounds can be set to the nominal cycle time and the model degenerates to a feasibility problem generating a stable timetable if possible, without taking stability into account.

Table 1
Interpretation of optimizer results.

Solver status	Variable ranges	Interpretation		
		A timetable exists?	A timetable generated?	Optimal timetable generated?
Intermediate, no solution	$t_{LP} \leq T$	Unknown	No	No
	$T < t_{LP}$	No	N/A	N/A
Infeasible		No	N/A	N/A
Intermediate solution	$t \leq T$	Yes	Yes	Unknown
	$T < t$ $\cap$ $t_{LP} \leq T$	Unknown	No	No
	$T < t_{LP}$	No	N/A	N/A
Optimal solution	$t \leq T$	Yes	Yes	Yes
	$T < t$	No	N/A	N/A

T is the nominal cycle time, t is the best solution value for the minimum cycle time, and t_{LP} is its best LP lower bound for the given optimizer run.

3.5. The expanded timetable

As mentioned earlier, a feasible solution to the optimization problem with t corresponds to a stable periodic timetable with cycle time t. However, what in practice really is required, is a periodic timetable according to the nominal cycle time T. There are a few ways to transform and further optimize the obtained timetable to end up with a timetable of cycle time T.

The simplest method, that might not be sufficient in all practical purposes, is rescaling, resulting in a valid but potentially too slow timetable (if $t \ll T$). Note that as we use the minimum cycle time for our objective function, the rescaled timetable is still optimally stable as the minimum cycle time is unchanged due to rescaling. As long as $t < T$, a feasible set of the optimization problem can be transformed to have cycle time T in a straightforward way by scaling each event time τ_i to $\tau_i \cdot T/t$ and then the expanded timetable still obeys all minimum running, dwell time and infrastructure headway constraints while keeping the same minimum cycle time when compressing the timetable. To see an example of such a timetable expansion, see Fig. 11 in Section 6.

Note that the minimum cycle time is equal for the expanded and the compressed timetable because of the definition of the minimal cycle time: both timetables have the same event orders, and the minimal cycle time is the shortest period for which a feasible timetable exists for the given event orders. As the compressed timetable was optimized to have the minimal cycle time for the given line pattern inputs, this means that the expanded timetable has maximal gap between the minimal cycle time and the nominal timetable period, and therefore maximal stability.

This expanded timetable fixes the departure event times and includes some running time supplements and headway buffer times due to the expansion. In case of initial delays, the delay absorption is thanks to delayed trains making use of these supplements and buffer times and their delay being reduced at each such event. Trains that are not delayed by an initial delay or a by a knock-on delay do not alter their planned event times, but with no train re-ordering this does not cause further delays: if the timetable is feasible, then because of the linearity of the model an event either is on time, has primary delay, or has at most the delay amount of one of the events it depends on via a time constraint.

In case that the rescaled timetable is not yet satisfactory, it is possible to further improve as follows. If some run or dwell times become unacceptably long by the expansion, the original model can be re-executed with lower upper bounds. The model can be also re-run with the constants $L = U = T$ and using the sum of run and dwell times as the objective function to be minimized, see also the “Line-level activity upper bounds” in the next section.

Finally, the output of this model can serve as an initial solution to another timetable optimization model that may improve travel times or other objectives but needs pre-set train orders. This second stage optimization model is an LP problem with all original binary variables fixed, as the optimal train order for maximal throughput is determined by the current model. For more advanced approaches for this follow-up optimization step of buffer time allocation, see Kroon et al. (2008) and Burdett and Kozan (2015).

3.6. Model extensions

While the optimization model above is able to capture the scheduling problem of a periodic train network with flexible train orders, running and dwell times, taking into account infrastructure limitations and optimizing for timetable stability, there are a few timetable requirements that are often present in practice and need to be incorporated into the model to ensure its usefulness. In the following, we describe such extensions, addressing train services at frequencies higher than once per timetable period, parallel train lines, connection constraints, and line-level duration upper bounds.

We assumed that all train lines depart from their first station once within each period. In practice, train lines can have different frequencies, with the lowest frequency lines defining the main period length and other trains running e.g. twice or four times per period. Higher frequency train lines can be modeled easily as a set of multiple train lines with frequency one each, however, in this case it is not guaranteed that these trains will have a regular interval service. In fact, the optimal solution will often be "batching" all these trains together: running them directly one after the other. Therefore, a new type of constraint is necessary to ensure that these train lines are separated in a regular fashion.

We extend the tuple definition of events with a counter *RunNr* indicating the successive train runs within the period. Therefore, an event $i \in \mathcal{E}$ becomes the tuple

$$i = (Line, Station, EventType, RunNr).$$

For all trains running once per period $RunNr = 1$. We further define the set of *regularity activities*

$$\mathcal{A}_{reg} := \{(i,j) \times \mathcal{E}x\mathcal{E} \mid Line_i = Line_j, Station_i = Station_j,\ EventType_i = EventType_j, RunNr_i + 1 = RunNr_j\}.$$

Then using the notation F_{Line_i} for the frequency of the line of event i for the given cycle time, we introduce the constraint

$$\tau_j - \tau_i + z_{ij}t = t/F_{Line_i} \qquad \text{for all } (i,j) \in \mathcal{A}_{reg} \tag{27}$$

which can be converted to the MILP-compatible form

$$\tau_j - \tau_i + y_{ij} - (1/F_{Line_i})t = 0 \qquad \text{for all} (i,j) \in \mathcal{A}_{reg} \tag{28}$$

$$y_{ij} \le t \qquad \text{for all } (i,j) \in \mathcal{A}_{reg} \tag{29}$$

$$y_{ij} - Uz_{ij} \le 0 \qquad \text{for all } (i,j) \in \mathcal{A}_{reg} \tag{30}$$

$$y_{ij} - t + U\left(1 - z_{ij}\right) \ge 0 \qquad \text{for all } (i,j) \in \mathcal{A}_{reg} \tag{31}$$

$$y_{ij} \ge 0 \qquad \text{for all} (i,j) \in \mathcal{A}_{reg} \tag{32}$$

Note that Eqs. (29)–(32) are identical to Eqs. (20)–(22) and (25) and can thus be conveniently merged by extending $\mathcal{A}$ with $\mathcal{A}_{reg}$. Eq. (28) is a new type of equality as it includes the cycle time multiplied by some factor.

Another common market requirement concerns groups of train lines that have a different route or stop pattern, but share their route and stop pattern on a substantial part of their trips. In this case, it is often desired that these lines are evenly spaced on the common route with equal stop pattern to providing a regular service. Such a constraint can be added using the same method as the high-frequency train lines above, except that the events connected by such a regularity activity belong to different train lines.

Further constraints can be added reflecting certain business requirements. One such a requirement can be a constraint on the duration between the arrival of one train and the departure of another, to ensure smooth passenger connections. Such requirements can be added as new activities in the graph using the structure of inequalities used for run and dwell times: we can define a new activity type $\mathcal{A}_{transfer}$ that connects the arrival of one train to the departure of another train and its bounds are the acceptable minimum and maximum transfer times including walking time.

3.6.1. Line-level activity upper bounds

The values of running and dwell activity upper bounds are decided as a compromise between timetable flexibility and the amount of acceptable time allowances: too little allowed time allowance restricts the model in finding feasible solutions, while too much time allowance can result in running times unacceptably high in practice. However, these two bounds for time allowances are meaningful in different scale: flexibility in running time is more important in a local level planning through a bottleneck, while a practical threshold for maximum time allowance is more applicable regarding the whole line or a longer segment. In other words, the model can be improved by allowing for more flexibility locally, given that the time allowances are not too high on a global level. To achieve this, the running and dwell activity upper bounds can be increased, and to limit the extension of running times of each line, new upper bounds are defined for the sum of run and dwell times of each line.

Let $\mathcal{A}_l$ be all run and dwell activities of a single run of line l (note that in case of multiple runs, it is sufficient to consider only the first), i.e.

$$\mathcal{A}_l = \left\{(i,j) \in \mathcal{A}_{\text{run}} \cup \mathcal{A}_{\text{dwell}} | Line_i = Line_j = l, RunNr_i = RunNr_j = 1\right\}, \tag{33}$$

and let *Lines* the set of all lines. Then if such suitable upper bound for line l is U_l, then the new constraints are as follows:

$$\sum_{(i,j)\in\mathcal{A}_l} \left(\tau_j - \tau_i + y_{ij}\right) \le U_l \qquad \text{for all } l \in Lines. \tag{34}$$

Note that the model only stays feasible if

$$\sum_{(i,j)\in\mathcal{A}_l} L_{ij} \leq U_l \quad \forall\, l \in Lines \tag{35}$$

and for line l these constraints can only become an equality if

$$U_l \leq \sum_{(i,j)\in\mathcal{A}_l} L_{ij}. \tag{36}$$

4. Dimension reduction techniques

The following pre-processing steps aim at reducing the search space of the MILP model.

4.1. Connected components

In the graph representation, different train lines are connected to each other because of the infrastructure, regularity and connection constraints. However, it can still be that the graph is separable to multiple connected components, for example if a train line is operated independently of other lines. As one connected component has no effect on the others, the optimization model could be executed separately for each connected component. In the later examples in this paper, all cases are connected, i.e. they consist of a single connected component.

4.2. Greatest common divisor of frequencies

In case of regular-interval timetables, often many train services have a headway time of less than the full hour, namely 30, 20, 15 or sometimes even 10 min. If the greatest common divisor g of all line frequencies is larger than 1, the timetable can be calculated with updated frequencies $F' = F/g$. This reduces the number of events by a factor of $1/g$ and reduces the number of processes even to a larger extent, because of the structure of the infrastructure processes connecting all events related to a given resource. The new calculated timetable with cycle time t can then be simply scaled back to the original frequencies by duplicating the events at times τ_i to new times $\tau_i + t,\ \tau_i + 2t,\ \dots,\ \tau_i + (g-1)t$ and the re-scaled timetable has a minimum cycle time $g \cdot t$.

4.3. Single train line with frequency $F = 1$

It is possible to go one step further in the spirit of the greatest common divisor reduction described above, if we consider the following. If there exists exactly one train line l_1 with frequency equal to 1, while all other frequencies have a greatest common divisor $g > 1$, then the optimal timetable is equivalent to also l_1 having frequency g. This is because the whole timetable is symmetric to a time shift of t/g, except l_1, therefore given a stable timetable, line l_1 can be multiplied to all other $g-1$ time slots. To take advantage of this symmetry, we first increase the frequency of l_1 to g, calculate the timetable taking advantage of the greatest common divisor reduction, and finally remove $g-1$ instances of l_1 to get back to the required frequency of 1.

4.4. Redundant infrastructure constraints

If after the previous two steps there still is a set of train lines with equal frequency that is larger than one, it is possible to remove a substantial amount of infrastructure constraints. Take each pair, e.g. lines l_1 and l_2, and while keeping all constraints between the first run of l_1 and any run of l_2, remove all constraints between any further runs of l_1 and l_2. Formally, let us define any total strict ordering $\prec$ of the train lines l_i (e.g. based on the ordering of the line numbers), and then for each pair of train lines $l_1 \prec l_2$ with $1 < F_{l_1} = F_{l_2}$, remove all infrastructure constraints for event pairs (i, j) where $Line_i = l_1,\ Line_j = l_2$ and $RunNr_i > 1$ (recall that F_l is the frequency of line l and $RunNr$ is an attribute of an event referring to the run number which can be larger than one for frequencies larger than one). An example is shown in Fig. 6. Note that because of the symmetry of the train lines, this reduced set will ensure that all trains are separated accordingly: for example, in Fig. 6, if there was an infrastructure conflict between the second run of line i and a run of line j, because of the symmetry enforced by the regularity constraints this would also mean an infrastructure conflict between the first run of line i and a run of line j, which is not possible as by definition we did not remove any infrastructure constraints related to the first run.

4.5. Symmetry-breaking constraints

For many line patterns there are a larger number of symmetric solutions possible, such as two solutions that only differ in a shift of all events in time or swapping the first and the second run of a train line with frequency 2. The optimization run can be speeded up substantially by “breaking” these symmetries and fixing one of many symmetrical choices. Therefore these constraints are also called *symmetry-breaking constraints* (Liberti, 2008).

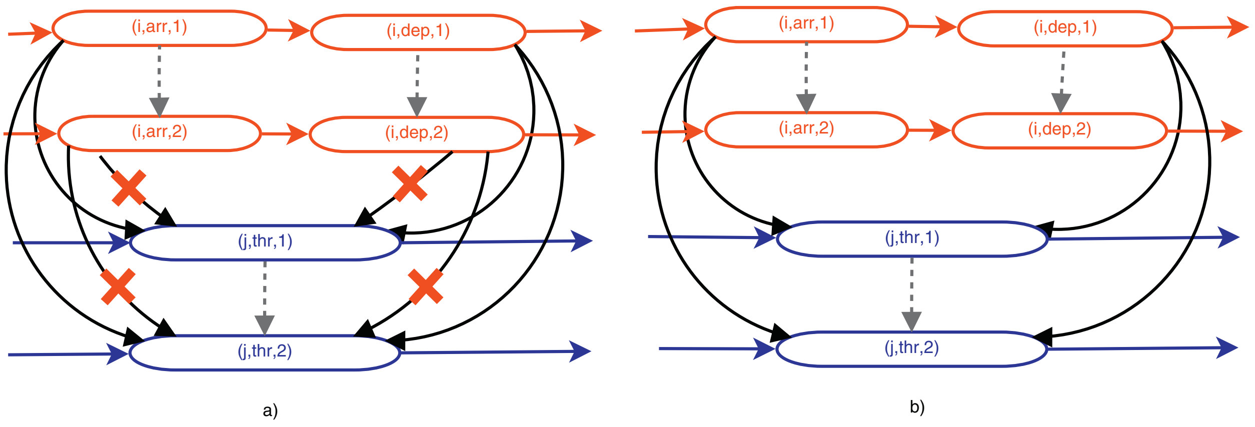


Fig. 6. Example for removal of redundant infrastructure constraints: with a red cross overlay on panel (a) and removed in panel (b) (the regularity constraints are shown as dotted gray lines, station id is not shown in nodes for clarity.) (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

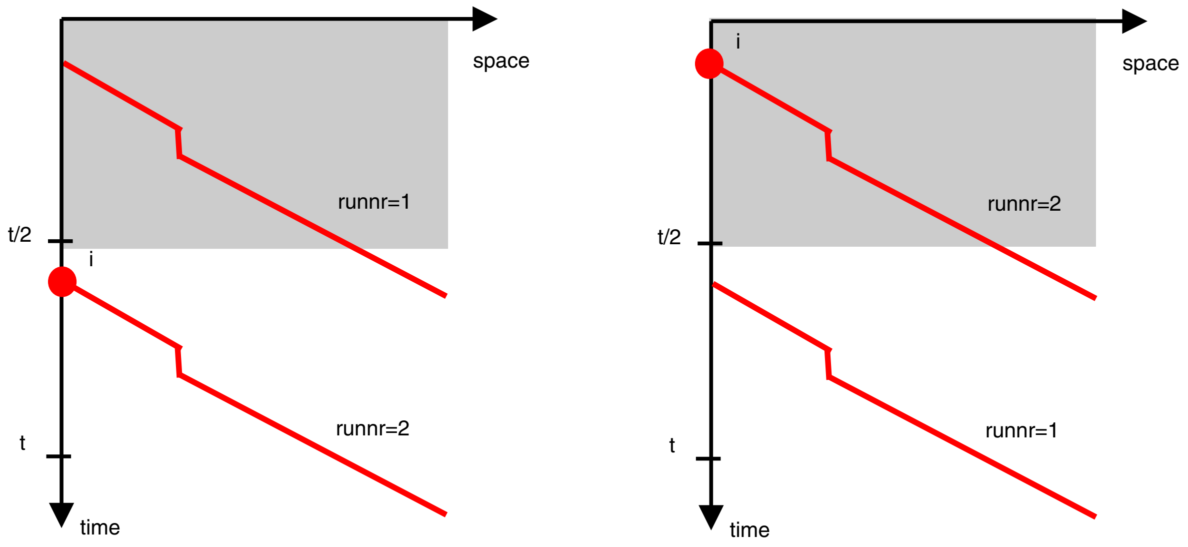


Fig. 7. Two timetables that only differ in their run numbers, to illustrate symmetry-breaking constraints. (For interpretation of the references to colr in this figure, the reader is referred to the web version of this article.)

Two constraints are added to avoid multiple practically identical solutions. Firstly, if two solutions are identical except a uniform shift in time for all event times, they also represent timetables with identical characteristics and minimum cycle time. Therefore, it is possible without loss of generality to choose a single event and fix its event time. Therefore let *init* be this one initial event, and we set $\tau_{\text{init}} = 0$.

The other constraint considers train lines with frequencies larger than one. In this case, a new timetable created by shifting the event times of this line with a multiple of the headway time t/F is identical to the original timetable. Therefore if $\mathcal{F}$ is a set containing a single (arbitrary) event for each train line and F_i is the frequency of the line of event i,

$$\tau_i < \tfrac{t}{F_i} \qquad \forall i \in \mathcal{F}. \tag{37}$$

Similarly to constraint (12), the strict inequality can be replaced by a non-strict one by replacing t with $t - \delta$. Note that there is no need to restrict constraint (37) to the lines with frequencies larger than one: if $F_i = 1$ then this constraint degenerates to the existing (12).

For example, on Fig. 7, the timetables on the left and the right of the red train with frequency 2 are identical, except that the run numbers are shifted. If we require an arbitrary point i to have an event time less than t/F_i, in this case $t/2$, then only the right side version stays valid and the solution space is reduced.

4.6. Marking limits

We call the number of times a process (i, j) crosses the cycle time boundary its *marking*, equal to the related variable z_{ij}. As we earlier ensured that all processes are shorter than the cycle time, a marking is in {0, 1}. In practice, as the majority of dwell and run activities are substantially shorter than the cycle time, this means that a chain of run-dwell processes of a certain train line crosses the cycle time boundary much less often than the number of its arcs, meaning that the majority of the arcs have marking zero. To capture this observation, we can define valid inequalities for the sum of markings for each train line by calculating the sum of minimum and maximum process durations and comparing them to the maximum and minimum cycle time bound, respectively. More concretely, recall the earlier definition of $\mathcal{A}_l$ (33) for the set of all run and dwell times of the first run of line *l*, and *Lines* as the set of all lines. Note that if a process (or chain of processes) has duration *d* and the cycle time is *T*, then the number of times that this process can cross the cycle time boundary is within $[\lfloor d/T \rfloor, \lceil d/T \rceil]$. Applying this for variable durations and cycle time, for the first run of all lines (note that second and further runs are not necessary to include as they are constrained by the regularity constraints):

$$\left\lfloor \sum_{(i,j)\in\mathcal{A}_l} L_{ij}/U \right\rfloor \le \sum_{(i,j)\in\mathcal{A}_l} z_{ij} \le \left\lceil \sum_{(i,j)\in\mathcal{A}_l} U_{ij}/L \right\rceil \forall l \in Lines$$

Note that the effectiveness of this reduction technique is closely related to the size of the cycle time bounds defined.

4.7. Run and dwell durations as secondary objective

While the main objective of our model is minimizing the cycle time, in general many different timetable versions can exist with equal train orders and equal minimum cycle time, but slightly different running and dwell times. To break this symmetry, we introduce the sum of running and dwell times in the objective function as a secondary objective (with an appropriately small weight). Note that this reduction method has the practical advantage of making the model choose a timetable with the lowest time allowances among multiple timetables with identical minimum cycle time.

5. An iterative solution approach

In the previous section we have seen that the presented the solving techniques for the optimization model that is able to calculate, as we will see in the computational results, a stable intercity timetable for the whole Dutch network with flexible train orders and running and dwell times. This model solution approach, however, might not always be suitable to calculate larger timetable instances, such as a large regional timetable with both local and intercity trains, in one optimization solver run in a suitable time. Therefore, in this section we present an iterative method to calculate large stable timetables very fast, in just a few seconds, and optimize these timetables for stability in a few hours.

The key idea of this method is that while the cycle time *t* is a variable, nevertheless it is bounded by the predefined limits (*L*, *U*) and the smaller this range is, the faster the optimization solver can advance within the reduced search space. Hence, we introduce an iterative calculation where the cycle time range (*L*, *U*) is periodically adjusted and the calculation resumed.

We take advantage of the fact that many MILP solvers, including CPLEX, allow a definition of an initial feasible set during the declaration of the MILP problem, that guarantees feasibility and may further speed up the optimization process: we use the intermediate solution of one solve iteration to initialize the following iteration. We define the function name $MILPSolve(\mathcal{E}, \mathcal{A}, L, U, k)$ that represents solving the MILP problem defined earlier given the events $\mathcal{E}$ and activities $\mathcal{A}$, as well as the cycle time bounds (*L*, *U*), and the optional variable *k* containing an intermediate solution.

Algorithm 1 describes the structure of the iterative process. Given the timetable input values $(\mathcal{E}, \mathcal{A})$, the nominal cycle time *T*, and the iteration configuration parameters $F > 1$, $0 < g_{min} < g_{init} < 1$, we first run *MILPSolve*() with fixed cycle time $t = L = U = T$ to obtain a feasible solution. Then the range (*L*, *U*) is initialized such that $L < t = U$ according to the initial relative gap parameter g_{init}. Consequently, we iteratively run the solver *MILPSolve*() until optimality within the bounds, or until a timeout is reached since the last change in the objective value. The bounds are re-set after each iteration according to the factor *F*: if optimality was found at the lower bound then the gap $(U - L)/U$ is increased by a factor of *F*, and if new solutions were found and the cycle time variable is still at the current upper bound then the gap is decreased by a factor of *F*. And otherwise the relative gap does not change. In all three cases, the bounds are re-set as $U = t$ and *L* according to the defined gap. The iterations stop when optimality is reached or the relative difference between *L* and *U* reached the minimum relative gap g_{min} when we return the intermediate solution. We used the values $F = 2$, $g_{init} = 0.01$, and $g_{min} = 0.001$. For the timeout of *MILPSolve*() we used the internal iteration counter of CPLEX and the timeout value of 800,000 CPLEX iterations, which is approximately between 2–4 min depending on the problem size. The reason we used a timeout based on the iteration count and not on clock time is that this way the solver runs are fully reproducible. As timeouts are used, optimality is not guaranteed in this iterative algorithm, but for the majority of real-life examples we considered, it is actually reaching optimality faster than the non-iterative method.

For obtaining an attractive calculation time, it is instrumental that the reset of the cycle time bounds (*L*, *U*) is performed in an adaptive fashion. In particular, if the difference $U - L$ between the cycle time bounds is too large, a single iteration can take too long time, or reach timeout. On the other hand, if the cycle time bound is too small, the number of iterations can be too high, again causing a long total calculation time. Therefore we define the adaptive *ResetBounds*() function as described in Algorithm 2. As a default rule, we set $L := (1 - g)t$ and $U := t$ where *g* is the relative gap of the previous run. However, if the previous run was perceived too fast (*t* actually reached the lower bound *L*), or too slow (*t* did not improve from $t = U$ until timeout), we increase or decrease relative gap g, respectively, using the relative gap factor variable *F* set as a parameter of the function.

Algorithm 1 Iteratively optimize timetable.

```
function IterativelySolve($\mathcal{E}$, $\mathcal{A}$, $T$, $F$, $g_{init}$, $g_{min}$)
    k ← MILPSolve($\mathcal{E}$, $\mathcal{A}$, $T$, $T$) // initial solution for solver
    $U \leftarrow T$
    $L \leftarrow (1 - g_{init})T$
    while $g_{min} < (U - L)/U$ do
        (SolverStatus, $t$, $k$) ← MILPSolve($\mathcal{E}$, $\mathcal{A}$, $L$, $U$, $k$)
        if SolverStatus = 'Optimal' and $L < t$ then
            return ($t$, $k$) // optimal solution
        ($L$, $U$) ← ResetBounds($L$, $t$, $U$, $F$)
    return ($t$, $k$) // intermediate solution
```

Algorithm 2 Reset search bounds given intermediate solution.

```
function ResetBounds($L$, $t$, $U$, $F$)
    $g \leftarrow (U - L)/U$ // gap
    if $L = t$ then // optimality within bounds: increase relative gap
        $g \leftarrow F * g$
    else if $t = U$ then // no new solutions: decrease relative gap
        $g \leftarrow g/F$
    $L \leftarrow (1 - g) * t$
    $U \leftarrow t$
    return ($L$, $U$)
```

6. Computational results

We tested the optimization on seven network scenarios of increasing size, using all the dimension techniques implemented from Section 4, first without and then with the iterative approach included. All data is from the timetable database used by the timetable stability evaluation tool PETER (Goverde, 2007) that is in turn based on the DONS database (Hooghiemstra and Teunisse, 1998) describing the 2007 Dutch periodic timetable for the morning peak. In the PETER timetable file, the national Dutch hourly timetable is included in the detail of stations, junctions and other timetable points together with the minimum running and dwell times, as well as the default minimum infrastructure headways between all train event pairs sharing the same infrastructure.

Table 2 contains a detailed description of the used data format. Note that as we store process durations and the calculated event times as continuous (floating point) variables, and treated as real values by the optimization model, the model supports minimum process time constraints of seconds precision. We included all constraints and minimum process times

Table 2
List of input data types used from the DONS/PETER timetable format.

Entity	Attribute	Example	Note
Cycle time	Cycle time	1 h	Stored as floating point
Station	*Station Abbr.*	Asdz	
	Station Type	IC	Intercity station, other station or other timetable point
	Station Name	Amsterdam Zuid	For visualization purposes
	Longitude	4.873337603	Retrieved from OpenOV[a]
	Latitude	52.33886371	Retrieved from OpenOV
Line	*Line ID*	270	Stored as integer
	Direction	0	0 or 1
	Line type	IC	Intercity or local train
	Nr. of runs	2	Per cycle time, stored as integer
Event	*Event ID*	l270_r2_d0_arr_Asdz	Generated unique identifier from the other variables
	Line ID	270	Matches *Line ID* of *Line*
	Direction	0	Matches *Direction* of *Line*
	Run Number	2	Integer between 0 and Nr. of runs of *Line*
	Event Type	*arr*	*dep*, *arr* or *thr*
	Station Abbr.	Asdz	Matches *Station Abbr.* of *Station*
Process	*From Event ID*	l270_r2_d0_arr_Asdz	Matches *Event ID* of *Event*
	To Event ID	l270_r2_d0_dep_Asdz	Matches *Event ID* of *Event*
	Process Type	*dwell*	*run*, *dwell* or *infra*
	Min. Duration	60 s	Stored as floating point, i.e. at least seconds precision

[a] Stichting OpenGeo (2013).

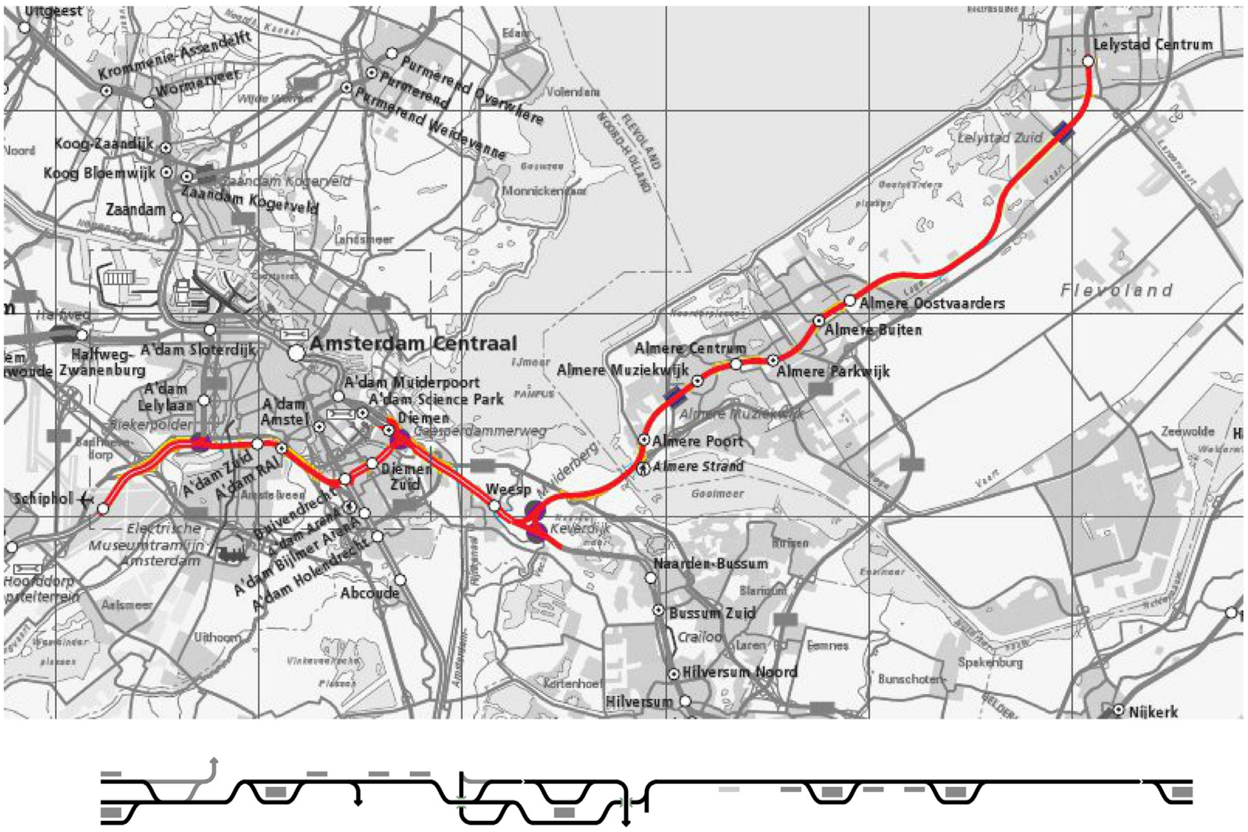


Fig. 8. The SAAL corridor on a geographic map and its schematic track layout in the West-East direction (thick red line: two tracks, double red line: four tracks, Map source: ProRail). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

for run, dwell, and infrastructure headways from the PETER timetable file, however we did not include connection constraints to ensure flexibility in the train orders. For the minimum headway times of infrastructure constraints we used the normative default values (such as three minutes) as only these norms were given.

During the preprocessing, the network is filtered to the desired network area and subset of lines, as we describe the scenarios later. Maximum durations of run processes are set as $U_{ij} = 1.3L_{ij} + 20$ s for each run process (i, j) where L_{ij} is the minimum duration as read from the PETER file and the bounds are expressed in seconds. The cycle time for the scenarios based on this timetable was 60 min, reduced to 30 min in the preprocessing according to the greatest common divisor of frequencies preprocessing step described in Section 4, as all lines included in the scenarios have a 30 min symmetry. We also verify at this time whether for all line pairs using the same station or pair of stations illegal overtakings at a station or between stations are forbidden by the existing constraints, and if not, the dummy nodes are added as described in Section 3.5, both for too large dwell times and too large running times. The preprocessing of converting the PETER files to an input file to the MILP solver takes at most a few minutes on a standard PC.

6.1. Network scenarios

The first scenario consists of the single railway corridor Schiphol-Amsterdam-Almere-Lelystad ("SAAL"), as shown on a geographic map and as a schematic track layout in Fig. 8. This railway line consists of two-track and four-track sections, all unidirectional, and two junctions with flyovers at Diemen and Weesp as pictured. Including all local and intercity train lines on this corridor, the model consists of 18 train lines and 13 stations, that translate to 156 event nodes and 455 activity arcs in the periodic event-activity network. Recall that the reason for the high number of event nodes is that the model includes events not just at stations but also at other timetable points, such as junctions, where the separation of trains is necessary.

The further scenarios are including networks larger than the SAAL corridor pictured in Fig. 8. The scenario "AMS-R" includes the full length trips of all local trains stopping at Amsterdam Centraal. The scenario "UT-ASS-IC" includes the full length trips of all intercity trains stopping at Utrecht Centraal or Amsterdam Sloterdijk. The scenario "NL-IC" includes the full trips of all Intercity trains in the Netherlands. Scenario "UT" consists of the full trips of all intercity and local trains calling at Utrecht Centraal station. Scenario "NL-IC+SAAL" is a combination of the earlier two scenarios of the same name. Finally, scenario "NVG" consists of all local and intercity trains running in the large urban region (*Noordvleugel*, "North Wing")

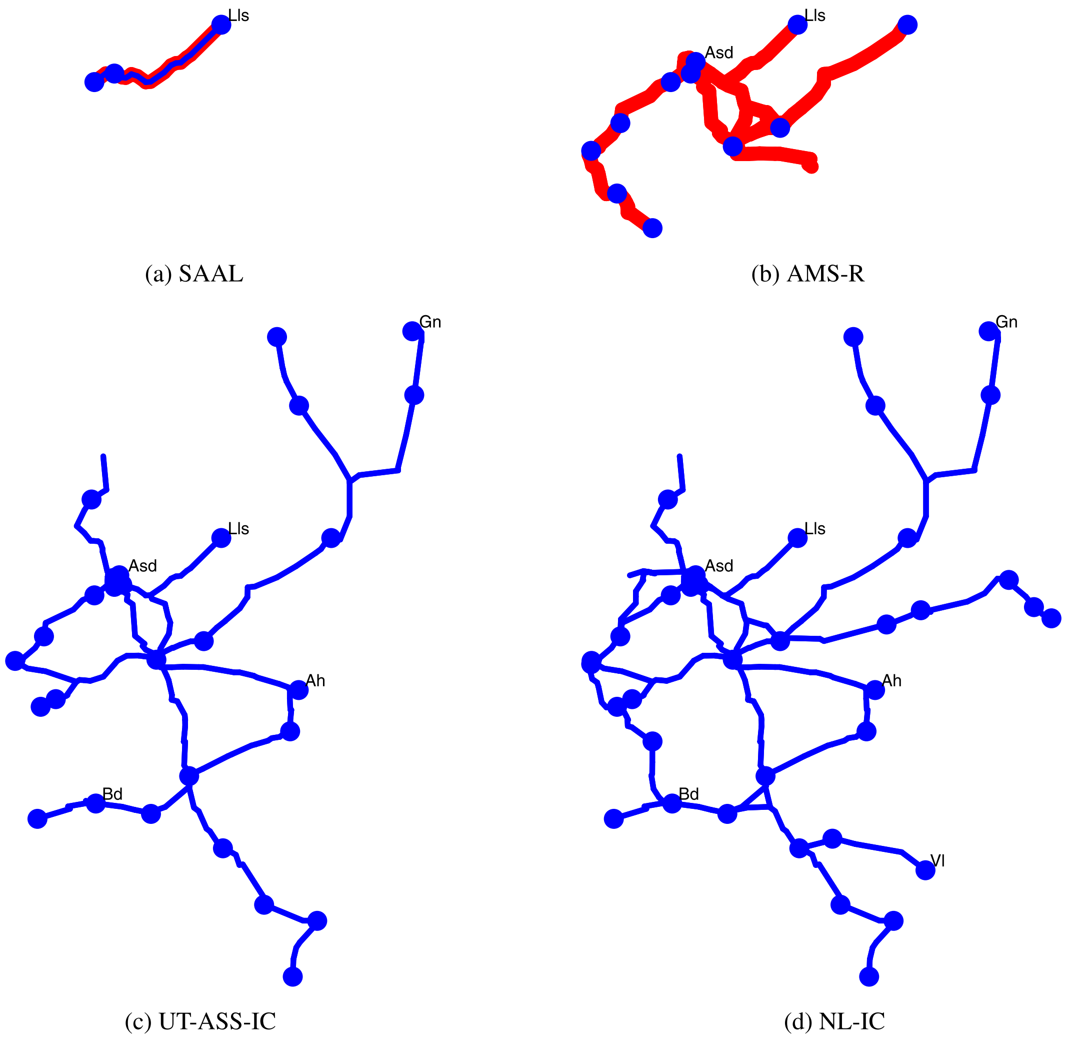


Fig. 9. Maps of the different scenarios (blue – intercity train lines and intercity stations, red – local train lines, "Asd" – Amsterdam Centraal, "Lls" – Lelystad Centrum, "Bd" – Breda, "Ah" — Arnhem Centraal, "Vl" – Venlo, "Gn" – Groningen). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

around Amsterdam, bounded by stations Leiden Centraal, Gouda, Utrecht Centraal, Amersfoort and Lelystad Centrum, with the trip segments only within this region. See Figs. 9 and 10 for a comparative map of all seven scenarios, and see Table 3 for the number of lines, stations, and periodic event-activity network dimensions of all scenarios.

6.2. Dimension reduction techniques without iterative run

For solving the MILP problem we used IBM ILOG CPLEX version 12.4 on a generic PC with 12 GB RAM and a six-core 3.47 GHz CPU with all cores used during the CPLEX run. Fig. 11 shows the optimization results for the SAAL corridor in the West–East direction. On the compressed timetable, an hourly pattern compressed into the cycle time of 36 min and 36 s is pictured, and the critical infrastructure headways are pictured (black dotted lines) that determine the minimum cycle time. The expanded timetable represents a stable hourly timetable with minimal cycle time. As mentioned earlier, the expanded timetable can be further optimized with another optimization model using fixed train orders and cycle time, if necessary.

The results of the other three scenarios and calculation times for all scenarios are reported in Table 3. All preprocessing techniques from Section 4 were used, but without the iterative approach in Section 5. We can conclude that the optimization model finds the optimal solution in a few minutes for smaller networks represented by event-activity networks with less than a thousand arcs or nodes. However, one of the two larger intercity networks takes 22 min to run and this model also proved ineffective in quickly calculating stable timetables for larger scenarios. Therefore we below also use the iterative method of Section 5 to address this scalability problem.

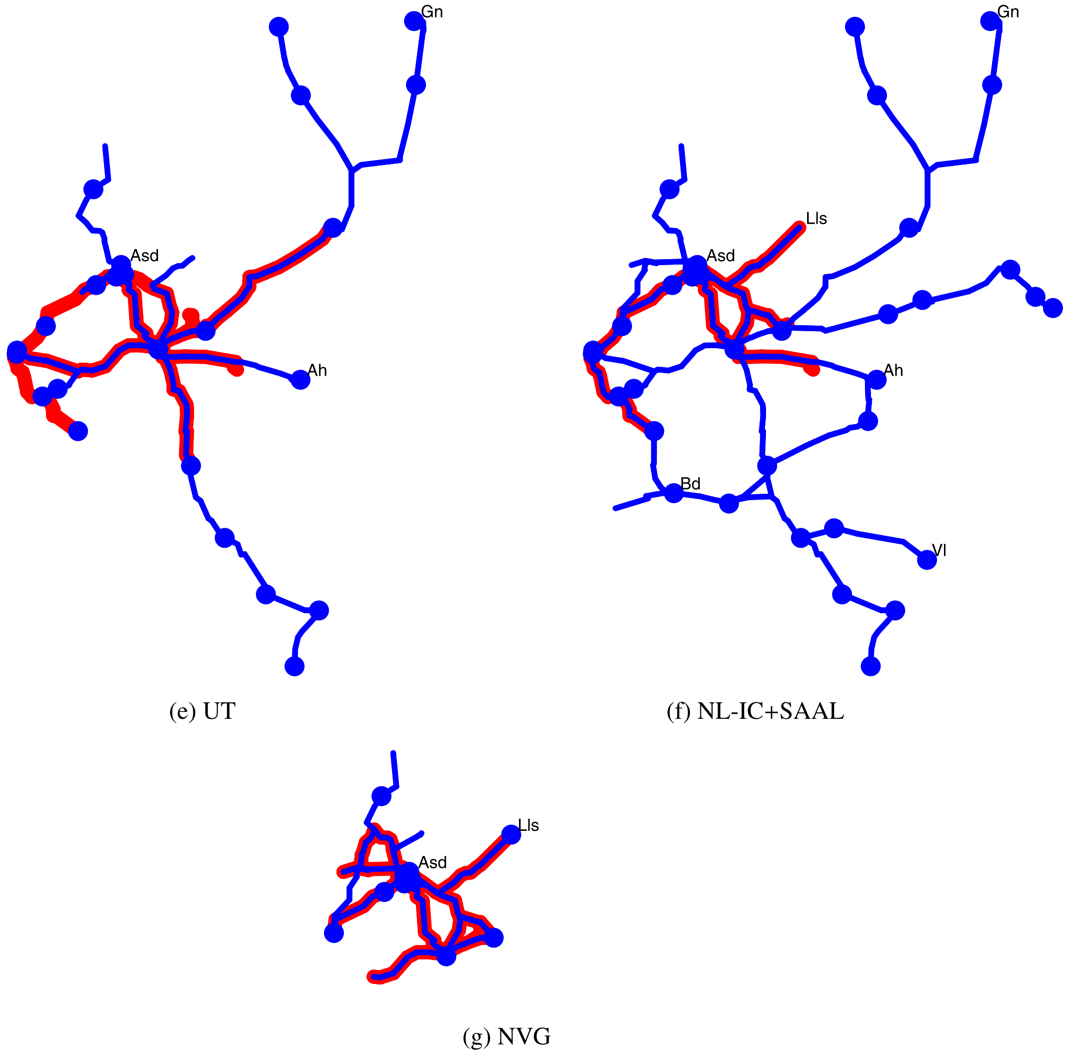


Fig. 10. Maps of further different scenarios (blue – intercity train lines and intercity stations, red – local train lines, "Asd" – Amsterdam Centraal, "Lls"—Lelystad Centrum, "Bd" – Breda, "Ah" – Arnhem Centraal, "Vl" – Venlo, "Gn" – Groningen). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

Table 3
Graph size for different scenarios and optimal timetable calculation times.

Scenario	Lines	Trains	Stations	Nodes	Arcs	Solution time	$^t/_T$
SAAL	18	72	13	156	455	00:00:45	61.0%
AMS-R	16	64	42	605	791	00:02:30	51.4%
UT-ASS-IC	20	80	142	1033	1473	00:22:33	40.0%
NL-IC	30	120	193	1515	2175	00:05:12	40.0%
UT	28	112	140	1352	2552	04:00:00<	N/A
NL-IC+SAAL	35	140	195	1837	3103	04:00:00<	N/A
NVG	65	260	82	1594	3917	04:00:00<	N/A

6.3. Iterative processing

Using the iterative process of Section 5, we were able to extend the scope of our optimization model to quickly calculate stable timetables of large regions with mixed traffic. Fig. 12 shows the expanded timetables of two representative corridors of the NVG scenario, in both directions. To illustrate the effect of the iterative calculation with varying cycle time ranges, Fig. 13 shows the objective value of the optimization solve runs as functions of the calculation time. Fig. 13(b) illustrates the mechanism of the iterative calculation the best: the long horizontal lines of the objective value signify situations when the

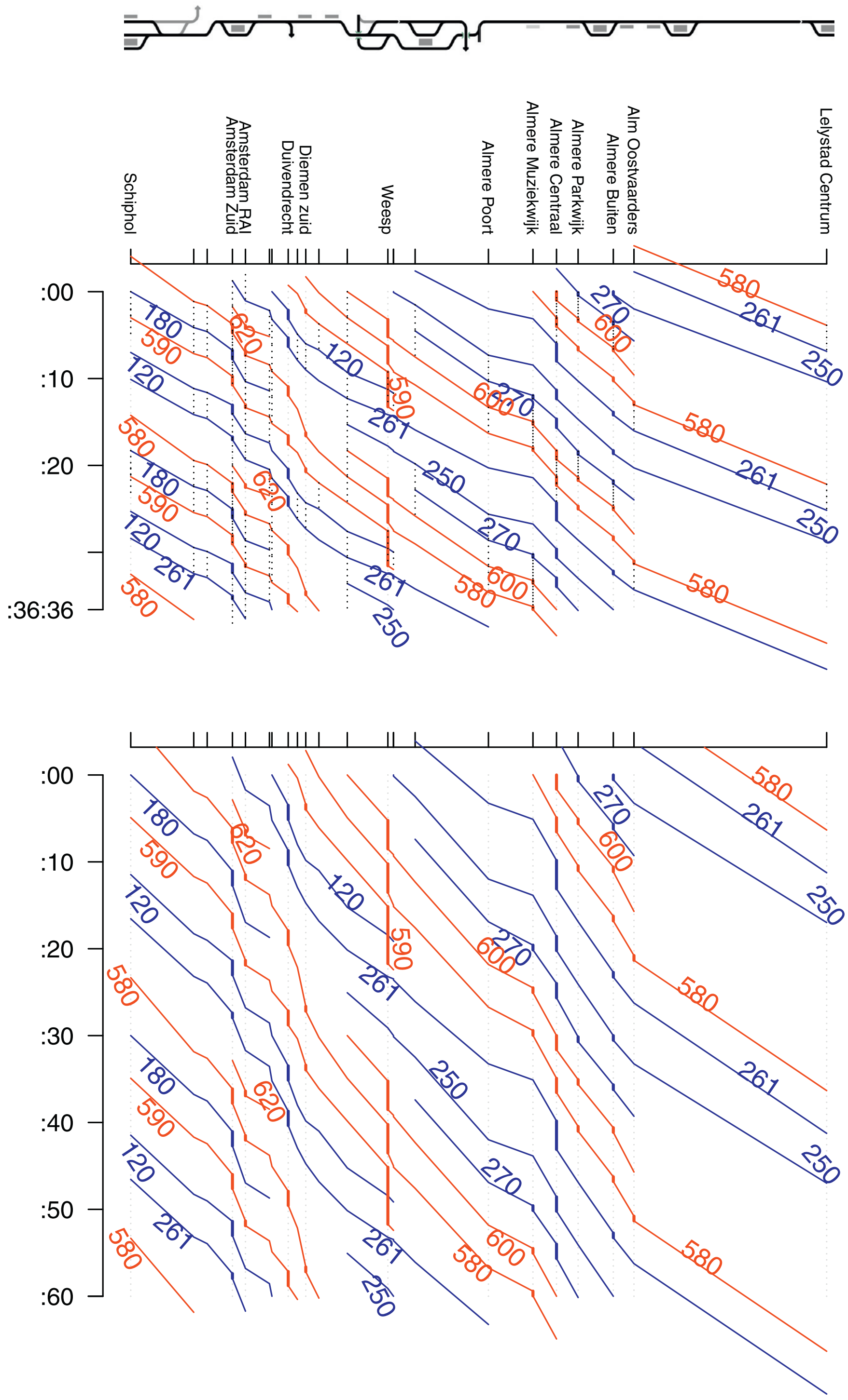


Fig. 11. Schematic track layout (top), optimal compressed (middle) and expanded (bottom) timetable for the SAAL corridor (one direction pictured, red: local trains, blue: intercity trains, black dotted lines: critical infrastructure constraints). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

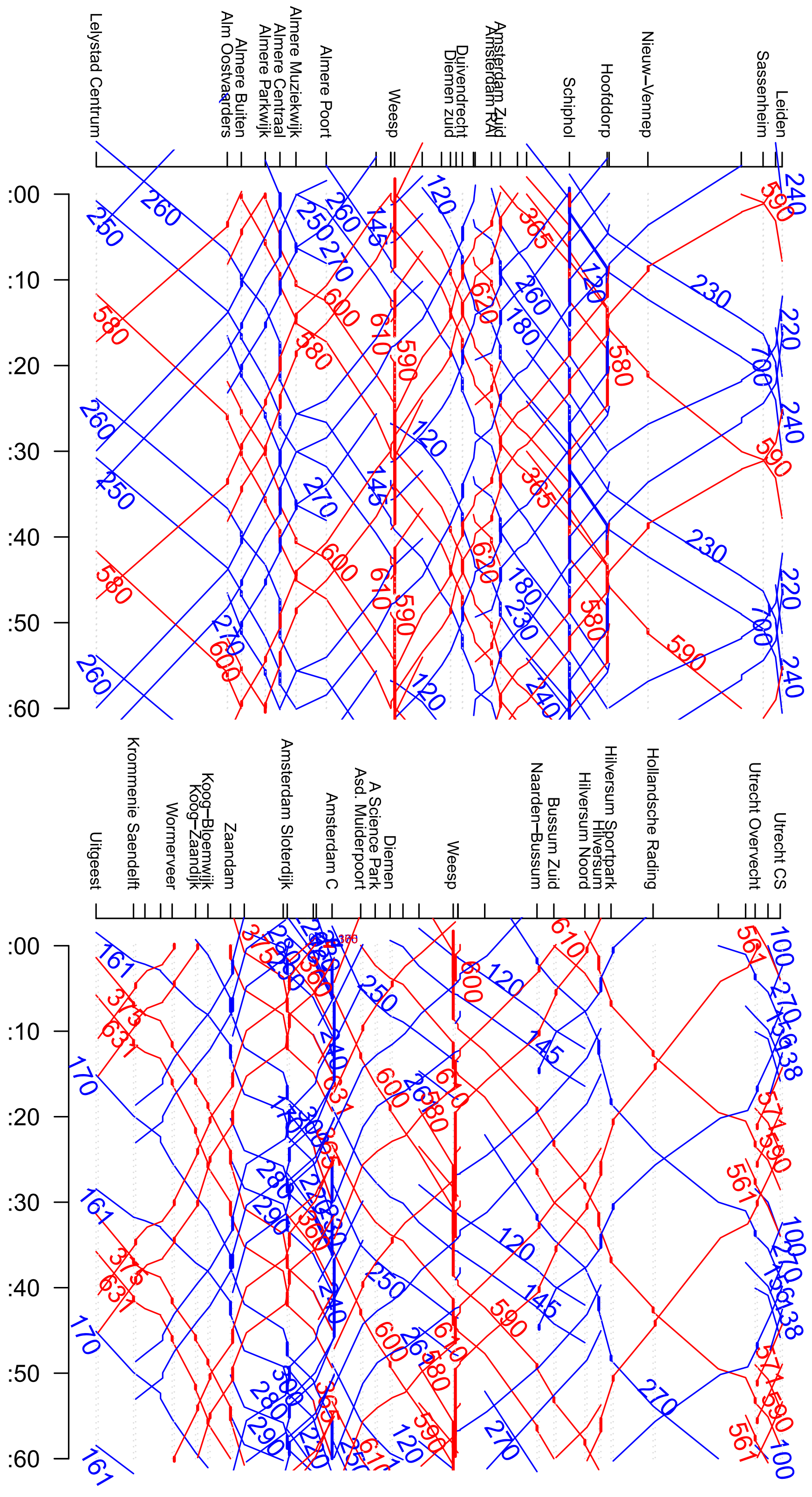
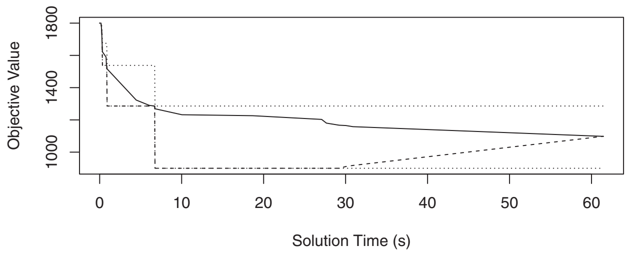
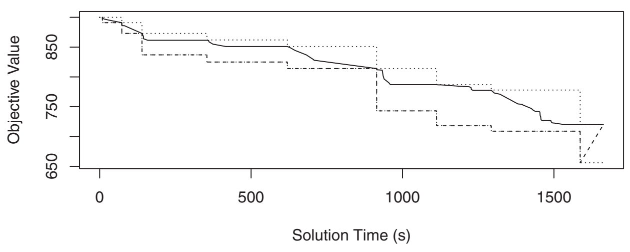


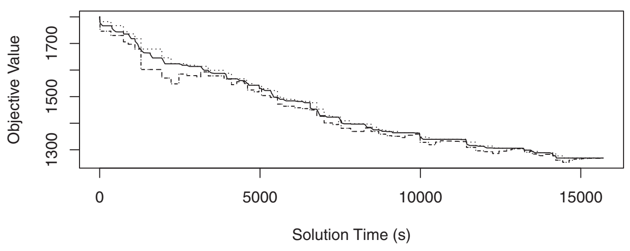
Fig. 12. Optimized expanded timetable of the NVG scenario, Leiden-Lelystad Centrum (above) and Uitgeest-Utrecht (below) corridor.



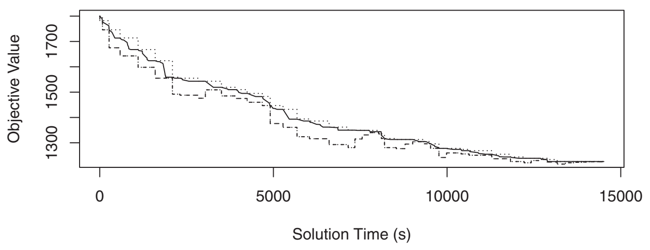
(a) Scenario SAAL



(b) Scenario NL-IC



(c) Scenario UT



(d) Scenario NVG

Fig. 13. Intermediate objective value and its bounds during the solver run for scenarios SAAL, and NL-IC, UT and NVG (dotted lines – objective bounds, dashed line – LP bound, continuous line: intermediate objective value).

optimization solver was unable to improve the solution until a timeout. After a re-set of the objective bounds, however, the solver was able to continue and find optimality.

Table 4 shows calculation times for all scenarios both using the basic solve method seen in the previous section and the current iterative solve. All preprocessing techniques from Section 4 were used both in the basic and the iterative case. In the latter, we make the distinction between the calculation time until a stable timetable and to optimality. In some cases, it was not possible to prove mathematical optimality before timeout, in these cases calculation time to the best attained value is reported.

Figs. 14 and 15 plots calculation times to feasibility and optimality on a logarithmic scale as a function of the size of the graph. Note that for the three largest cases, only the iterative approach is able to find a stable and optimal timetable, this is represented by the three points of the basic solution method in the upper right corner of the charts corresponding to the "4h< " mark of the vertical axis. We can conclude that the iterative solution method enables a significant reduction in computation times and allows stable timetable calculations of large mixed traffic networks in a few minutes, and stability optimization in a few hours.

Table 4
Stable and optimal timetable calculation times with the basic and the iterative method for different scenarios.

	Basic solve	Iterative solve			
Scenario	Optimal	Stable	Best	Iterations	$^t/_T$ (%)
SAAL	00:00:45	00:00:00	00:01:01[a]	7	61.0
AMS-R	00:02:30	00:02:02	00:02:02[a]	7	51.4
UT-ASS-IC	00:22:33	00:00:06	00:05:09[a]	8	40.0
NL-IC	00:05:12	00:00:08	00:25:34[a]	10	40.0
UT	04:00:00<	00:00:13	03:57:30	72	70.5[a]
NL-IC+SAAL	04:00:00<	00:00:11	01:47:39	27	61.0[a]
NVG	04:00:00<	00:00:02	03:35:00	42	68.1[a]

[a] In these cases the best obtained timetable is also proven to be optimal.

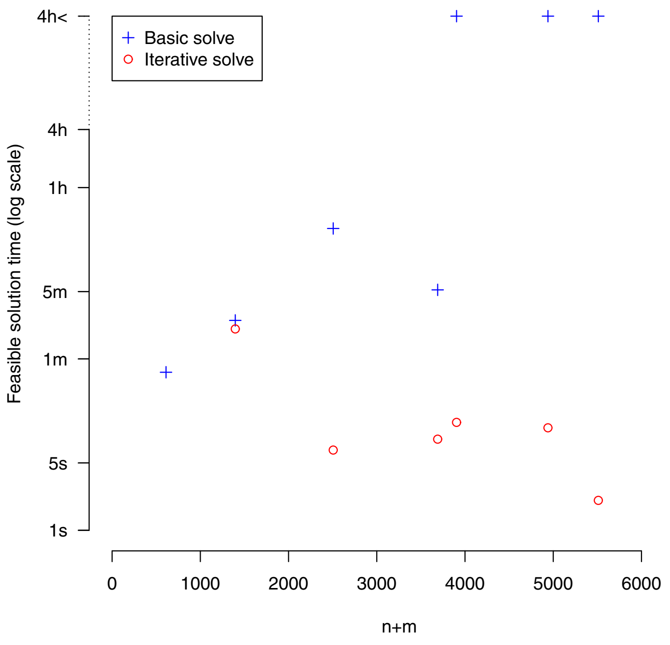


Fig. 14. Calculation time of a stable timetable as a function of the number of nodes and arcs in the event-activity network.

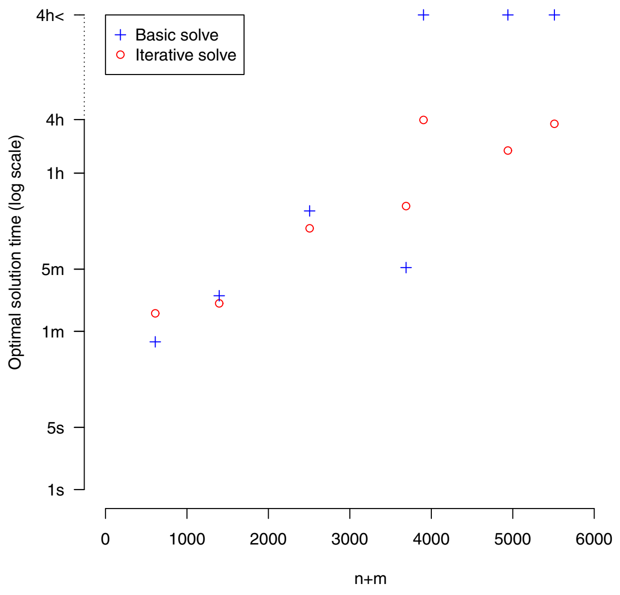


Fig. 15. Calculation time of an optimal timetable as a function of the number of nodes and arcs in the event-activity network (optimum not proven in the largest 3 cases).

7. Conclusions

We presented a periodic railway timetable optimization model that can handle flexible train orders, running and dwell times, and uses a measure of timetable stability as the objective function. We introduced several dimension reduction techniques, as well as an iterative optimization approach, to ensure that the model is applicable to large mixed traffic networks. Using available timetable planning data for the Dutch national railway network, we applied the optimization model to several networks of different sizes and train classes to illustrate the usefulness of the model.

The results show that our approach is able to calculate a stable timetable for a large mixed traffic network in just a few seconds, and optimize the timetable for stability in 2–3 h calculation time on a generic computer. This makes the model highly suitable both for quick comparison of many different timetable scenarios of different frequencies and line patterns, and for improving existing timetables to increase their stability by the appropriate reordering of trains and reallocation of buffer times according to the stability-oriented optimization.

References

Bergmann, D.R., 1975. Integer programming formulation for deriving minimum dispatch intervals on a guideway accommodating through and local public transportation services. Transp. Plan. Technol. 3 (1), 27–30.

Besinovic, N., Goverde, R.M.P., Quaglietta, E., Roberti, R., 2016. An integrated micro-macro approach to robust railway timetabling. Transp. Res. Part B 87, 14–32.

Braker, J.G., 1993. Algorithms and Applications in Timed Discrete Event Systems. Ph.D. thesis, Delft University of Technology.

Burdett, R., Kozan, E., 2015. Techniques to effectively buffer schedules in the face of uncertainties. Comput. Ind. Eng. 87 (C), 16–29. doi:10.1016/j.cie.2015.04.024.

Cacchiani, V., Caprara, A., Fischetti, M., 2012. A Lagrangian heuristic for robustness, with an application to train timetabling. Transp. Sci. 46 (1), 124–133.

Cacchiani, V., Caprara, A., Toth, P., 2010. Scheduling extra freight trains on railway networks. Transp. Res. Part B 44 (2), 215–231.

Cacchiani, V., Toth, P., 2012. Nominal and robust train timetabling problems. Eur. J. Oper. Res. 219 (3), 727–737.

Caimi, G., Burkolter, D., Herrmann, T., Chudak, F., Laumanns, M., 2008. Design of a railway scheduling model for dense services. Netw. Spat. Econ. 9 (1), 25–46.

Caprara, A., Galli, L., Toth, P., 2011. Solution of the train platforming problem. Transp. Sci. 45 (2), 246–257.

De Kort, A., Heidergott, B., Ayhan, H., 2003. A probabilistic (max, +) approach for determining railway infrastructure capacity. Eur. J. Oper. Res. 148 (3), 644–661.

Dollevoet, T., Huisman, D., Schmidt, M., Schöbel, A., 2012. Delay management with rerouting of passengers. Transp. Sci. 46 (1), 74–89.

Fischetti, M., Monaci, M., 2009. Light robustness. In: Ahuja, R.K., Möhrung, R.H., Zaroliagis, C.D. (Eds.), Robust and Online Large-Scale Optimization, LNCS 5868. Springer, Berlin, pp. 61–84.

Fischetti, M., Salvagnin, D., Zanette, a., 2009. Fast approaches to improve the robustness of a railway timetable. Transp. Sci. 43 (3), 321–335.

Goverde, R.M.P., 1999. Improving punctuality and transfer reliability by railway timetable optimization. In: Proceedings of the 5th TRAIL Annual Congress. TRAIL Research School, Delft.

Goverde, R.M.P., 2007. Railway timetable stability analysis using max-plus system theory. Transp. Res. Part B 41 (2), 179–201.

Goverde, R.M.P., 2010. A delay propagation algorithm for large-scale railway traffic networks. Transp. Res. Part C 18 (3), 269–287.

Goverde, R.M.P., 2014. Timetable stability analysis. In: Hansen, I.A., Pachl, J. (Eds.), Railway Timetabling & Operations. Eurailpress, Hamburg, pp. 133–153.

Goverde, R.M.P., Besinovic, N., Binder, A., Cacchiani, V., Quaglietta, E., Roberti, R., Toth, P., 2016. A three-level framework for performance-based railway timetabling. Transp. Res. Part C 67, 62–83.

Goverde, R.M.P., Hansen, I.A., 2013. Performance indicators for railway timetables. In: Proceedings of IEEE International Conference on Intelligent Rail Transportation (ICIRT), Beijing, China, pp. 301–306.

Großmann, P., Hölldobler, S., Manthey, N., Nachtigall, K., Opitz, J., Steinke, P., 2014. Solving periodic event scheduling problems with SAT. In: Jiang, H., Ding, W., Ali, M., Wu, X. (Eds.), Advanced Research in Applied Artificial Intelligence. Springer Berlin Heidelberg, pp. 166–175.

Heidergott, B., Olsder, G.J., van der Woude, J., 2005. Max Plus at Work: Modeling and Analysis of Synchronized Systems: A Course on Max-Plus Algebra and Its Applications. Princeton University Press.

Heydar, M., Petering, M.E.H., Bergmann, D.R., 2013. Mixed integer programming for minimizing the period of a cyclic railway timetable for a single track with two train types. Comput. Ind. Eng. 66 (1), 171–185.

Hooghiemstra, J.S., Teunisse, M.J.G., 1998. The use of simulation in the planning of the dutch railway services. In: Medeiros, D., Watson, E., Carson, J., Manivannan, M. (Eds.), Proceedings of the 1998 Winter Simulation Conference, pp. 1139–1145.

International Union of Railways (UIC), 2004. UIC Code 406: Capacity.

International Union of Railways (UIC), 2013. UIC Code 406: Capacity.

Kroon, L.G., Huisman, D., Abbink, E., Fioole, P.-J., Fischetti, M., Maróti, G., Schrijver, A., Steenbeek, A., Ybema, R., 2009. The new Dutch timetable: the OR revolution. Interfaces 39 (1), 6–17.

Kroon, L.G., Maróti, G., Helmrich, M., Vromans, M.J.C.M., Dekker, R., 2008. Stochastic improvement of cyclic railway timetables. Transp. Res. Part B 42 (6), 553–570.

Kroon, L.G., Peeters, L.W.P., 2003. A variable trip time model for cyclic railway timetabling. Transp. Sci. (1994).

Liberti, L., 2008. Automatic generation of symmetry-breaking constraints. In: Lecture Notes in Computer Science, 5165, pp. 328–338.

Liebchen, C., 2008. The first optimized railway timetable in practice. Transp. Sci. 42 (4), 420–435.

Liebchen, C., Lübbecke, M., Möhring, R., Stiller, S., 2009. The Concept of Recoverable Robustness, Linear Programming Recovery, and railway applications. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5868 LNCS, pp. 1–27. doi:10.1007/978-3-642-05465-5_1.

Liebchen, C., Schachtebeck, M., Schöbel, A., Stiller, S., Prigge, A., 2010. Computing delay resistant railway timetables. Comput. Oper. Res. 37 (5), 857–868.

Lindner, T., 2000. Train Schedule Optimization in Public Rail Transport. Technische Universität Braunschweig Ph.D. thesis.

Lindner, T., 2011. Applicability of the analytical UIC code 406 compression method for evaluating line and station capacity. J. Rail Transp. Plan. Manag. 1 (1), 49–57.

Lindner, T., Zimmermann, U.T., 2005. Cost optimal periodic train scheduling. Math. Methods Oper. Res. 62 (2), 281–295.

Lusby, R., Larsen, J., Ryan, D., Ehrgott, M., 2011. Routing trains through railway junctions: a new set-packing approach. Transp. Sci. 45 (2), 228–245.

Malcolm, D.G., Roseboom, J.H., Clark, C.E., Fazar, W., 1959. Application of a technique for research and development program evaluation. Oper. Res. 7 (5), 646–669.

Mussone, L., Wolfler Calvo, R., 2013. An analytical approach to calculate the capacity of a railway system. Eur. J. Oper. Res. 228 (1), 11–23.

Nachtigall, K., 1996. Periodic network optimization with different arc frequencies. Discret. Appl. Math. 69 (1–2), 1–17.

Nachtigall, K., Voget, S., 1996. A genetic algorithm approach to periodic railway synchronization. Comput. Oper. Res. 23 (5), 453–463.

Nachtigall, K., Voget, S., 1997. Minimizing waiting times in integrated fixed interval timetables by upgrading railway tracks. Eur. J. Oper. Res. 610–627.

Odijk, M.A., 1996. A constraint generation algorithm for the construction of periodic railway timetables. Transp. Res. Part B 30 (6), 455–464.

Petering, M.E.H., Heydar, M., Bergmann, D.R., 2016. Mixed-Integer programming for railway capacity analysis and cyclic, combined train timetabling and platforming. Transp. Sci. 50 (3), 892–909. doi:10.1287/trsc.2015.0652.

Schöbel, A., 2006. Optimization in Public Transportation: Stop Location, Delay Management and Tariff Planning from a Customer-Oriented Point of View, 3. Springer.

Schrijver, A., 1998. Routing and timetabling by topological search. Doc. Math. J. DMV Extra ICM III, 687–695.

Schrijver, A., Steenbeek, A., 1993. Spoorwegdienstregelingontwikkeling. Technical Report.

Schuele, I., Schroeder, M., Kuefer, K.H., 2009. Synchronization of regional public transport systems. In: Proceedings of International Conference on Urban Transport XV, 107, pp. 301–311.

Serafini, P., Ukovich, W., 1989. A mathematical model for periodic event scheduling problems. SIAM J. Discret. Math. 2 (4), 550–581.

Sparing, D., Goverde, R.M.P., 2013. An optimization model for periodic timetable generation with dynamic frequencies. In: Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2013). The Hague, The Netherlands, pp. 785–790.

Stichting OpenGeo, 2013. OpenOV.

Williams, H.P., 1990. Model Building in Mathematical Programming Fourth Edition, 3rd ed. John Wiley & Sons.

Wong, R.C.W., Yuen, T.W.Y., Fung, K.W., Leung, J.M.Y., 2008. Optimizing timetable synchronization for rail mass transit. Transp. Sci. 42 (1), 57–69.