Analysis of Strengths and Weaknesses of a MILP Model for Revising Railway Traffic Timetables*

Fahimeh Khoshniyat[1] and Johanna Törnquist Krasemann[2]

1 **Department of Science and Technology, Linköping University, Norrköping, Sweden**
`fahimeh.khoshniyat@liu.se`

2 **Department of Computer Science and Engineering, Blekinge Institute of Technology, Karlskrona, Sweden**
`johanna.tornquist.krasemann@liu.se`

Abstract

A railway timetable is typically planned one year in advance, but may be revised several times prior to the time of operation in order to accommodate on-demand slot requests for inserting additional trains and network maintenance. Revising timetables is a computationally demanding task, given the many dependencies and details to consider. In this paper, we focus on the potential of using optimization-based scheduling approach for revising train timetables during short term planning, from one week to few hours before the actual operation. The approach relies on a MILP (Mixed Integer Linear Program) model which is solved by using the commercial solver Gurobi.

In a previous experimental study, the MILP approach was used to revise a significant part of the annual timetable for a sub-network in Southern Sweden to insert additional trains and allocate time slots for urgent maintenance. The results showed that the proposed MILP approach in many cases generates feasible, good solutions rather fast. However, proving optimality was in several cases time-consuming, especially for larger problems. Thus, there is a need to investigate and develop strategies to improve the computational performance. In this paper, we present results from a study, where a number of valid inequalities has been selected and applied to the MILP model with the aim to reduce the computation time. The experimental evaluation of the selected valid inequalities showed that although they can provide a slight improvement with respect to computation time, they are also weakening the LP relaxation of the model.

1998 ACM Subject Classification G.1.6 Optimization

Keywords and phrases Railway, Timetable, Short term planning, Boosting Methods, Valid inequalities

Digital Object Identifier 10.4230/OASIcs.ATMOS.2017.10

1 Introduction

A railway timetable is typically planned one year in advance, we call it a master timetable, but it often needs to be revised as the time of the operation approaches. A revision might be necessary for various reasons including for receiving requests to plan additional trains, or to plan for urgent maintenance before the actual time of the operation. Applying optimization

* This study was conducted within the research project "'RELÄET", which is financially supported by grants TRV2016\36067 from The Swedish Transport Administration (Trafikverket). The authors are grateful for all data and support.

methods for revising timetables can be very beneficial especially in highly utilized networks. However, railway timetabling is a complex task and it is almost impossible for human brains to schedule a timetable optimally in dense networks. Hence the need for efficient optimization tools is evident.

A previous study by [3] showed that the applied optimization model can be slow when solving large problems including inserting additional trains and allocating slots for urgent maintenance. The results from [3] showed that in most of the cases, where at most two trains are inserted simultaneously or a track should be allocated for one hour duration maintenance, a feasible timetable can be found within the first minute of model execution but proving optimality of the solution can be time consuming. For larger problems, i.e. inserting more than two trains simultaneously, or longer durations for track allocation for urgent maintenance, even finding a feasible solution can be challenging. Hence, there is a need to investigate and develop strategies for speeding up the solution process.

The optimization-based approach applied in this paper relies on a MILP (Mixed Integer Linear Programming) formulation which is an extension of a previous event-based model developed by [8]. This model was, however, developed for real-time rescheduling purposes, but it is applicable for timetable planning wherever there is a master timetable to be considered. This model can be used for planning operational details including explicit order of trains, explicit order of arrivals at stations where simultaneous entrance at the stations is not allowed, explicit track allocation for bidirectional multi track lines and stations, track and platform lengths.

In order to reduce computation time in MILP models, there exist several approaches including reformulating the problem and developing effective search algorithms. In this paper we focus on strengthening the formulation of the current MILP model. We develop, implement and evaluate certain boosting methods inspired by [4], [10] and [7]. The boosting methods are focused on generating several types of valid inequalities which are implemented in the MILP formulation. The impact of the selected valid inequalities on model runtime is also studied. MILP models are known to often have weak LP relaxations because of the use of so-called disjunctive big M constraints. Therefore we also analyze the impact of the big M constraints on our model performance.

The approach has been experimentally applied on various scenarios for Swedish timetable instances. A railway network in the south of Sweden, including Blekinge Kustbana and the Southern main line between Hässleholm and Malmö, has been selected for the analysis. The master timetable of 2015 is considered. In our experiments we have used the commercial solver Gurobi to solve the problems.

To summarize, the research presented in this paper aims to: i) develop and apply valid inequalities for our MILP model and assess their impact on model performance, ii) analyze the impact of having big M constraints on model performance.

In the remainder of this paper in Section 2, we describe our MILP model which we call the basic model in the rest of the paper. In Section 3, a summary of related previous work is presented. In Section 4, the selected boosting methods are explained. In Section 5, details of the problems and scenarios are presented. Section 6 holds the results and Section 7 summarizes the conclusions and outlines the future research. The full model is presented in Appendix A.

2 Basic model

We consider a MILP model based on the model presented in [9]. The model is further extended in [3] to solve certain timetabling problems which are also addressed in this study.

2.1 General description

In the proposed model, the train traffic is modelled as a set of train events to be allocated to slots on the available track resources. An event represents a train i passing a section j while occupying track q. We distinguish between train event s and section event k. To denote a train involved in the k^{th} event of section j, we write $i_{(j,k)}$ and we let $s_{(j,k)}$ denote the corresponding train event. Given a train i and its train event s, we identify the corresponding section as $j_{(i,s)}$. Basically, train and section events are two different representations of the same event.

We denote T as the set of trains and J is the set of sections. S_i refers to the ordered set of events for train $i \in T$, it contains all the section numbers traversed by train i. K_j refers to the ordered set of events for section $j \in J$, it contains all the train numbers that pass section j. C_j refers to the number of tracks at each section j, while $q_{(i,s)}$ specifies the track number assigned to train i at train event s.

Each train event s, related to train i, has two continuous variables, one for the start time $t^B_{(i,s)}$ and one for the end time $t^E_{(i,s)}$ of the event and one binary variable $x_{(i,s,q)}$ which takes value of one if track q is allocated for train event s. Each section event k, related to section j, has two binary variables for the precedence of trains. In single track lines, the overlap between section events is not allowed, hence there are only two situations allowed after rescheduling: i) either the order of two events is the same as the initial order or ii) the order is changed. A binary variable $\gamma_{(j,k,v)}$ takes value one if the order between two section events k and $k+v$ is the same as the initial order and zero otherwise. The initial order of trains is based on the master timetable. In one section with multiple tracks, events may overlap if they are allocated different tracks, so there is not only a binary choice but multiple alternative choices. In multi-track lines, two section events k and $k+v$ either 1) keep their initial order, or 2) switch the order or 3) have time overlap. Hence, a new binary variable $\lambda_{(j,k,v)}$ is introduced. In multi track lines, $\lambda_{(j,k,v)}$ takes value zero if the order between two section events k and $k+v$ is the same as the initial order and one if the order is swapped. This is elaborated further in Figure 1 since understanding the concept and relations between these variables is the key to generating some of the valid inequalities analyzed in Section 4.1.

We do not allow $\gamma_{(j,k,v)}$ and $\lambda_{(j,k,v)}$ take value one simultaneously, therefore we have $\lambda_{(j,k,v)} + \gamma_{(j,k,v)} \leq 1$ (constraint 26). We also need to make sure that where on the same track, either $\gamma_{(j,k,v)}$ or $\lambda_{(j,k,v)}$ takes value one, therefore we have $x_{(i_{(j,k+v)},s_{(j,k+v)},q_{(i,s)})} + x_{(i_{(j,k)},s_{(j,k)},q_{(i,s)})} \leq \gamma_{(j,k,v)} + \lambda_{(j,k,v)} + 1$ (constraint 33). By this formulation $\gamma_{(j,k,v)}$ and $\lambda_{(j,k,v)}$ both can take value zero simultaneously only when they are not happening on the same track. In this situation the events k and $k+v$ can be overlapped. Note that when the events k and $k+v$ do not happen on the same track, either $\gamma_{(j,k,v)}$ or $\lambda_{(j,k,v)}$ can take value one or they can both be equal to zero, however, we do not care what value they take when not happening on the same track.

The complete model is presented in Sections A.1-A.4. The basic model covers details including track allocation at stations considering train length and track length, restricting simultaneous train entrances at stations where required and respecting technical minimum headway at sections with multiple block sections.

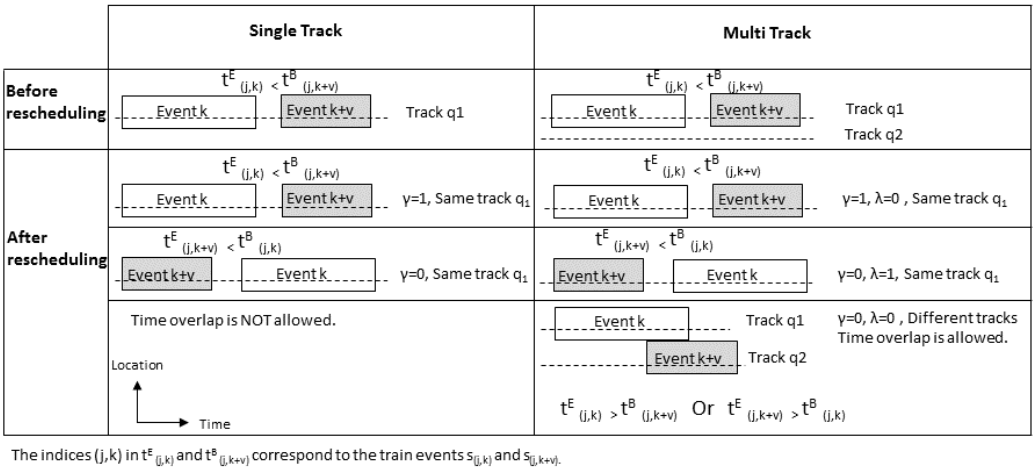

Figure 1 Elaborating variables $\gamma_{(j,k,v)}$ and $\lambda_{(j,k,v)}$.

2.2 Further extensions

We focus on solving problems in which a new train is to be inserted in a master timetable or a slot to be allocated in for urgent maintenance. [3] further developed the model to insert additional trains, the details are presented in the Appendix, Section A.5. Allocating slots for maintenance can be handled as inserting a virtual train in which the minimum travel time equals to the duration of maintenance.

3 Related research

To improve model runtime one can develop efficient algorithms, e.g. based on heuristics and Lagrangian relaxation. However, before developing algorithms we should first try to strengthen the model formulation. An overview of the studies related to applying different modelling and solving algorithms in railway timetabling problems, can be found in [5] and [1]. Applying Integer Programming (IP) and MILP formulations in general timetabling problems as well as in train timetabling problems are very common. In IP formulations the number of variables and constraints grow rapidly if the precision for time variables changes from minutes to seconds. In train timetabling problems IP formulations have been applied for calculating a timetable in minute precision. However, during short term planning, typically we have a large number of variables and constraints and also the time precision should be in seconds. Hence, the use of IP formulation is less practical and MILP formulations are more common. On the other hand, the LP relaxation of MILP formulations including so-called disjunctive big M constraints is known to often be rather weak and solving larger problems with existing powerful commercial MIP solvers can be very difficult and time-consuming. The focus of this research is on reformulation and the reviewed literature here is focused on methods to strengthen MILP formulations.

In [4] possible situations that might arise when solving common MIP formulations with state-of-the-art optimization solvers are described and explained. Recommendations on how to analyze the model properties based on the solver output and potential strategies for model improvements, are also presented.

[10] also proposed several methods for improving MILP formulations and their runtime.

Among them, hybrid Big M formulation, incremental formulation (effective branching) and large formulations (decomposition) can be mentioned. The hybrid Big M formulation is described in more detail in Section 4.2.

For a general scheduling problem, [6] introduced four different time representations for MILP formulations and proposed the corresponding mathematical formulations for each type of the representations. The proposed time representations are based on defining some priority slots and are as follows: i) if several operations can be done in each priority slot or not, ii) if the overlapping of the operations is allowed or not, and iii) if the start time of the operations are synchronized or iv) fixed. They introduced various valid inequalities based on the structure of the data and the type of the formulation and they focused on generating a matrix for non-priority slots. When translating their method for railway scheduling problems, it is applicable only for single track problems. However it can be developed further for double tracks.

In studies related to railway scheduling, [11] applied preprocessing methods to reduced the problem size for routing problems in railway stations in a Dutch network. They also generated some clique inequalities based on the structure of their model. [7] studied several boosting methods for their proposed MILP model. They tested ten different methods and their combinations, and studied the runtime of their proposed model. They reported that for all instances except one, their proposed boosting methods did not have a significant effect on model runtime. Their interpretation of the results is that the initial model is already well-formulated and CPLEX branching strategy is already very good for this kind of problem. However, these results are related to the properties of their model and its specific context.

4 Testing selected boosting methods

We focus on evaluating three sets of valid inequalities. Furthermore we analyze the effects of having constraints with big M.

4.1 Evaluating selected valid inequalities

We develop three sets of valid inequalities and evaluate their impact on model runtime. VIE1 is developed based on some logical relations between variables and is model specific. We think that VIE1 can strengthen the bounds since it can remove undesired MIP solutions. VIE2 is based on the idea of having explicit constraints for transitivity relations between the events in the model. VIE2 is inspired by [7] since their model structure is similar to ours. However, their presented transitivity valid inequalities need to be developed further to be applicable for our model. VIE3 is also based on a logical relation between different variables. We think that it can improve the model runtime although it cannot remove undesired MIP solutions.

4.1.1 Enforcing γ and λ to zero when they are not on the same track (VIE1)

On line sections with multiple tracks ($C_j \geq 2$), non-zero γ and λ are only relevant when their corresponding event is scheduled on the same track. If their corresponding event is not scheduled on the same track then both γ and λ can be set to zero. The following constraints impose that when $x_{(i_{(j,k+v)},s_{(j,k+v)},q)}$ or $x_{(i_{(j,k)},s_{(j,k)},q)}$ (only one of them) is equal to one then both γ and λ should equal to zero. For $j \in J, s \in S_i, k \in \{1...|K_j|-1\}, v \in \{1...|K_j|-k\}, q \in$

$\{1...C_j\}$:

$$x_{(i_{(j,k+v)},s_{(j,k+v)},q)} - x_{(i_{(j,k)},s_{(j,k)},q)} \leq 1 - \gamma_{(j,k,v)} - \lambda_{(j,k,v)} \quad (1)$$

$$x_{(i_{(j,k)},s_{(j,k)},q)} - x_{(i_{(j,k+v)},s_{(j,k+v)},q)} \leq 1 - \gamma_{(j,k,v)} - \lambda_{(j,k,v)} \quad (2)$$

Valid inequalities are commonly defined as constraints which can cut a fractional part of a solution from a relaxed solution and they never cut a MIP solution. On the other hand, valid inequalities VIE1 can cut undesired MIP solutions (they remove those solutions in which variables $\lambda_{(j,k,v)}$ and $\gamma_{(j,k,v)}$ take value 1 when events k and $k+v$ do not happen on the same track), hence they do not fit in the traditional definition of valid inequalities even though they never cut an optimal solution if there is only one unique optimal solution. However, if there are several degenerate optimal solutions, those in which the value of variables $\lambda_{(j,k,v)}$ and $\gamma_{(j,k,v)}$ is one, when not on the same track, will be cut.

4.1.2 Transitivity (VIE2)

The second set of valid inequalities are inspired from a study done by [7]. These valid inequalities enforce the implicit relations between events in an explicit way. Given three successive events on the same track, this set of valid inequalities imposes that if $\lambda_{(j,k,1)}$ between the first event and the second event is 1 and also $\lambda_{(j,k+1,1)}$ between second event and third event is 1 then the $\lambda_{(j,k,2)}$ between the first event and the third event should also be 1. The same relation is relevant for γ as well.

In single track sections ($C_j = 1$), for $j \in J, s \in S_i, k \in \{1...|K_j| - 2\}$:

$$\gamma_{(j,k,1)} + \gamma_{(j,k+1,1)} \leq 1 + \gamma_{(j,k,2)} \quad (3)$$

$$2 - \gamma_{(j,k,1)} - \gamma_{(j,k+1,1)} \leq 2 - \gamma_{(j,k,2)} \quad (4)$$

In multiple tracks sections ($C_j \geq 2$), for $j \in J, s \in S_i, k \in \{1...|K_j| - 2\}$:

$$\lambda_{(j,k,1)} + \lambda_{(j,k+1,1)} \leq 1 + \lambda_{(j,k,2)}\,, \quad (5)$$

$$\gamma_{(j,k,1)} + \gamma_{(j,k+1,1)} \leq 1 + \gamma_{(j,k,2)}\,. \quad (6)$$

As explained before, in the basic model $\gamma_{(j,k,v)}$ and $\lambda_{(j,k,v)}$ can take value of one even though events k and $k+v$ are not happening on the same track, in these cases the above valid inequalities can cut an undesirable MIP solution and are not cutting only fractional parts of a relaxed solution.

4.1.3 Balance between number of section events and track occupancy (VIE3)

The third set of valid inequalities emphasizes that the sum of track occupancy ($x_{(i,s,q)}$) over all track events (k) and all tracks (q) in each section (j) equals to the number of section events ($|K_j|$). These constraints only apply to sections with multiple tracks ($C_j \geq 2$).

$$\sum_{q=1}^{C_{j_{(i,s)}}} \sum_{k=1}^{|K_j|} x_{(i_{(j,k)},s_{(j,k)},q)} = |K_j| \qquad , \qquad j \in J\,. \quad (7)$$

4.2 Hybrid Big M

This method is inspired by [10]. Basically for each big M constraint we attempt to find the smallest value for M which is sufficiently large. In [10] it is referred to as sharp M. We

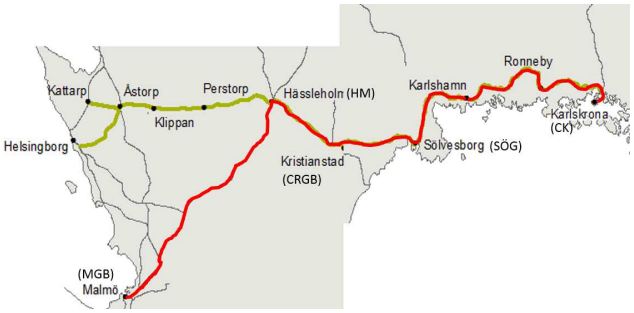

Figure 2 Location of the case study (red line).

categorize constraints by analyzing their left hand side. According to [10], those constraints that have similar left hand side can take the same value for big M. In the basic model the value of M is $24 * 60 * 60 = 86400$ seconds, since the time window in our tested problems is always shorter than 24 hours. For testing the impact of having hybrid M, we define six categories for big M. In each category the value of the big M varies per section j. Category 1 includes equation 23. Category 2 includes equations 24 and 25. Category 3 includes equations 27 and 34. Category 4 includes equation 28. Category 5 includes equations 29 and 35 and category 6 includes equation 30. We take the sharp M values from the obtained optimal solutions when applying the basic model and we solve the problems again and compare the model performance.

5 The studied corridor and timetable instances

The experiments are performed on a timetable instance from Sweden. The timetable instance is extracted from the corridor Karlskrona to Malmö, see Figure 2. A part of the line is single track (from Karlskrona to Hässleholm) and the other part of the line is double track (from Hässleholm to Malmö). The instance is a time window between 16:00 and 23:00 on October 15, 2015. The selected corridor consists of 85 stations. The published daily timetables from Swedish Transport Administrations (Trafikverket) are not always completely feasible and may contain minor violations depending on how e.g. minimum headway values are applied. The selected timetable instance has therefore been calibrated prior to our experiments, in order to ensure feasibility under the assumptions we have made. The studied corridor is highly utilized. The total number of trains running within the time window (16:00-23:00) in the studied network is 295. The peak hours happen between 18:00-20:00 with 27 trains run per track.

5.1 Problem types and scenarios

We test the selected boosting methods on three types of problems:

- Inserting one single train: We select 6 additional trains to be inserted in a master timetable. The selected trains have different minimum travel times, they run on different parts of the corridor and in different directions. We generate random insertion time for each additional train. The random insertion time is the departure time from the first section. Each of the 6 trains are assigned 8 random insertion times which sums to 48 scenarios in total.

- Allocating slots for urgent maintenance, fixed hour, different sections: One hour maintenance between 21:00-22:00 is to be allocated for all line sections with double tracks. This hour is selected since maintenance is preferred to be done after the rush hours. In the studied corridor we have 20 line sections with double tracks that means we have 20 scenarios.
- Allocating slots for urgent maintenance, variable hour, fixed section: For section TÖ-HÖ one hour is randomly selected from the time window (16:00-23:00). To be consistent when comparing the results for maintenance problems, here we test 20 scenarios as well. This section is selected since TÖ-HÖ is located in the middle of the double track segment of the line and has an equally distributed traffic in both direction.

6 Results

This section holds the main results from the studied scenarios described in Section 5. The model is implemented in Java and solved using Gurobi 6.5 on a PC i7-5600U at 2.60 GHz, 8 GB of RAM, running with windows 7-64.

6.1 Results of the experiments for the selected valid inequalities

We test the selected valid inequalities on scenarios from all three types of problems. We solve each scenario four times, first with applying the basic model, then applying each of the mentioned valid inequalities. The objective function when inserting a single train is to minimize the time deviation from the master timetable for arrival and departure times for all the train events at all sections, plus total travel time of the additional train, plus track deviations from the master timetable for all trains. However, in the Swedish context, the scheduled departure and arrival times of the existing trains are not to be changed by the infrastructure manager. Furthermore, in our scenarios the assigned tracks for pre-planned stops at stations, should remain unchanged. For the maintenance problems, the objective function is the same as before, but the weight for track deviations is set to 300 (seconds). From a minimization problem perspective, this means that a track change would be equivalent to a 5 minute time deviation for one train. In maintenance problems arrival/departure times and track allocations for the existing trains are flexible.

6.1.1 VIEs defined as main constraints

The influencing factors and the performance indicators that we consider when analyzing the model performance are: size of problems, number of simplex iterations, time for presolve, normalized objective value and gap over model runtime and the time for finding the first feasible solution.

Size of problems. When implementing valid inequalities, the number of variables remains the same but the number of constraints increases and this might lead to tighter bounds and improve the performance of the model which should be tested. In the current experiments for all three types of problems, the number of constraints doubled approximately after implementing VIE1 (e.g. from approximately 2,800,000 in the basic model to approximately 5,700,000 when implementing VIE1 for inserting a single train). For VIE2 and VIE3 the number of constraints increased slightly (less than 500,000 for the same example).

Simplex iterations. Having fewer number of simplex iterations is not an indication of a better performance. However, comparing the number of simplex iterations can show how the model behaves when implementing certain valid inequalities. Experiments shows that the number of simplex iterations was at least doubled when implementing VIE1. The changes in the number of iterations for the other set of valid inequalities were not considerable.

Time for presolve. Time for presolve when implementing valid inequalities either increased or remained almost the same as the time for presolve in the basic model. For VIE1, time for presolve increased by 2 to 3 times.

Normalized objective value and gap over model runtime and the time for finding the first feasible solution. Figure 3 shows the results from one scenario (1 out of 48) for inserting a single train. On the left we have the normalized objective value and on the right we have gap over model runtime. For this instance we can see that VIE2 and VIE3 perform almost the same as the basic model while VIE1 changes the performance. It is obvious that in the basic model and VIE2 and VIE3 the heuristics performed better, the first solution is found earlier. Later and close to the optimal solution, the heuristics find the optimal solution. In VIE1 the first solution is found later (compared to the basic model and other VIEs) and then there is a steady computation over several iterations to find the optimal solution. Figure 4 shows the normalized objective values and gap for 48 scenarios when applying VIE1. The dotted lines are the results from applying VIE1 and the solid lines are the results from the basic model. We can see that in almost all of the scenarios applying VIE1 makes the model slower often because of not finding a good first feasible solution fast. We have also done some experiments with providing the first feasible solution, still VIE1 does not perform better than the basic model and the other two VIEs. From the gap results from the same set of experiments, it is obvious that the first feasible solution is found later, compared to the basic model, and the found solutions by heuristics are less frequent. When applying VIE1 proving optimality takes more time compared to the other scenarios and the basic model.

▶ Remark. Based on the evidences we can conclude that VIE1 causes weaker LP relaxations. In Table 1, for single train insertion problem, when applying VIE1, none of the scenarios had a shorter computation time compared to the basic model. When applying VIE2, in 30 out of 48 scenarios (62% of the scenarios) the computation time was shorter (with maximum 436 seconds and average 43 seconds, on average 28%, improvements). When applying VIE3, in 24 out of 48 scenarios (50% of the scenarios) the computation time was shorter (with maximum 155 seconds and average 25 seconds, on average 25% improvements). When implementing all sets of valid inequalities at the same time, VIE1 have a large influence on the results and the results have the same properties as implementing VIE1 only. In Table 2, the results for maintenance problems are presented. In most of the scenarios a feasible solution is found within the time limit but the gap can be large. We can observe that VIE2 and VIE3 could reduce the average computation time in scenarios with random one hour maintenance while VIE1 increased the average runtime in all maintenance scenarios.

6.1.2 VIEs defined as user-defined cuts

We test whether applying VIEs as user cuts (instead of main constraints) can reduce the computation time since user cuts are generated only if some user defined criteria are valid. In our method we define the criterion where the gap is below 20%. However, VIE1 and VIE2 are not pure valid inequalities and they indeed can cut feasible MIP solution in some scenarios,

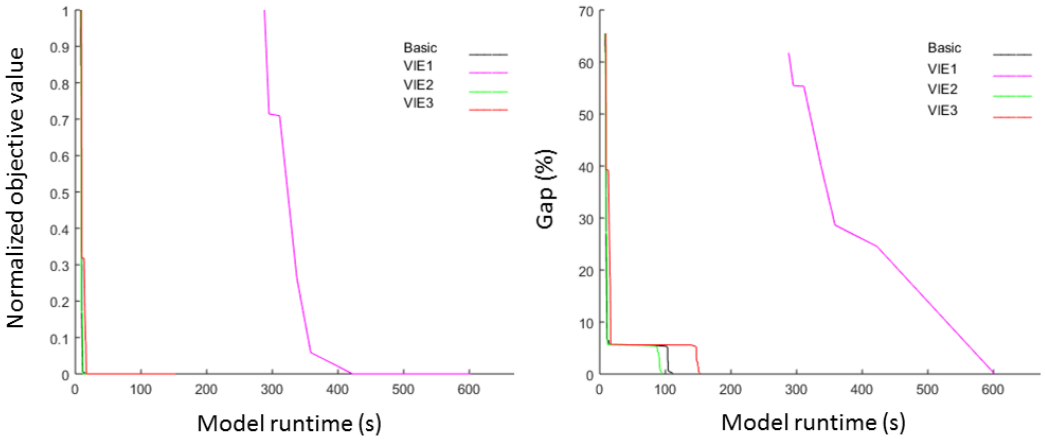

Figure 3 Comparing the normalized objective values and gap for one scenario instance.

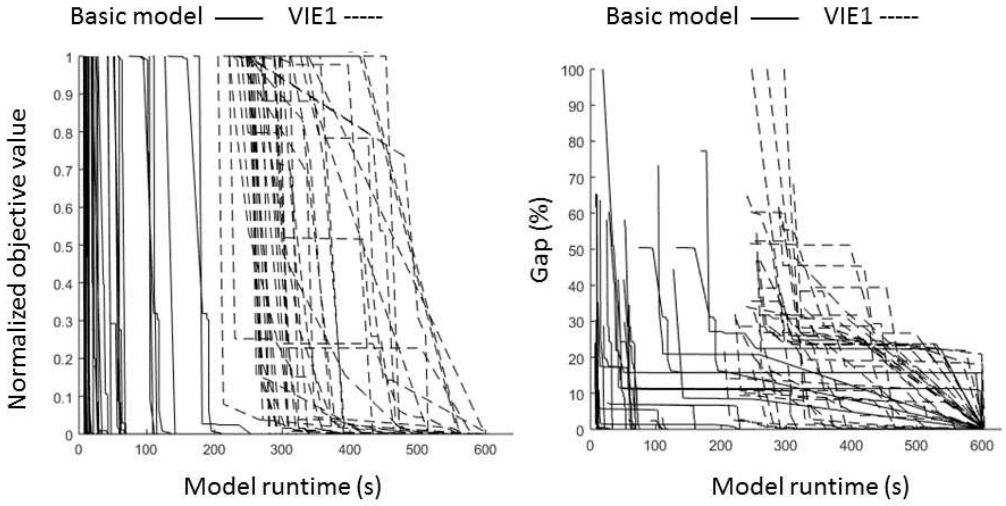

Figure 4 Normalized objective values and gap for 48 scenarios when inserting a single train.

Table 1 Results of the scenarios for testing the impact of valid inequalities on inserting a single train problem, 48 scenarios (min,avg,max).

Inserting a single train	Computation Time (mm:ss)	Gap (%)	Objective value	Non-optimal solutions (#)	Improve in computation time (%)
Basic	(00:07,02:13,–)	(0,2,22)	(1587,3802,7734)	6	NA
VIE1	(01:38,08:10,–)	(0,9,32)	(1587,3986,7751)	31	0
VIE2	(00:07,02:04,–)	(0,2,22)	(1587,3802,7734)	6	28% of 62% of scen.
VIE3	(00:07,02:07,–)	(0,2,22)	(1587,3802,7734)	6	25% of 50% of scen.

Gap tolerance: 0.03(%), Time limit: 10 min. '–' Terminated by time limit. All values are rounded.

Table 2 Results of the scenarios for testing the impact of valid inequalities on allocating urgent maintenance problems, 20 scenarios, (min,avg,max).

Closing section	VIE.	Maintenance time window	Computation time (mm:ss)	Gap (%)	Objective value
Random	Basic	21-22	(00:03,00:47,–)	(0,4,66)	(1998,9725,60900) 1 non-opt. 2 infeasible
	VIE1	21-22	(00:07,01:13,–)	(0,4,75)	(1998,10354,72220) 1 non-opt. 2 infeasible
	VIE2	21-22	(00:03,00:49,–)	(0,4,69)	(1998,9830,62777) 1 non-opt. 2 infeasible
	VIE3	21-22	(00:03,00:47,–)	(0,4,67)	(1998,9873,63549) 1 non-opt. 2 infeasible
TÖ-HÖ	Basic	1h random	(00:12,02:59,–)	(0,1,10)	(9053,25296,41783) 2 non-opt. 6 infeasible
	VIE1	1h random	(00:29,03:01,–)	(0,4,30)	(9053,25505,42465) 2 non-opt. 6 infeasible
	VIE2	1h random	(00:10,02:00,–)	(0,1,10)	(9053,25294,41783) 2 non-opt. 6 infeasible
	VIE3	1h random	(00:09,01:49,–)	(0,1,11)	(9053,25292,41783) 2 non-opt. 6 infeasible

Gap tolerance: 0.03(%), Time limit: 10 min. '–' Terminated by time limit. All values are rounded.

therefore they cannot be implemented as user-defined cuts by definitions in Gurobi [2]. On the other hand, VIE3 can be implemented as user-defined cuts instead of main constraints. By doing so, unlike what we expected, the average of model runtime increased from 127 s to 156 s and 9 solutions out of 48 scenarios were non-optimal after 10 min model execution, compared to 6 in the basic model. One reason might be because of having too many Gurobi call-backs during the optimization process.

Updating VIE2. Notice that in VIE2 we did not include the condition of *"when happening on the same track"*. We can apply VIE1 and VIE2 together to make sure that VIE2 is enforced when on the same track. In this case, as mentioned before, the results are affected very much by VIE1. However, we can also reformulate VIE2 such that it will be enforced only when events are happening on the same track. We tested whether we can transform VIE2 to pure valid inequality by applying some changes. Hence, we include track allocation in the constraints 5 and 6 and update them, in multi tracks sections ($C_j \geq 2$), for $j \in J, s \in S_i, k \in \{1...|K_j|-2\}, q \in \{1...C_j\}$:

$$\lambda_{(j,k,1)} + \lambda_{(j,k+1,1)} + x_{(i_{(j,k)},s_{(j,k)},q)} + x_{(i_{(j,k+1)},s_{(j,k+1)},q)} + x_{(i_{(j,k+2)},s_{(j,k+2)},q)} \leq 4 + \lambda_{(j,k,2)} \quad (8)$$

$$\gamma_{(j,k,1)} + \gamma_{(j,k+1,1)} + x_{(i_{(j,k)},s_{(j,k)},q)} + x_{(i_{(j,k+1)},s_{(j,k+1)},q)} + x_{(i_{(j,k+2)},s_{(j,k+2)},q)} \leq 4 + \gamma_{(j,k,2)} \quad (9)$$

The above constraints narrow VIE2 to those events that happen on the same track q. We call constraints 8 and 9 Updated-VIE2. The results shows that when inserting one single train, still the Updated-VIE2 can cut MIP solutions, meaning that they are not pure valid inequalities and they cannot be implemented as user cuts. However, when implemented as additional main constraints, in 33% of scenarios (16 out of 48) we had improvements in runtime with average 18% and maximum 70%. The average runtime became 118 s (compared to 124 s). For maintenance problems, in the scenarios that we tested, Updated-VIE2 never cut a MIP solution and hence could be implemented as user cuts. The results show that for random section scenarios only in 2 scenarios (out of 20) the model runtime decreased (with max 2%). In all the scenarios for random one hour in section TÖ-HÖ the model runtime increased (compared to the basic scenario).

6.2 Results of the experiments for hybrid big M

Applying the hybrid big M method shows the impact of the big M value on the computation time. In real practice big M can be bounded to the time window of the timetable. The results show that the computation time is very sensitive to sharp values for M. The computation time for those scenarios which took 1 minute approximately (in all three types of problems), dropped to 1 or 2 seconds. However, in real practice it is not easy to estimate the sharp value of big M since if M is not big enough it might change the solution space. On the other hand having very small values for big M might cause confusion in Gurobi because of the scaling problem. Hence, when the sharp value is unknown, it is better to have relatively large big M values and let the solver deal with the scaling problem itself.

6.3 Other influencing factors

We use solver Gurobi with default parameter setting. However, Gurobi has an option for parameter tuning and we can let the parameter tuning option select the best values for parameters and analyze the effects on model runtime. When inserting a single train, following the tuning option recommendations we could decrease model runtime for an instance from 115 s to 18 s. Following the recommendation from the parameter tuning option for maintenance problem, model runtime decreased from 6:06 to 4:43 (mm:ss) for an instance. Although the tuning option can help decreasing the model runtime, it is difficult to find a tuning pattern for different types of problems. This means that to have relatively good results, the set of tuning parameters needs to be optimized for each separate problem.

7 Conclusions and future research

The strengths and weaknesses of a MILP model are analyzed for certain practical timetabling problems. To decrease the computation time, three different groups of valid inequalities were tested on a rather limited set of scenarios and their impact on computation was studied. Results show that valid inequalities VIE2 (transitivity) and VIE3 (section balance) for some scenarios can improve model performance but they did not have a significant effect on the computation time. Valid inequalities VIE1 (enforcing values for γ and λ), caused weak relaxation and they did not improve model performance for the limited number of scenarios we evaluated. The reason behind is that this set of valid inequalities can cut degenerate MIP solutions. Therefore, they make the model tighter and remove some feasible MIP solutions which have a negative impact on LP relaxation and the embedded heuristics in Gurobi. Valid inequalities can be implemented as user defined cuts in Gurobi only if the

VIEs do not cut any feasible MIP solutions. In our scenarios this method is therefore possible only for VIE3 (section balance). However, after implementing VIE3 as user-cuts the model runtime increased slightly. This might be because of having too many call-backs during the optimization process. The parameter tuning option in Gurobi is strong but it is scenario specific and the optimal parameter setting for one scenario cannot be extended to the other scenarios. Furthermore, this tool is only useful if optimality can be reached in a reasonable time. Providing sharp big M values for the studied scenarios, improved model performance which can indicate that the model is sensitive to big M values.

In this research only those boosting methods related to improving model formulation were explored. Introducing auxiliary variables to improve LP relaxations and boosting methods related to providing search algorithms can be investigated in future studies.

References

1 Valentina Cacchiani, Dennis Huisman, Martin Kidd, Leo Kroon, Paolo Toth, Lucas Veelenturf, and Joris Wagenaar. An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63:15–37, May 2014. doi:10.1016/j.trb.2014.01.009.

2 Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2017. URL: http://www.gurobi.com.

3 Fahimeh Khoshniyat and Johanna Törnquist Krasemann. On-demand timetabling in dense railway networks: Methods and challenges. In *Proceedings of 7th International Conference on Railway Operations Modelling and Analysis - RailLille*, 2017.

4 Ed Klotz and Alexandra M. Newman. Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, 18(1–2):18–32, October 2013. doi:10.1016/j.sorms.2012.12.001.

5 Richard M Lusby, Jesper Larsen, Matthias Ehrgott, and David Ryan. Railway track allocation: models and methods. *OR spectrum*, 33(4):843–883, 2011.

6 Sylvain Mouret, Ignacio E. Grossmann, and Pierre Pestiaux. Time representations and mathematical models for process scheduling problems. *Computers & Chemical Engineering*, 35(6):1038–1063, June 2011. doi:10.1016/j.compchemeng.2010.07.007.

7 Paola Pellegrini, Grégory Marliere, Raffaele Pesenti, and Joaquín Rodriguez. RECIFE-MILP: An Effective MILP-Based Heuristic for the Real-Time Railway Traffic Management Problem. *IEEE Transactions on Intelligent Transportation Systems*, 16(5):2609–2619, October 2015. doi:10.1109/TITS.2015.2414294.

8 Johanna Törnquist and Jan A. Persson. N-tracked railway traffic re-scheduling during disturbances. *Transportation Research Part B: Methodological*, 41(3):342–362, 2007. doi:10.1016/j.trb.2006.06.001.

9 Johanna Törnquist Krasemann. Computational decision-support for railway traffic management and associated configuration challenges: An experimental study. *Journal of Rail Transport Planning & Management*, 2015. doi:10.1016/j.jrtpm.2015.09.002.

10 Juan P. Vielma. Mixed Integer Linear Programming Formulation Techniques. *SIAM Review*, 57(1):3–57, January 2015. doi:10.1137/130915303.

11 Peter J. Zwaneveld, Leo G. Kroon, and Stan P.M. van Hoesel. Routing trains through a railway station based on a node packing model. *European Journal of Operational Research*, 128(1):14 – 33, 2001. doi:10.1016/S0377-2217(00)00087-4.

A Appendix: The applied timetabling model

In this appendix a complete description of the used optimization model is given. The model is a mixed integer linear program (MILP) and an event-based description of railway traffic. An event represents a train i passing a section j while occupying track q. We distinguish between train event s and section event k. To denote a train involved in the k^{th} event of section j, we write $i_{(j,k)}$ and we let $s_{(j,k)}$ denote the corresponding train event. Given a train i and its train event s, we identify the corresponding section as $j_{(i,s)}$.

Sets and parameters. T: set of trains. J: set of sections. S_i: ordered set of events for train $i \in T$, it contains all the section numbers traversed by train i. K_j: ordered set of events for section $j \in J$, it contains all the train numbers that pass section j. C_j: number of tracks at each station j. $\hat{t}^B_{(i,s)}$: initial start time of train $i \in T$ at event $s \in S_i$. $\hat{t}^E_{(i,s)}$: initial end time of $i \in T$ at event $s \in S_i$. t^c_j: clearance time at section j. $t^{min}_{(i,s)}$: minimum time required for train i to pass event s. $q_{(i,s)}$: track number assigned to train i at train event s. l_i: Length for train i. $L_{(j,q)}$: Length for track q in section j. M: sufficiently large constant, here 86400 seconds. h_t: technical minimum headway time between any two consecutive trains i and i+1 on a section. $\rho_{(i,s)}$ is 1 if train i has a scheduled stop at train event s and 0 otherwise. $\delta_{(i,s)}$ is 1 if event s for train i is on a line section and 0 if event s for train i is at a station section. $\mu_{(j)}$ is 1 if section j is a line section and 0 if section j is a station section. $\phi_{(i)}$ is 1 if direction for train i is southbound and 0 if direction for train i is northbound. $\theta_{(j)}$ is 1 if simultaneous arrival at station j is allowed and is 0 if simultaneous arrival at station j is not allowed.

Variables. $t^B_{(i,s)} \geq 0$: start time of train $i \in T$ at event $s \in S_i$. $t^E_{(i,s)} \geq 0$: end time of train $i \in T$ at event $s \in S_i$. $d_{(i,s)} \geq 0$: absolute deviation between the new calculated departure time and the initial departure time for train i at event s. $a_{(i,s)} \geq 0$: absolute deviation between the new calculated arrival time and the initial arrival time for train i at event s. $x_{(i,s,q)}$ is 1 if track q is assigned to train i at event s and 0 otherwise. $\gamma_{(j,k,v)}$ is 1 if event k occurs before event k+v, as in the initial timetable (k<k+v) and is 0 otherwise. $\lambda_{(j,k,v)}$ is 1 if event k is changed to occur after event k+v (k<k+v & $C_j \geq 2$), and 0 otherwise. $\omega_{(j,k,v)}$ is 1 if event k is on a station and occurs before event k+v and (k<k+v,$\delta_{(i_{(j,k)},s_{(j,k)})} = 0$, $\delta_{(i_{(j,k+v)},s_{(j,k+v)})} = 0$, $C_j \geq 2$). It takes value 0 if event k is on a station and occurs after event k+v. $y_{(i,s)} \geq 0$ is 1, if the allocated track number is different from the allocated track number in the master timetable, 0 otherwise.

Objective function

$$\min[\sum_{i\in T}\sum_{s\in S_i}((d_{(i,s)} + a_{(i,s)}) \cdot w_1) + \sum_{i\in T}\sum_{s\in S_i}(y_{(i,s)} \cdot w_2)] \tag{10}$$

Constraints

$$|t^B_{(i,s)} - \hat{t}^B_{(i,s)}| \leq d_{(i,s)} \quad , \quad i \in T, s \in S_i \tag{11}$$

$$|t^E_{(i,s)} - \hat{t}^E_{(i,s)}| \leq a_{(i,s)} \quad , \quad i \in T, s \in S_i \tag{12}$$

$$x_{(i,s,q')} \leq y_{(i,s)} \quad , \quad i \in T, s \in S_i, C_{j_{(i,s)}} \geq 2 \quad \& \quad q'_{(i,s)} \neq q_{(i,s)} \tag{13}$$

$$y_{(i,s)} \leq 1 - x_{(i,s,q')} \quad , \quad i \in T, s \in S_i, C_{j_{(i,s)}} \geq 2 \quad \& \quad q'_{(i,s)} = q_{(i,s)} \tag{14}$$

$$y_{(i,s)} \leq 0 \quad , \quad i \in T, s \in S_i, C_{j_{(i,s)}} = 1 \tag{15}$$

$$t^E_{(i,s)} = t^B_{(i,s+1)} \quad , \quad i \in T, s \in S_i \tag{16}$$

$$t^E_{(i,s)} \geq t^B_{(i,s)} + t^{min}_{(i,s)} \quad , \quad i \in T, s \in S_i \tag{17}$$

$$t^B_{(i,1)} \geq \hat{t}^B_{(i,1)} \quad , \quad i \in T \tag{18}$$

$$t^E_{(i,s)} \geq \hat{t}^E_{(i,s)} \quad , \quad i \in T, s \in S_i : \rho_{(i,s)} = 1 \tag{19}$$

$$x_{(i,s,q_{(i,s)})} = 1 \quad , \quad i \in T, s \in S_i : \delta_{(i,s)} = 1 \tag{20}$$

$$t^B_{(i,1)} = \hat{t}^B_{(i,1)} \quad , \quad i \in T : \hat{t}^B_{(i,1)} \leq T_0 \tag{21}$$

$$x_{(i,1,q_{(i,1)})} = 1 \quad , \quad i \in T : C_{j_{(i,1)}} \geq 2, \hat{t}^B_{(i,1)} \leq T_0 \tag{22}$$

When event k happens before event $k+v$, as it is in the initial plan then: $j \in J$ & $k \in \{1...(|K_j|-1)\}$ & $v \in \{1...(|K_j|-k)\} : \phi_{i_{(j,k)}} \neq \phi_{i_{(j,k+v)}}$

$$t^B_{(i_{(j,k+v)},s_{(j,k+v)})} - t^E_{(i_{(j,k)},s_{(j,k)})} \geq \gamma_{(j,k,v)} \cdot t^c_j - M \cdot (1 - \gamma_{(j,k,v)}) \tag{23}$$

When event k happens after event $k+v$, the order is reversed compared to the initial plan.

$$t^B_{(i_{(j,k)},s_{(j,k)})} - t^E_{(i_{(j,k+v)},s_{(j,k+v)})} \geq (1 - \gamma_{(j,k,v)}) \cdot t^c_j - M \cdot \gamma_{(j,k,v)} \tag{24}$$

When the order of trains is reversed compared to the initial order; this constraint is only necessary for double tracks.

$$t^B_{(i_{(j,k)},s_{(j,k)})} - t^E_{(i_{(j,k+v)},s_{(j,k+v)})} \geq \lambda_{(j,k,v)} \cdot t^c_j - M \cdot (1 - \lambda_{(j,k,v)}) \tag{25}$$

$$\lambda_{(j,k,v)} + \gamma_{(j,k,v)} \leq 1 \quad , \quad C_j \geq 2 \tag{26}$$

For single and double track keeping the initial order: $j \in J$ & $k \in \{1...(|K_j|-1)\}$ & $v \in \{1...(|K_j|-k)\} : \phi_{i_{(j,k)}} = \phi_{i_{(j,k+v)}}$

$$t^B_{(i_{(j,k+v)},s_{(j,k+v)})} - t^B_{(i_{(j,k)},s_{(j,k)})} \geq h_t - M \cdot (1 - \gamma_{(j,k,v)}) \tag{27}$$

$$t^E_{(i_{(j,k+v)},s_{(j,k+v)})} - t^E_{(i_{(j,k)},s_{(j,k)})} \geq h_t - M \cdot (1 - \gamma_{(j,k,v)}) \tag{28}$$

For double tracks with reverse order: $j \in J$ & $k \in \{1...(|K_j| - 1)\}$ & $v \in \{1...(|K_j| - k)\}$: $\phi_{i_{(j,k)}} = \phi_{i_{(j,k+v)}}$

$$t^B_{(i_{(j,k)},s_{(j,k)})} - t^B_{(i_{(j,k+v)},s_{(j,k+v)})} \geq h_t - M \cdot (1 - \lambda_{(j,k,v)}) \tag{29}$$

$$t^E_{(i_{(j,k)},s_{(j,k)})} - t^E_{(i_{(j,k+v)},s_{(j,k+v)})} \geq h_t - M \cdot (1 - \lambda_{(j,k,v)}) \tag{30}$$

$$\sum_{q=1}^{C_{j_{(i,s)}}} x_{(i,s,q_{(i,s)})} = 1 \qquad , \qquad i \in T, s \in S_i, C_{j_{(i,s)}} \geq 2 \tag{31}$$

$$x_{(i,s,q_{(i,s)})} = x_{(i,s+1,q_{(i,s+1)})} \qquad , \qquad i \in T, s \in S_i : C_{j_{(i,s)}} \geq 2, C_{j_{(i,s)}} = C_{j_{(i,s+1)}} \tag{32}$$

For $j \in J, s \in S_i, i \in T, k \in \{1...|K_j| - 1\, v \in \{1...|K_j| - k\} : \mu_{j_{(i,s)}} = 0 \& C_j \geq 2 \& \theta_{(j)} = 0$

$$x_{(i_{(j,k+v)},s_{(j,k+v)},q_{(i,s)})} + x_{(i_{(j,k)},s_{(j,k)},q_{(i,s)})} \leq \gamma_{(j,k,v)} + \lambda_{(j,k,v)} + 1 \tag{33}$$

$$t^B_{(i_{(j,k+v)},s_{(j,k+v)})} - t^B_{(i_{(j,k)},s_{(j,k)})} \geq t^c_j \cdot \omega_{(j,k,v)} - M \cdot (1 - \omega_{(j,k,v)}) \tag{34}$$

$$t^B_{(i_{(j,k)},s_{(j,k)})} - t^B_{(i_{(j,k+v)},s_{(j,k+v)})} \geq t^c_j \cdot (1 - \omega_{(j,k,v)}) - M \cdot \omega_{(j,k,v)} \tag{35}$$

$$\gamma_{(j,k,v)} \leq \omega_{(j,k,v)} \tag{36}$$

$$\lambda_{(j,k,v)} + \omega_{(j,k,v)} \leq 1 \tag{37}$$

For $i \in T, s \in S_i, q \in \{1...|C_j|\} : \mu_{j_{(i,s)}} = 0 \& \rho_{i,s} = 0 \& C_{j_{(i,s)}} \geq 2 \& l_i > L_{(j,q)}$

$$t^E_{(i,s)} - t^B_{(i,s)} \leq M \cdot (1 - x_{(i,s,q)}) \tag{38}$$

A.1 Adjustments in the extended model

The model is extended to solve problems of inserting additional trains and allocating slots for urgent maintenance (defined as virtual additional trains). All the additional trains should respect all the previous constraints. Therefore the additional trains are added to the set of trains T but they can be distinguished by their train number.

Additional sets and parameters. n: Number of new trains to be inserted. m_i: Train numbers for train i. The list of trains also include the train numbers for the additional trains. o_e: Train numbers for additional train e. $p_{(i,e)}$: Minimum travel time for train i when $m_i = o_e$. r: Start time of the time window in seconds. Here it is equal to 16*60*60. u: End time of the time window in seconds. Here it is equal to 23*60*60. g_e: Random Start Time of additional train e. d_i: Last event for train i at its destination.

Additional variables. $z_{(i,e)}$: Travel time for train i when $m_i = o_e$. It is a continuous variable. $y_{(i,s)} \geq 0$: 1, if the allocated track number is different from the allocated track number in the master timetable, 0 otherwise.

Adjusted objective function. When inserting additional trains, since the existing trains are fixed then the time deviation from the master timetable for the existing trains will be zero. The part for track deviation can be separated for the existing and the additional trains. w_1 and w_1 are some weights.

$$\min[\sum_{i\in T}\sum_{s\in S_i}(y_{(i,s)}\cdot w_1) + \sum_{i\in T}\sum_{e=1}^{n}(z_{(i,e)}\cdot w_2)] \tag{39}$$

Additional constraints. For the travel time of the additional trains: For $i \in T, e \in \{1...n\}$: $m_i = o_e$

$$t^E_{(i,d_i)} - t^B_{(i,1)} \leq z_{(i,e)} \tag{40}$$

For respecting the insertion time window: For $i \in T, e \in \{1...n\}$: $m_i = o_e$

$$t^E_{(i,d_i)} \leq u \tag{41}$$

$$t^B_{(i,1)} \leq r \tag{42}$$

$$t^B_{(i,1)} \leq g_e + 0.5\cdot 60\cdot 60 \tag{43}$$

$$t^B_{(i,1)} \geq g_e - 0.5\cdot 60\cdot 60 \tag{44}$$

The arrival and departure times for the existing trains are fixed according to their corresponding values in the master timetable. For fixing the arrival and departure times for the existing trains: For $i \in T, e \in \{1...n\}$: $m_i \neq o_e$

$$t^B_{(i,s)} = \hat{t}^B_{(i,s)} \tag{45}$$

$$t^E_{(i,s)} = \hat{t}^E_{(i,s)} \tag{46}$$

For the maintenance problems, constraints 45 and 46 are relaxed.