Deliverable D6.1
Report on the description of algorithms for long-term timetabling, short-term timetabling and rolling stock planning

Project acronym:	MOTIONAL
Starting date:	01-12-2022
Duration (in months):	46
Call (part) identifier:	HORIZON-ER-JU-2022-01
Grant agreement no:	101101973
Due date of deliverable:	30-11-2024 (M24)
Actual submission date:	03-12-2024
Code	FP1-WP6-D-SINTEF-01-01
Responsible/Author:	Carlo Mannino (SINTEF A.E. NRD)
Dissemination level:	PU
Status:	Issued

Reviewed: Yes
Alessandro Mascis
Stefanie Schöne

This project has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101101973.

Document history		
Revision	*Date*	*Description*
1	18-10-2024	Complete draft ready for internal review
2	27-11-2024	Complete version ready for external review
3	02-12-2024	Revised version ready to be sumitted

Report contributors		
Name	Beneficiary Short Name	Details of contribution
Carlo Mannino	NRD A.E. SINTEF	Author Sections 1, 3, 8, 8.1, 10
Dennis Huisman	NSR	Author Sections 2, 4, 5, 6, 9
Giorgio Sartor	NRD A.E. SINTEF	Author Section 7, 7.1
Bjørnar Luteberget	NRD A.E. SINTEF	Author Section 6.4
Gábor Maróti	NSR	Author Section 7.3
Jord Boeijink	NSR	Author Section 8.2
Paolo Ventura	SIEMENS	Author Section 7.5
Leonardo Lamorgese	SIEMENS	Author Section 7.5
Emma Solinen	TRV	Reviewer Section 6.3, 6.5, 7.2
Sara Gestrelius	TRV A.E. RISE	Author Section 6.5
Carl Henrik Häll	TRV A.E. LIU	Author Section 7.2
Mikael Fredriksson	TRV A.E. LIU	Author Section 7.2
Ingrid Johansson	TRV A.E. KTH	Author Section 6.3
Philipp Widmann	DLR	Author Section 6.2
Florian Brinkmann	DLR	Author Section 6.2
Rémy Chevrier	SNCF	Author Section 7.4
Christelle Lérin	SNCF	Author Section 7.4
Roberto Bianchi	HITACHI	Author Section 7.1
Enrique Gómez	INDRA	Author Section 7.6
Carmen Ramos	INDRA	Author Section 7.6
Nabil Absi	EMSE	Author Section 7.4
Stéphane Dauzère-Pérès	EMSE	Author Section 7.4
Karim Terfasse	EMSE / SNCF	Author Section 7.4
Yahan Lu	NSR A.E. TU Delft	Author Section 6.1

Table of Contents

1. Executive Summary 4
2. Abbreviations and acronyms 5
3. Background 6
4 .Objective/Aim 10
5. State-of-the-Art in planning and timetable optimization 11
6. Algorithms for long-term timetabling 14
6.1 Algorithm for creating strategic network timetables 15
6.2 Robust station routing and strategic station planning 26
6.3 Analysis of station capacity utilisation 36
6.4 Conflict-based search algorithm for incremental timetabling 40
6.5 Usability of an optimization-based decision support system for annual capacity allocation .45
7. Algorithms for short-term timetabling 71
7.1 An efficient MILP-based algorithm for handling TCRs 71
7.2 An algorithm for adjusting train timetables in case of TCRs 78
7.3 Algorithm for constructing a nation-wide adjusted hourly timetable 87
7.4 Optimized Short-Term Insertion of Additional Trains Into Existing Timetables 98
7.5 A new spatial decomposition for (Short-term) Train Timetabling 108
7.6 A genetic algorithm for decision support in timetable optimisation 115
8. Algorithms for rolling-stock planning 138
8.1 A MILP-based algorithm for rolling stock scheduling 138
8.2 A local-search algorithm for rolling stock stabling 147
9. Interaction with other Flagship Projects 162
10. Conclusions 163
11. References 165
12. Appendices 170

1. Executive Summary

This deliverable describes the main developments carried out in WP6, with focus on models and algorithms to improve long-term and short-term timetabling of the railway network.
The activities in WP6 aim at increasing infrastructure and transport utilisation capacity through optimised and robust timetables, synchronized with rolling stock planning.
These objectives have been targeted through

a. The development of advanced algorithms for the generation and adjustment of timetables and rolling stock planning, which will be further developed and completed in WP7
b. The definition a suitable family of use cases, which will be demonstrated in WP7
c. The implementation of specific technical enablers.

Addressed technical enabler and the defined use cases and demonstrators are synthetically reported in the background Section 3 of this document.

All the timetabling and rolling stock planning problems tackled in WP6 require to find good quality solutions fulfilling various physical and logical requirements, and the business rules of the railway infrastructure managers and railway undertakings. As such, they can be viewed as *optimization problems*, which can be modelled and solved by means of *mathematical optimization*, an AI discipline which concerns the making of optimal decisions. With few exceptions, the models developed in WP6 are based on Mixed Integer Linear Programming or Constraint Programming, solved then by means of specialized commercial solvers (as CPLEX, or GUROBI), or by ad-hoc heuristic algorithms, such as *local search, genetic algorithms, simulated annealing.* Mathematical decomposition and graph theory are also exploited to model and solve various problems.

Although the algorithms developed in WP6 are not yet fully completed – they will be in the first year of WP7 – still some interesting and promising conclusions can be drawn. In fact, tests on realistic instances have been performed. It turns out that the developed methods work well for the size and the type of instances for which they are designed. In turn, this implies that we can expect they will tackle the instances arising in the demonstrations of the planned use-cases. Ultimately, this means that in general the approaches will be able to support human planners in their activities, and to automatize segments of the current planning process. Preliminary results show that solutions of high quality can be produced in short computing time. One limit is that, since the algorithms will be completed and demonstrated in WP7, these conclusions are still very preliminary. Also, the solution of full integrated problems appears to be still out of reach, and we need to content ourselves with tackling suitable subproblems. For instance, we can possibly compute an optimal or quasi-optimal timetable for the entire Norwegian network, and subsequently calculate an associated optimal rolling stock rotation, and solve the stabling problem, but we are still far from being able to solve to optimality the three problems jointly.

2. Abbreviations and acronyms

Abbreviation / Acronym	Description
AHT	Adjusted Hourly Timetable
AI	Artificial Intelligence
ALNS	Adaptive Large Neighbourhood Search
CBS	Conflict Based Search
CP	Constraint Programming
DM	Direct Method
EMU	Electric Multiple Unit
ERTMS	European Traffic Management System
GA	Genetic Algorithm
HMI	Human Machine Interface
HSM	Hot Start Method
HTA	Hierarchical Task Analysis
IM	Infrastructure Manager
KPI	Key Performance Indicator
LP	Linear Programming
LTT	Long-term Train Timetabling
MILP	Mixed Integer Linear Programming
MTU	Macroscopic Track Usage
NAHT	Network Adjusted Hourly Timetable
OD	Origin-Destination
OR	Operations Research
OSRD	Open-source Railway Designer
PESP	Periodic Event Scheduling Problem
PPT	PowerPoint
RMCP	Railway Minimum Clustering Problem
RU	Railway Undertaking
STDCM	Short-term Digital Capacity Management
STT	Short-term Train Timetabling
TCR	Temporary Capacity Restrictions
TMS	Traffic Management System
TOC	Train Operation Companies
TRL	Technical Readiness Level
TRSP	Train Routing and Scheduling Problem
TT	Train Timetabling
TTR	Timetable Redesign
WP	Work Package

3. Background

The present document constitutes the Deliverable D6.1 "Report on the description of algorithms for long-term timetabling, short-term timetabling and rolling stock planning" in the framework of the Flagship Project FP1– Motional as described in the EU-RAIL MAWP.

WP6 is mainly devoted to improving long-term and short-term timetabling of the railway network. In turn, this is obtained by developing advanced optimization models and algorithms for the generation, adjustment and assessment of timetables and related rolling stock planning. The algorithms will be demonstrated in decision support tools in WP7.

As a result, technical enablers TE3, TE4 and (partly) TE6 are implemented. In particular, we will refer to:

- TE3: Decision support for short term planning.
    a. Advanced algorithms for the adjustment of timetables to accommodate scenarios with additional/modified train paths, for example, due to change of (forecasted) demand.
    b. Advanced algorithms for the adjustment of timetables to accommodate scenarios with TCRs (temporary capacity restrictions), for example, planned maintenance.
    c. Algorithms that are available for usage both by timetable planners as well as traffic management staff.
- TE4: Train path and schedule optimisation methods and strategies.
    a. Algorithms for creating timetables at network, regional and corridor level, including an iterative approach in collaboration with timetable planners. On the station level, an aggregated view is used.
    b. Methods for station timetable planning based on a detailed representation of the station infrastructure. This includes the analysis of station infrastructure with respect to capacity utilisation and potential bottlenecks.
    c. Algorithms for creating a macroscopic timetable for a large network, e.g. one country (strategic, 5-10 years ahead) optimised on Generalised Journey Time (GJT). GJT is weighted travel time including waiting times and penalties for transfers.
    d. Algorithms for identifying critical network areas and links.
    e. Usability design principles for long-term timetabling decision support systems with optimisation.
- TE6: Integration of TMS with a) yard capacity planning and b) station capacity planning
    l. Compute an integrated rolling stock stabling plan that contains parking of trains, shunting between station and yard (and vice versa), and the capacity scheduling for cleaning and inspection activities in related tracks.

Regarding the distinction between long-term and short-term timetabling (following definitions from MOTIONAL deliverable D3.1), we define long-term timetabling as timetabling on the strategic and tactical planning level, and short-term timetabling as timetabling op the operational planning level.

Strategic planning deals with decisions about the design of the railway infrastructure, both the network topology (e.g., a new track in a station) as well as technical systems (e.g., a new safety system). This determines the capacity of a railway network. Given the railway network and given the line plan, tactical timetable planning consists of constructing the yearly timetable. This is done once a year and constitutes the basis for the next level of planning.

By operational planning we define the process of adjusting the current yearly timetable to create a new (possibly temporary) timetable that takes into account additional needs and constraints. This process can last from 1 day to 12 months ahead of the day of operations for the respective train. For instance, one may want to temporarily adjust the current timetable to account for maintenance activities which will block some tracks for a week. Or, one may want to add a special train to perform some extra service. We assume that the additional needs and constraints that are planned for do not derive from some unforeseen disruptive event (like a track failure) or from a primary train delay. These cases are handled by other railway processes, like disruption management or standard dispatching and are treated in WP10-18.

Finally, rolling stock planning is the process of managing the physical train units necessary to carry out the timetables. In WP6 we focus on two main planning problems: stabling and rotation. Stabling is the process of parking trains in dedicated yards (where also other activities, such as washing and maintenance, are carried out). In some cases, stabling also includes shunting, which is the set of (coordinated) movements of trains in order to be stabled or to perform other activities in yards. Instead, rotation is the assignment of train units to train services.

To achieve its goals, the WP is naturally divided into three tasks (6.2, 6.3 and 6.4) - besides the necessary preparatory work of task 6.1:

- Task 6.2, devoted to long-term timetabling
- Task 6.3, devoted to short-term timetabling
- Task 6.4, devoted to rolling stock planning

The algorithms developed in WP6 will be extended and demonstrated in the subsequent WP7. WP7 includes 10 demonstrators, which are connected to individual tasks and specific use-cases.
Demonstrators:

- 8.1: Demonstration of algorithms for generating strategic timetables (task 6.2, TE4). Use cases: UC-FP1-WP3-22, UC-FP1-WP3-23, UC-FP1-WP3-24
- 8.2: Demonstrate how a planner can interact with an optimisation-based timetable planning tool to resolve conflicts in the long-term planning process (task 6.2, TE4a, TE4d). Use case: UC-FP1-WP3-21.
- 9: Timetable optimiser and decision support system for adjusting the annual timetable on a line or network level based on the activities of subtask 6.3.1 (TE3). Use case: UC-FP1-WP3-25.
- 10.1: Demonstration of algorithms for planning of planned maintenance work for the entire Dutch network. Cancellations and alternative routes will be considered (task 6.3, TE3b). Use case: UC-FP1-WP3-19
- 10.2: Demonstrate the use of short-term planning algorithms for re-scheduling trains in case of TCRs at the Alnabru-Malmö line (task 6.3, TE3b). Use case: UC-FP1-WP3-18

- 10.3: Demonstrate the use of algorithms for inserting short-term train paths in a planned timetable (task 6.3, TE3a). Use case: UC-FP1-WP3-29
- 10.4: Demonstrate the use of short-term planning algorithms that identify and solve conflicts by different means (task 6.3, TE3c). Use case: UC-FP1-WP3-26
- 10.5: Demonstrate functionalities for short-term planning for rescheduling timetables in case of TCR and managing additions or modifications of new tracks on request (task 6.3, TE3). Use case: UC-FP1-WP3-27.
- 11.1: Demonstration of algorithms for rolling stock rotation (task 6.4, TE4a). Use case: UC-FP1-WP3-28.
- 11.2: Demonstration of algorithms for rolling stock stabling (task 6.4, TE6l). Use case: UC-FP1-WP3-20.

The methods and demonstrators will make use of the following Use Cases:

- **UC-FP1-WP3-18: Decision support for timetable planning with a temporary single-track section** (Track capacity restriction, TCR). It is time-consuming to make temporary timetables which in practice often results in the trains running according to the original timetable, with delays. With an algorithm that returns a new timetable given the new prerequisites, the timetable planner could get input to which decisions to make according to some KPIs.
- **UC-FP1-WP3-19: Decision support for constructing adjusted hourly timetables.** Preventive maintenance of the railway infrastructure necessitates the closure of some parts of the networks for a few days, forcing adjustments on the generic timetable.
- **UC-FP1-WP3-20. Decision support for rolling stock stabling.** The instance refers to the node Utrecht. Utrecht Central is the most central and busiest train station (both in terms of number of passengers and number of trains) of the Netherlands and has been found to be a hard nut to crack.
- **UC-FP1-WP3-21. Usability of an optimization-based decision support system for long term timetabling.** This use case considers a planner who wants to solve the conflicts for a train, or a set of trains, in the long-term planning process.
- **UC-FP1-WP3-22. Decision support for strategic timetabling.** This use case focuses on cyclic timetables with a cycle time of 1 hour. The objective is to minimize the total generalized travel time for all passengers together. This includes waiting time, in-train time and transfer time.
- **UC-FP1-WP3-23. Decision support for strategic station routing.** For a given macroscopic timetable, we will demonstrate an algorithm that finds a robust station routing or reports its inability to do so. Furthermore, when provided with predefined infrastructure variants containing small modifications, we will assess their benefit.
- **UC-FP1-WP3-24. Decision support for tactical timetabling.** This use case focuses on one or more lines in Norway. New timetables are generated from scratch using an interactive approach.
- **UC-FP1-WP3-25. Using timetable optimizer and decision support for timetable adjustments.** This use case considers the introduction in a given timetable of new or

changed paths and of new or changed TCRs. It also considers the assessment of the optimized plan before implementing it, and the synchronization with the TMS (Traffic Management System).

- **UC-FP1-WP3-26. Decision support for timetabling by conflict detection and resolution (CDR) algorithms.** This use case considers the introduction in a given timetable of temporary capacity restrictions and possessions of tracks, changes in existing train paths and the introduction of new train paths.
- **UC-FP1-WP3-27. Use of timetable optimizer and decision support for Short Term Period Timetabling.** This use case focuses on the Genoa SCCM area. Given the official timetable, it will look at scheduled works or accidental events that completely interrupt a stretch of line, also without predefined durations. It will also look at changes of the station layout and new train paths in the timetable.
- **UC-FP1-WP3-28. Automatic rolling stock planning.** This Use Case is complementary to the Use Case UC-FP1-WP3-24. For every new timetable generated, we will compute an optimal (or almost optimal) rolling stock plan, with the objective of minimizing the number of locomotives necessary to fulfil the timetable.
- **UC-FP1-WP3-29 – Optimized insertion of short-term train-paths into a predefined timetable.** This use case considers the insertion of new train paths into a given timetable.

4. Objective/Aim

The objective of WP6 is to improve long-term and short-term timetabling of the railway network. The activities aim at increasing infrastructure and transport utilisation capacity through optimised and robust timetables, synchronized with rolling stock planning. In WP6, advanced algorithms are developed for the generation and adjustment of timetables, and for rolling stock planning. These algorithms will be demonstrated in WP7.

For the 12 use cases UC-FP1-WP3-18 up to UC-FP1-WP3-29 described in D3.1, a dedicated algorithm/methodology is developed. These algorithms are described in Chapters 6-8. Each use case has its own section. Chapter 6 discusses the use cases (UC-FP1-WP3-21, UC-FP1-WP3-22, UC-FP1-WP3-23, UC-FP1-WP3-24) for long-term timetabling. The methods developed in this chapter are the result of Task 6.2, corresponding to TE4. In Chapter 7, we present the algorithms for the use cases (UC-FP1-WP3-18, UC-FP1-WP3-19, UC-FP1-WP3-25, UC-FP1-WP3-26, UC-FP1-WP3-27, UC-FP1-WP3-29) on short-term timetabling. This corresponds to Task 6.3 and TE3. Finally, Chapter 8 presents the developed methods for the use cases (UC-FP1-WP3-20, UC-FP1-WP3-28) on rolling stock planning (Task 6.4 and TE6).

The different sections are all set-up in a similar way. We start with a detailed description of the planning problem considered. Then we describe the algorithm/methodology to solve a use case. Note that we can only provide a brief description in the section itself, the (mathematical) details can be found in the different appendices. Next, we present test results of each algorithm on representative problem instances. These instances can either be artificial or real-world. To test the algorithms, a prototype implementation of each algorithm has been made. Relevant details about the implementation are provided in each section. Finally, we identify potential open issues for each algorithm, which can be further explored in WP7.

With the detailed description of each method in the appendix, the algorithms are reproducible. Most appendices will be submitted to scientific journals such that the dissemination of the project is maximized.

5. State-of-the-Art in planning and timetable optimization

As part of Task 6.1, we have performed an extensive literature review on optimization methods in railway planning. The focus of this survey is on long-term timetabling (Task 6.2), short-term timetabling (Task 6.3) and rolling stock planning (Task 6.4). We review the basic models and most important literature in the last decades and focus on applications at European railways, although relevant applications on other continents are reviewed as well. For a global survey on railway planning, we refer to Harrod (2012).

Our survey has been submitted to a scientific journal and is currently under review. It can be found in Appendix A. In this section, we summarize and connect it to state-of-the-art in practice as described in D3.1.

Timetabling

Railway timetabling can be classified in several ways. We distinguish between long-term and short-term timetabling, and by the type of network infrastructure (independent routes versus dependent routes).

In the literature, long-term timetabling refers to strategic and tactical planning in the classical decision levels. Short-term timetabling is generally known as operational planning. In long-term timetabling, strategic decisions about the design of the railway infrastructure are taken. This can deal with the network topology or with novel signalling and train control technology (such as ERTMS). Next, tactical decisions are made such as constructing the annual timetable. The annual timetable is constructed once a year. Short-term timetabling consists of the process of adjusting the current yearly timetable to a new (possibly temporary) timetable on one or several days in this year. This process lasts from 1 day to 12 months ahead of the day of operations. Reasons for adjusting the timetable are maintenance activities that block some tracks for a certain period or adding special trains to perform extra services.

The timetabling literature varies from a single independent route to a sophisticated network with multiple dependent routes. In the first case, the timetable of a particular route is independent of the timetable of other routes. This holds for instance for some dedicated high-speed railway lines. On the other hand, complex networks like e.g. in Germany or the Netherlands have a strong inter-dependency between different routes, e.g. to ensure connections between each pair of stations at a major station. These networks often have a cyclic or periodic timetable that repeats itself after a cycle period. Moreover, the timetable is often symmetric in both directions, which could result in so-called symmetry stations where all trains meet each other at e.g. minute .00 and minute .30.

There are two basic formulations for timetabling models for independent routes:

1. Basic big-*M* formulation (Section 2.1 in Appendix A),
2. Basic time-indexed formulation (Section 2.2 in Appendix A).

Both formulations have been applied to long-term and short-term timetabling. However, most applications are in short-term timetabling. As described in the MOTIONAL deliverable D3.1, there are hardly any applications of these methods in current railway practice.

For complex networks with dependent routes, the construction of the timetable is usually split into two steps. Firstly, a timetable on the network level is generated. This means that arrival and

departure times of all trains at all stations are determined. The infrastructure is taken into account by considering headway times between different events (arrival and departure of trains). Secondly, the different stations are planned in detail. Here all microscopic details of the station such as switches and signals are considered.

The first step is usually solved with a PESP formulation. In Section 3.1 of Appendix A, we provide a basic PESP formulation. The second step can be solved by a mixed integer programming formulation such as the one presented in Section 3.2 of Appendix A.

Both methods have been implemented in the DONS system developed in the Netherlands and were applied for the first time to construct the Dutch 2007 timetable (Kroon et al., 2009).

Finally, DB has developed a tool, called Click&Ride, where last-minute requests for freight trains can be scheduled. In the annual timetable, they have included paths for freight trains. When a freight train is requested from a specified origin to a specified destination with a certain start time, an algorithm computes the shortest path (earliest arrival time) and checks that this path is not used twice. In an earlier planning stage, DB uses a more advanced algorithm, where multiple freight trains can be scheduled at the same time. This algorithm uses column generation to solve a set packing problem. We refer to Dahms et al. (2019) for a more detail explanation of the algorithms used at DB.

Rolling stock planning

Rolling stock refers to the powered and unpowered vehicles required to move passengers or freight. It is the most expensive and critical resource for a railway undertaking. In the early literature on rolling stock scheduling, the rolling stock units often correspond to locomotives. In the last decades, passenger railway operators mainly use Electric Multiple Units (EMUs). An EMU is a powered train unit that can drive by itself and can be combined with other EMUs to form a longer train.

Based on the required train paths, decided in train time timetabling, that must be covered, the goal in rolling stock scheduling is to minimize the total cost to operate the rolling stock. This typically means that the rolling stock units should move empty (typically for repositioning) as little as possible. Another related criterion is the minimization of the number of required rolling stock units required to cover the transportation demands. Various technical and functional constraints need to be satisfied, in particular the available number of rolling stock units of each type and the eligibility of a rolling stock unit to operate on its assigned train paths.

Fioole et al. (2006) consider the scheduling of rolling stock. Multiple objectives are considered, in particular the maximization of service quality and reliability and the minimization of operational cost. The proposed model allows trains to be combined and split. The method developed in this paper resulted in an application that is used on a regular basis at NS (Kroon et al., 2009). Other successful approaches relying on hyper-graphs have been applied by DB (Borndoerfer et al., 2016). For a detailed overview on the literature, we refer to Appendix A.

Passenger train units have to be parked and serviced during the night. This problem is called rolling stock stabling, which is a relevant and hard problem to solve when infrastructure is limited. This is especially the case in dense urban areas. Several approaches to tackle his problem have been introduced in the literature (see Appendix A) but it has not been applied to railway practice yet.

Concluding Remarks

To conclude, both timetabling and rolling stock planning received much attention in the scientific literature since the 1990s. The initial approaches covered the tactical level, i.e. construction of the annual timetable or rolling stock plan. At that time, only small-scale instances could be solved. In the first decade of the 21st century some major achievements have been accomplished, for instance the use of OR methods for both timetabling and rolling stock planning to develop and introduce the completely new 2007 timetable in the Netherlands.

In the last 10 years, new models and solution methods have been introduced in the literature to tackle the strategic and operational planning level. However, to the best of our knowledge, these methods have hardly been applied in practice, in particular, in the context of complex networks with dependent routes. Therefore, we consider this as a major challenge for the European railway sector.

Optimization methods on the strategic level are becoming more important to ensure an efficient use of the railway infrastructure. This, because if we look at the coming decades where on one hand rail plays a major role in the European Green Deal, but on the other hand financial sources for large investments will be limited. Algorithms developed in Task 6.2 can help to tackle this strategic challenge.

At the operational level, many challenges are resulting from the ageing infrastructure that needs a lot of maintenance in the coming years. This is the major reason why many optimization algorithms will be developed in Task 6.3.

6. Algorithms for long-term timetabling

This chapter presents the work performed in Task 6.2 dealing with optimization algorithms for long-term timetabling. Recall that in the literature, long-term timetabling refers to strategic and tactical planning. Sections 6.1-6.3 deal with methods for strategic planning. In Section 6.1, we consider an Adaptive Large Neighbourhood Search (ALNS) algorithm to solve strategic planning at network level. The result of this algorithm can be applied to get insights in extensions or upgrades of the railway network. However, the detailed lay-out of the station infrastructure is ignored. Section 6.2 deals with new mathematical models to compare different stations lay-outs. In WP7, both algorithms will be connected such that strategic network decisions about the complete network can be taken. Section 6.3 presents an assessment methodology to compute the capacity of a station lay-out.

Sections 6.4 and 6.5 discuss the applicability of decision support methods for tactical planning, i.e. the constructing of the annual timetable. In Section 6.4, a solution approach for incremental train timetabling is presented. This algorithm can be used to generate a new timetable to adjust a previous timetable, e.g. the one of last year, or an infeasible timetable, e.g. the timetable of different railway undertakings which are feasible by themselves but contains conflicts between different railway undertakings. Finally, Section 6.5 discusses a systematic approach to ask planners which kind of decision support they need. This research is very useful for all algorithm developed in WP6, but it is specifically applied to tactical timetabling.

In WP7, the different methods will be demonstrated in Demonstration 8.1 and 8.2. Table 6.1 provides an overview of the different sections, linking it to the different subtasks in the Grant Agreement, the use case and the demonstration in WP7. Note that there is no use case and demonstration linked to Section 6.3. This research task is not continued in WP7.

Section	Subtask	Use case	Demonstration
6.1	6.2.1	UC-FP1-WP3-22	8.1
6.2	6.2.3	UC-FP1-WP3-23	8.1
6.3	6.2.3	-	-
6.4	6.2.1, 6.2.2	UC-FP1-WP3-24	8.1
6.5	6.2.1	UC-FP1-WP3-21	8.2

Each section starts with describing the background, then give a detailed problem description defining the use case in detail. In addition, we give a detailed description of the methodology to tackle the use case. For all use cases (except the one in Section 6.5), this is done by describing the algorithm. In Section 6.5, a qualitative research methodology is described. Finally, we validate the developed algorithms on representative problem instances (TRL 4 validation). The results are reported at the end of each section.

6.1 Algorithm for creating strategic network timetables

6.1.1 Background

For a passenger railway operator, the timetable is the backbone of its entire operation and its core value towards the passengers. The design of a timetable starts 5-10 years in advance and is traditionally split into several phases. The strategic phase focuses on line planning. A line plan is a set of routes with stopping patterns and frequencies with a highly aggregated macroscopic scope but without any departure and arrival times. In the subsequent operational and tactical phases, the detailed timetable is created by specifying the services' departure and arrival times, respecting all infrastructure limitations (such as track capacity or node capacity) and safety considerations (such as headway times between the trains). In addition, the rolling stock and crew schedules are computed. Many European countries operate a cyclic timetable. For instance, in the Netherlands, passenger services operate with a time period of 1 hour: the basic shape of a day's timetable arises from carrying out a one-hour timetable repeatedly.
The main goal of strategic planning is to maximize passengers' service quality by optimizing timetables. At NSR, a timetable's service quality is measured by the generalized perceived travel time, defined as the weighted sum of the in-train travel time, the adoption time, transfer waiting time, and transfer penalties, summed over all origin-destination (OD) pairs. In this project, we want to be able to optimize the service quality by adjusting the macroscopic timetable, essentially adding departure and arrival times to line planning's scope. By doing so, the proposed approach is able to evaluate the service quality more precisely. For example, an appropriate choice of arrival and departure times may dramatically improve a passenger transfer, or, on the contrary, the tool can derive that a certain combination of attractive transfers can be offered simultaneously.

A recent research project (see, Polinder et al. 2021) has investigated a timetable optimization model to compute the departure and arrival times. However, the solution methods still need to be improved to generate realistic solutions for real-world networks. To the best of our knowledge, few studies have successfully addressed instances involving large-scale networks, such as an entire country's railway system. Therefore, we design a hybrid algorithm to be able to compute high-quality solutions for large-scale instances, such as a country's entire railway network. Specifically, we propose an iterative algorithm that combines an Adaptive Large Neighbourhood Search (ALNS) algorithm with the MILP solver to compute high-quality solutions for the entire real-world railway network. We develop several passenger-centric operators in the algorithm to guide the optimization of timetables effectively by using passenger information.

6.1.2 Problem Description

This section describes the strategic periodic timetabling problem with integrated passenger routing. Our work focuses on developing a one-hour timetable and we assume that passengers travel through the shortest path when this one-hour timetable is carried out repeatedly.
We follow the problem setting of Polinder et al. (2021). The input of our problem contains the train lines. Each line is characterised by a geographic route, by the stations where the trains of this line call, and the hourly frequency. The pure timetabling part of our problem setting amounts to deciding the departure and arrival times of the trains at each station. These times are to be chosen in such a way that fulfils some given lower and upper bounds on the travel times between consecutive stations and on dwell times. The travel times are limited from below by the speed of the trains. The dwell times must be sufficiently long for the disembarking and embarking of the passengers. Commercial considerations set upper bounds on the travel times and the dwell times

in order to provide attractively short journeys to the passengers.

This research measures the quality of a timetable by its impact on the passengers. We are given the OD matrix that describes for any pair of stations how many passengers per hour want to travel from one station to another. We assume that the passengers are uniformly distributed over the hour: 1/60 of the hourly passengers show up at the origin station in the first minute of the hour, 1/60 of in the second minute, and so on. In particular, we assume the passenger demand does not depend on the timetable.

A journey of each passenger starts with showing up at their origin station. Then the passenger takes a sequence of train rides (having a transfer between any pair of trains), until the destination station is reached. We express the travel experience of the passenger by accounting for the undesirable aspects of the journey. The *adoption time* is the interval between showing up at the origin station and embarking the first train in the sequence. The *in-train time* is the sum of the minutes spent on the train rides of the journey. The *transfer time* is the sum of the minutes spent during the underway transfers.

The *perceived travel time* quantifies the passenger's experience. It is computed as the weighted sum of the adoption time, the in-train time and the transfer time. In addition, the perceived travel time includes a fixed *transfer penalty* for each underway transfer.

We consider in our optimisation model a simplified passenger behaviour model where each passenger looks at the timetable and takes a journey from the origin station to the destination station with the smallest possible perceived travel time. Thanks to the additive nature of the perceived travel time, the passenger's preferred path can efficiently be computed as the shortest path in an appropriately defined directed graph. We want to emphasise that passengers choose the same path if they belong to the same OD pair and they show up at the same time at their origin station. However, different showing-up times in an OD pair lead in general to different paths.

The strategic timetabling problem is formulated as follows. Given the set of lines and the OD matrix, we want to find a one-hour timetable that minimises the sum of perceived travel times of all passengers.

We employ an extended periodic event-activity network to capture the dynamics of trains and passengers within the railway system. The construction of this periodic event-activity network is provided in Section 1 in the Appendix B. Figure 6.1.1 shows the travel possibilities of a passenger both in a small railway system and the extended periodic event-activity network. The passenger wants to travel from station A on line 1 to station F on line 2. If the solid red path is chosen, the passenger boards the train at station A on line 1, travels through station B, alights at station C on line 1, transfers to line 2 at station C, and arrives at station F on line 2. The solid blue path has a transfer at station C. The passengers will choose the path with the smaller perceived travel time.

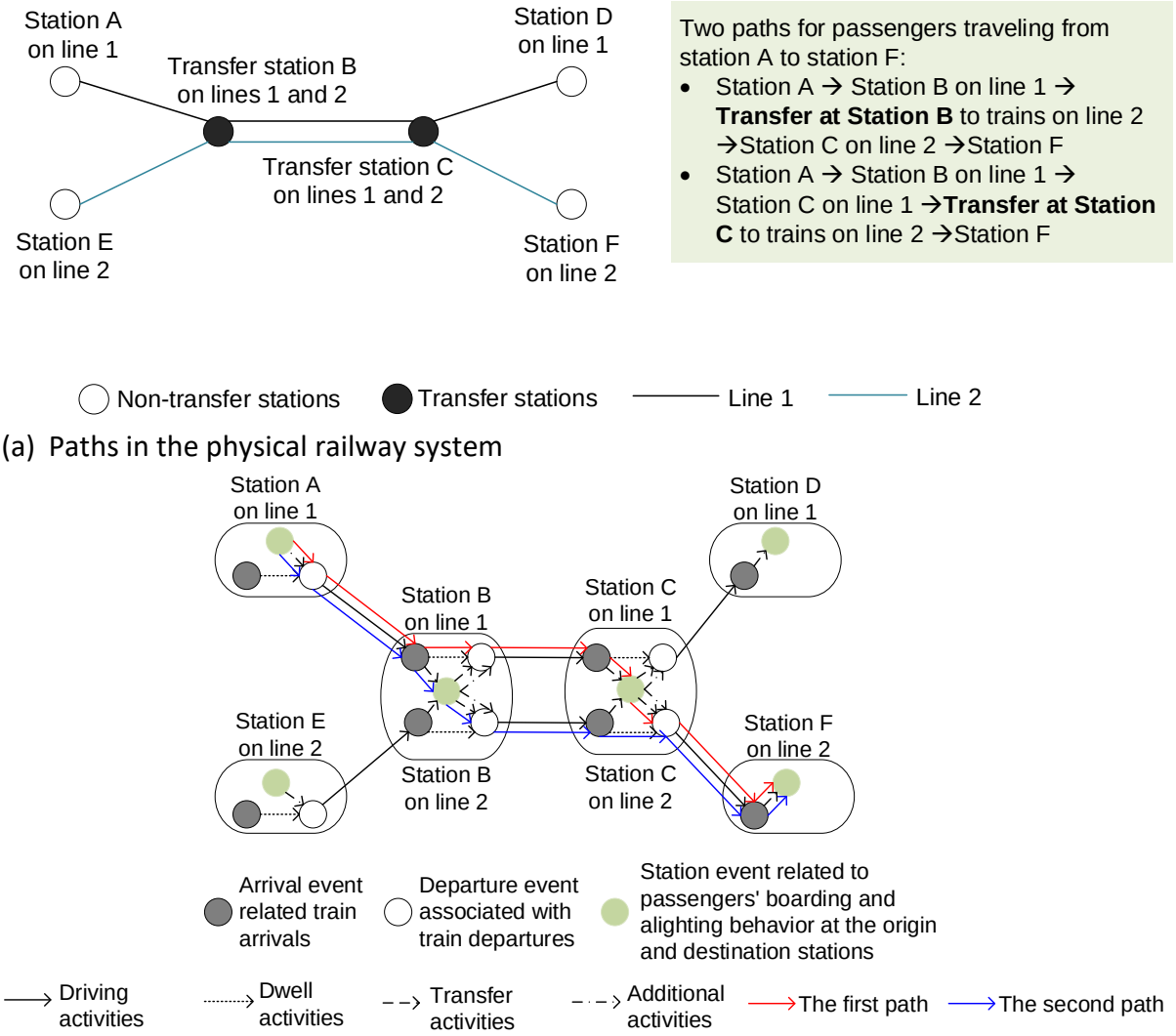


Figure 6.1.1: Illustration of passenger routing. The solid red and blue lines indicate two paths for passengers traveling from station A on line 1 to station F on line 2, where passengers make a transfer at station B or station C, respectively.

6.1.3 Solution methodology

For the developed mathematical formulation based on an event-activity network, we refer to Polinder et al. (2021), which is introduced in Section 2 in the Appendix B. This section focuses on presenting our solution methodology. In Section 6.1.3.1, we propose a decomposition framework for the model and an iterative hybrid algorithm that combines the adaptive large neighbourhood search (ALNS) algorithm with a standard MILP solver to find high-quality solutions within an acceptable computing time. In Section 6.1.3.2, we introduce the ALNS algorithm with passenger-centric operators in detail.

6.1.3.1 Decomposition approach and flow of the algorithm

In the proposed model, the high interdependence between timetabling and passenger routing significantly increases complexity, necessitating decomposition. The unified model's intricate constraints and computational intensity make it challenging to address it directly. Thus, it is divided

into two more manageable subproblems: the timetabling subproblem, which optimizes train schedules independently of passenger behaviours, and the passenger routing subproblem, which assumes a fixed timetable to optimize passenger behaviour. This division simplifies each component and facilitates the application of specialized solution methods tailored to the unique features of each subproblem, thereby enhancing overall model solvability and efficiency. For a detailed decomposition framework concerning the mathematical model, we refer to Section 3 in the Appendix B.

Based on the aforementioned decomposition framework, the ALNS algorithm is strategically designed to search for high-quality train timetables by solving the timetabling subproblem. To evaluate the timetables generated by the timetabling subproblem, the branch and bound method is utilized to solve the passenger routing subproblem. This process is repeated iteratively to search for high-quality train timetables and passenger route choices. Moreover, rather than exploring all neighbouring solutions of a given candidate, we introduce an outer loop that specifically optimizes the timetabling for one line by fixing all the other variables from the candidate solution. In this way, a small-scale model is generated and solved exactly using the MILP solver (or within a given computational time limit or an optimality gap limit). If a better solution is found, the candidate solution will be updated to perform the neighbour search again.

The proposed hybrid algorithm is denoted as *ALNS + MILP solver* and its overall flowchart is illustrated in Figure 6.1.2. We employ a MILP solver to call the branch and bound method to solve the passenger routing subproblem and the single-line timetabling problem with integrated passenger routing. The termination criteria for the entire algorithm are defined as follows: (1) reaching the maximum number of iterations, N^{max}; (2) the objective value remains unchanged for M consecutive iterations. In addition, we define the following alternating criterion to start the outer loop in addressing the single-line timetabling problem: it is applied after a specified number of iterations using the operators.

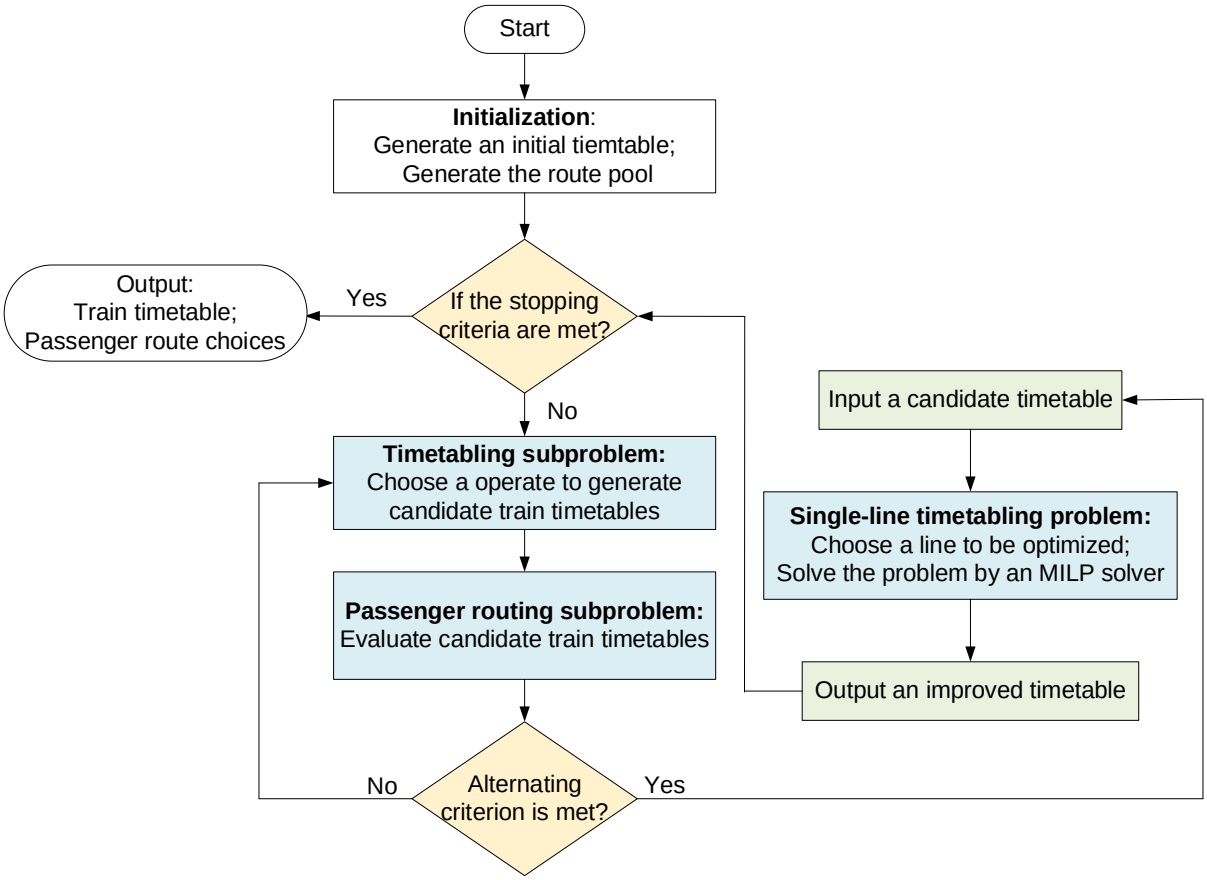


Figure 6.1.2: Overall flowchart of the ALNS + MILP solver algorithm.

6.1.3.2 Passenger-centric adaptive large neighbourhood search

The ALNS algorithm is an efficient heuristic algorithm, the core of which is the ability to use multiple operators in the same search process. The key to this algorithm is how to adaptively optimize the search strategy according to the information in solutions, thereby selecting the most appropriate operators to enhance performance. Initially, the ALNS algorithm assigns weights to each operator, dynamically adjusting these based on the operators' historical performance and frequency of use. For the integrated problem of timetabling and passenger routing, effectively using passengers' information to direct the search strategy for the timetabling subproblem is crucial for computational performance. This section provides a detailed discussion of the initial solution generation, operators, outer loop, the adaptive search strategy, and the simulated annealing rule.

Initial solution generation. The first step in designing an efficient ANLS algorithm is to find a good initial solution as the starting point of the search. Considering that an entire timetable for each train can be obtained when the departure time at the first station is determined, we design an encoding method based on the time of the first departure event with respect to each train. Thereafter, we generate an initial timetable satisfying all timetabling constraints according to the following method. First, we relax all variables to continuous variables, except for the timetabling-related variables. Then, we **solve** this relaxed model to optimality by a MILP solver. Note that this is a relatively easy problem, as no infrastructure constraints are considered in the strategic timetabling problem.

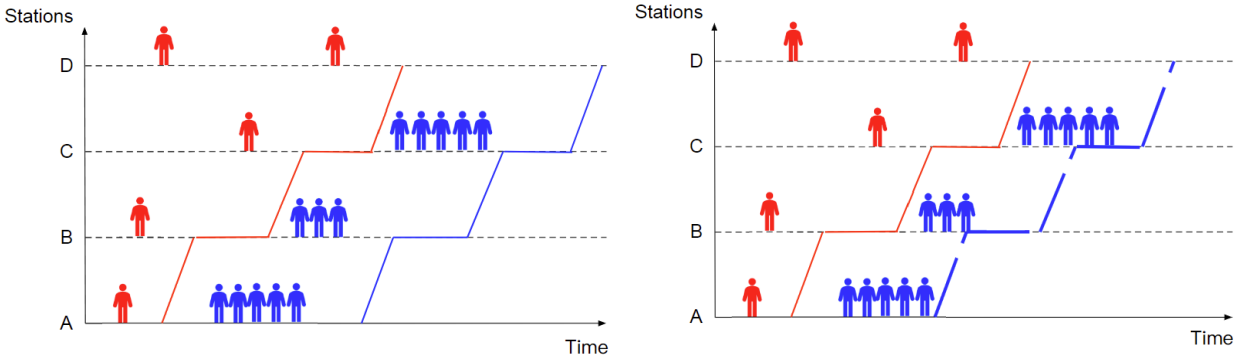
Operators. The ALNS algorithm uses a set of operators to generate new candidate solutions. In this project, we design the following three operators, with the last two relying on passenger

information from solutions.

(i) *Departure time operator.* The departure time operator aims to adjust the departure time of trains. First, we define the *line group* as a set of bound lines. We assign a specific number of lines to each line group, effectively generating various line groups. Upon randomly selecting a line group, this operator modifies the departure time at the first station of all trains on each line within this group by either advancing or delaying it. For the selected line group, departure times at the first stations are dynamically adjusted by β_1 minutes. This parameter remains fixed for the first N_1 iterations and subsequently decreases gradually as the iteration count progresses.

(ii) *Headway operator*. The goal of this operator is to adjust the headway on the congested lines. Specifically, we calculate the adoption time for passengers waiting for each train at all stations and then sort them. A selected number of trains, determined according to roulette rules, have their departure times advanced by τ_1 time intervals. This adjustment aims to reduce the headway between these trains and their respective preceding trains. For the trains selected by this operator, we dynamically adjust their departure times by τ_1 minutes. Initially, $\tau_1$remains constant for the first N_2 iterations and subsequently decreases as the iteration count increases.

The advantage of using the roulette rule is that they allow for a diverse but focused exploration of potential solutions. By assigning larger selection probabilities to trains with more waiting passengers, this rule efficiently guides the search towards more promising areas of the solution space while still maintaining the randomness necessary to escape local optima. This stochastic approach helps prevent the algorithm from becoming prematurely convergent on suboptimal solutions, thereby enhancing the robustness and effectiveness of the schedule optimization process.



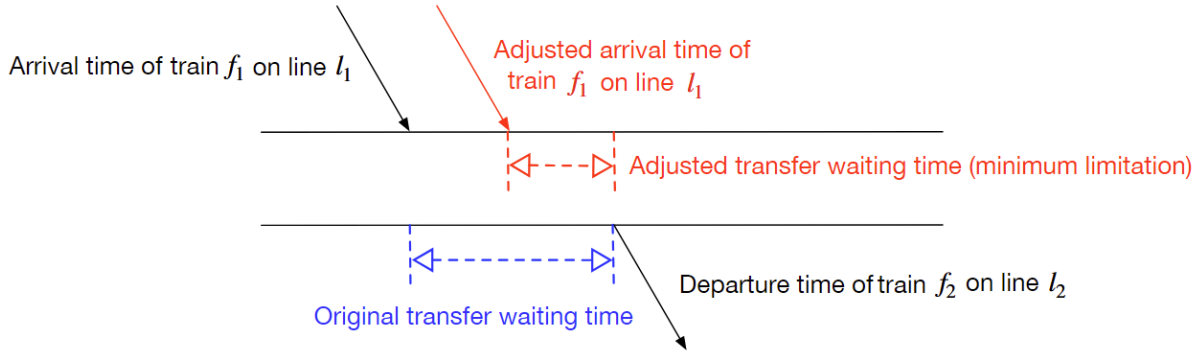
(a) The original timetable (b) The adjusted timetable

Figure 6.1.3 illustrates how a candidate solution is generated using the headway operator in the neighbourhood. The seed solution in Figure 6.1.3(a) represents a timetable for a line with a frequency of two trains. Notably, the second train experiences a larger waiting time of passengers. Therefore, as shown in Figure 6.1.3(b), using this operator will advance the departure time of the second train due to the high number of waiting passengers. In summary, the headway operator adjusts the headway by considering the passenger waiting times from the current solution, typically reducing the headway when there are more waiting passengers.

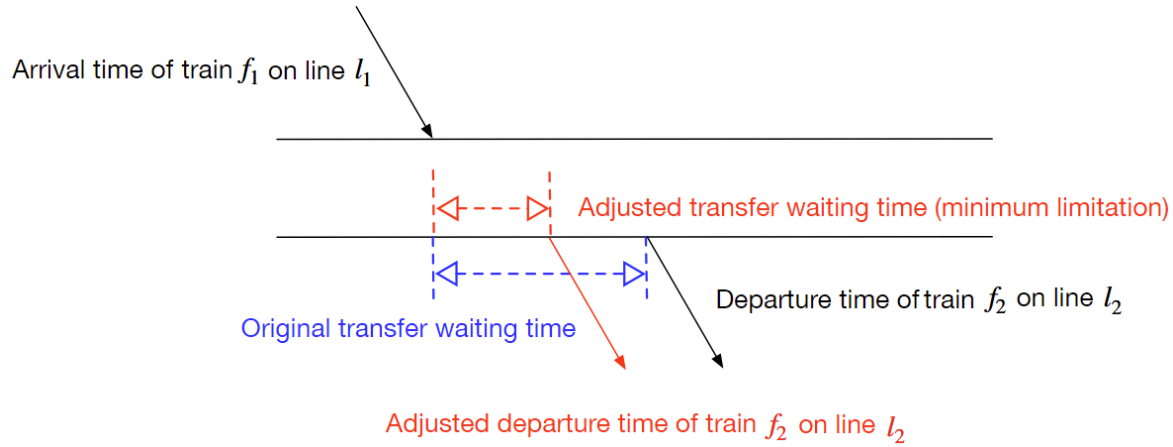
(iii) *Transfer operator*. This operator is designed to achieve seamless transfers. Specifically, we define the concept of a *transfer group*, denoted as $q_s = \{l_1, l_2, s_1, f_1, f_2\}$, where l_1 and l_2 represent the transfer-out and transfer-in lines at station s_1, f_1, and f_2 denote the transfer-out and transfer-in trains. We first calculate the total transfer waiting time

for each transfer group, which equals to number of transfer passengers and multiple transfer waiting times. Then, we sort the transfer groups according to the quantity of the total transfer waiting time. We select a subset of the transfer group following roulette rules, ensuring the larger the total transfer waiting time the higher the probability of the corresponding transfer group being selected. For the transfer group within the selected set of transfer groups, based on the principle of shortening the transfer waiting time between two lines, we randomly select the transfer-out or the transfer-in line for a transfer group. If the transfer-out line is selected, then the departure time of the transfer-out train at the originating station will be delayed by β_2 minutes; if the transfer-in line is selected, then the departure time of the transfer-in train at the originating station will be advanced by β_3 minutes. The exact values of parameters $\beta_2$and β_3 are adjusted dynamically. These two parameters are dynamically adjusted. It is a static parameter for the first N_3 iterations and then decreases gradually as the number of iterations increases.

Figure 6.1.4 provides an illustration of the timetable adjustment process. It demonstrates that the arrival time of the transfer-out train f_1 can be delayed to the earliest time slot which allows the minimum required time interval for a successful transfer. Conversely, the departure time of the transfer-in train f_2 can be advanced to a time slot where it aligns exactly with the transfer-out train's arrival time plus the minimum transfer time interval.



(a) The first possible adjusted result



(b) The second possible adjusted result

Figure 6.1.4: Illustration of the transfer operator. The arrival time of the transfer-out train will be delayed, or the departure time of the transfer-in train will be advanced to reduce the transfer waiting time.

Outer loop. The outer loop is designed to optimize the timetable for a single line. We input the

candidate solution and select a line according to the following rule. For each line, we calculate the demand whose routes pass this line. Then, we rank these lines by demand and employ the roulette wheel method to select the line for optimization.

Subsequently, we fix the variables for the remaining lines in the candidate solution, thereby defining a small-scale subproblem. This subproblem is then solved using the MILP solver, either within a predefined computation time limit or until a specified optimality gap is reached. If this process yields improved solutions, the candidate solution is updated accordingly, facilitating further iterations in the neighbourhood search. To further speed up the computation, we identify the OD pairs whose routes pass through this line and incorporate these OD pairs into optimization. After the timetable for the selected line is determined, we input all OD pairs and the newly updated timetable for all lines into the MILP solver and then evaluate its quality.

The adaptive search strategy. In each iteration, we select one operator (i.e., the departure time, headway, or transfer operators) to generate new solutions. We then utilize an adaptive search strategy to update the weights of these operators, which allows us to identify and select the most effective operator.

The strategies we implement for this process are detailed as follows. The scores and weights of operators are dynamically adjusted between 0 and 1, which ensures that the search direction continually progresses toward improved solutions. Initially, every operator starts with a weight of 1 and a score of 0. During each iteration, these scores and weights are dynamically updated based on operator performance. The detailed updating strategies are introduced in Section 4 in Appendix B.

The simulated annealing rule. After each iteration, the initial solution for the subsequent iteration is chosen based on the simulated annealing rule. This rule is inspired by the metallurgical process of annealing, which involves heating and then cooling a material to achieve its lowest energy state. The motivation for applying this rule in our algorithm is its capacity to accept not only improvements but also, crucially, occasional deteriorations in the quality of solutions. This feature helps to circumvent local minima and facilitates a more thorough exploration of the solution space, increasing the likelihood of reaching a global optimum.

6.1.4 Numerical experiments

To evaluate the effectiveness of our proposed algorithm, we conduct a series of numerical experiments using real-world data based on the Netherlands railway network. The experiments are performed on a laptop equipped with a 14th Gen Intel Core i9-14900H CPU, 16 cores, and 64 GB RAM, which operates at 2.2 GHz. We implement our algorithms in Java and use GUROBI 9.5.1 with default parameters.

6.1.4.1 Instances

The two instances that we use in this study are real-life instances of NSR, the largest operator of passenger trains in the Netherlands. The first instance is a network that contains four Intercity lines which physically form a triangle. The second instance is the 2024 Intercity network. In the remainder of this section, we describe the two instances in more detail. In both instances, we discretize a one-hour period into 600 intervals, each with a duration of 10 seconds. We use a synthetic OD matrix that estimates the passenger demand based on publicly available sources. We set the weights and fixed transfer penalty in the objective function according to Polinder et al.

(2021), as summarized in Table 6.1.1.

Table 6.1.1: Parameter setting of weights in the objective function.

Parameters	Values
Weight for in-train time	1
Weight for transfer time	1
Weight for adoption time	3
Transfer penalty	20 (min)

(i) A sub-network comprising four intercity lines. The first instance we consider in our project involves a sub-network of the Dutch railway system, comprising four intercity lines. The line plan for this instance is detailed in Table 1 in Section 5 in the Appendix B. This network consists of four bi-directional train lines, with frequencies of either two or one train per hour in both directions, summing up to a total of 12 trains. We analyse all OD pairs between any two stations within this network, totalling 1,316 pairs.

The cumulative percentage of the total passenger demand versus the number of OD pairs is presented in Figure 1 in Section 5 in the Appendix B. By analysing the passenger demand data, we observe that the distribution of OD-pair sizes is highly skewed: the 24 largest OD-pairs cover 35% of the total passenger volume. Given that GUROBI cannot yield a satisfactory solution within a 2-hour computation time for instances with all demand, we develop four case studies involving all four lines and various percentages of the passenger demand to validate the effectiveness of our algorithm compared to solving to optimality.

(ii) Intercity network. As the second instance, we consider the entire Intercity Network, which encompasses 74 stations, 50 bi-directional lines, and 80 services. This network serves 5,330 OD pairs. To reduce computational complexity, we allow each OD pair to utilize the three shortest physical routes.

6.1.4.2 Analysis of the effectiveness of operators and the outer loop

We first validate the effectiveness of the three proposed operators and the outer loop in the ALNS algorithm. To do so, we begin by employing the departure time operator alone. In the second stage of testing, we combine the departure time operator with the headway operator. Subsequently, we utilize all three operators at the same time. Finally, the integration of these three operators along with the outer loop is tested to evaluate the comprehensive impact on algorithm performance. This set of experiments is conducted on the sub-network with 50% passenger demand, where 15 routes for each OD pair are considered. The maximum number of iterations N^{max} and the number of iterations without improvements M are set as 150 and 50, respectively. If the outer loop is used, it is employed after every 10 successive iterations that involve the three operators. When using the outer loop, the termination criterion is set as an optimality gap smaller than 5%. Additionally, when optimizing a line's timetable, the OD pairs are ranked by passenger demand level, and the optimization process incorporates only the cumulative top 40% of the considered demand to speed up the solution process.

From the results presented in Table 6.1.2, we can observe that only using the departure time operator often results in convergence to a local optimum. The main reason for this limitation is that the operator relies heavily on randomness, which is insufficient to reach the global optimum. In contrast, the integration of the three operators with the outer loop enhances performance: the

objective value decreases from 125.10 to 124.21, and the number of iterations is reduced from 148 to 96. These results highlight the increased effectiveness of including the outer loop in finding better solutions. Consequently, all operators and the outer loop will be employed in the following experiments.

Table 6.1.2: Results of the algorithm among various settings of operators.

Operators	Perceived travel time per passenger (min)[1]	# of iterations	Computational time (s)
Departure time operator	127.72	57.00	60.03
Departure time operator; Headway operator	125.23	125.00	141.03
Departure time operator; Headway operator; Transfer operator	125.10	148.00	142.05
Departure time operator; Headway operator; Transfer operator; The outer loop	**124.21**	**96.00**	479.35

[1] We divided the objective function value by the number of considered passengers for readability purposes.

6.1.4.3 Best settings of parameters for the algorithm

Before testing the performance of the algorithm, we first determine the best parameters for the proposed algorithm (denoted as ALNS + MILP solver). We do so by running the algorithm for various values of their parameters on instances based on the sub-network and 50% passenger demand. In this series of experiments, we include 15 routes for each OD pair. The settings of using the outer loop is the same as those in Section 6.1.4.2. We vary the maximum number of iterations N^{max} and the number of iterations without improvements M. We report the detailed results in Table 6.1.3.

Table 6.1.3: Results of the algorithm for varying parameters.

Maximum number of iterations (N^{max})	Unchanged consecutive iterations (M)	Perceived travel time per passenger (min)[1]	Computational time (s)
150	30	125.26	254.77
150	50	**124.21**	**479.35**
200	50	127.37	403.05
200	100	127.19	401.92
300	100	126.78	911.44

[1] We divided the objective function value by the number of considered passengers for readability purposes.

We observe that the algorithm's performance does not consistently improve as the parameters increase, mainly due to its heuristic nature. However, the solution obtained with $N^{max} = 150$ and $M = 50$ are of best quality. Additionally, when N^{max} and M are set to 150 and 30, the algorithm achieves the shortest solution time. These results lead us to the conclusion that the best values for N^{max} and M in the following experiments based on the sub-network are 150 and 50, respectively.

6.1.4.4 Evaluation of the algorithm based on the sub-network

The performance comparison between GUROBI and our proposed algorithm (denoted as ALNS + GUROBI in this subsection) among various demand levels is presented in Table 6.1.4, where we set a time limit of 7,200 seconds for GUROBI. The settings for utilizing the outer loop are consistent with those described in Section 6.1.4.2, with a single adjustment aimed at accelerating the computations: the cumulative top 20% of the demand on the line to be optimized is incorporated for instances with 80% and 100% passenger demand. In Table 6.1.4, the first column indicates the involved percentage of the passenger demand. The second column denotes the solution method. The third to sixth columns represent the objective value, the computational time, and the bound obtained by GUROBI.

Table 6.1.4: Performance comparison between GUROBI and the ALNS + GUROBI algorithm.

Demand (%)	Solution method	Objective Value[1]	Computational time (s)	Upper bound[2]	Lower bound[2]
35	GUROBI	116.48	7,200.00	116.48	76.97
	ALNS + GUROBI	**115.49**	**309.70**	-	-
50	GUROBI	-	7,200.00	-	81.45
	ALNS + GUROBI	**124.21**	**479.35**	-	-
80	GUROBI	-	7,200.00	-	76.33
	ALNS + GUROBI	**147.37**	**1433.66**	-	-
100	GUROBI	-	7,200.00	-	66.56
	ALNS + GUROBI	**123.38**	**2,453.99**	-	-

[1] We divided the objective function value by the number of considered passengers for comparison purposes.
[2] Upper bound and lower bound represent the bounds generated by GUROBI after the 2-hour computation. Here, we also divided the obtained bounds by the number of passengers.

We can observe that the ALNS + GUROBI method consistently requires significantly less computational time compared to GUROBI alone. This reduction is particularly notable at a 50% demand level, where the computational time decreases from 7,200.00 seconds to 480 seconds, alongside an improvement in solution quality. As demand levels increase, the efficiency of our algorithm in reducing computational time becomes even more pronounced. For example, at 100% demand, while GUROBI fails to generate a feasible solution after a 2-hour computation, our algorithm can produce a solution within 2,500 seconds.

We next present the convergence curves of the algorithm for all instances across various demand levels in Figure 2 in Section 6 in the Appendix B. From these results, we can derive the following observations. (1) For all instances, there is a significant drop in the objective value within the initial iterations, indicating that the algorithm quickly finds high-quality solutions that considerably improve the objective value. (2) After the initial steep descent, the objective value stabilizes and shows minimal fluctuations. This trend suggests that the algorithm reaches near-optimal solutions quickly and then fine-tunes around these solutions.

6.1.4.5 Insights on the intercity network

We now report the preliminary computational results on the intercity network with all passenger demand across the entire system. We permit three routes per OD pair to reduce the solution challenge. The termination criteria for the algorithm stipulate either reaching a maximum of 150 iterations or an unchanged objective value for 30 consecutive iterations to balance the solution

time and quality. These criteria are based on the findings of the numerical experiments conducted above. The line-based optimization operator is used after 20 iterations with the other operators. In addition, when optimizing the timetable of a line, a cumulative 30% of passenger demand is included in the optimization process to speed up the solution process.

Based on the aforementioned settings, we first use the following approach to compute lower bounds to assess the quality of the solution. For any OD pair, we compute its idle perceived travel time as the in-vehicle time and the transfer penalty. Then we sum up to obtain the total idle perceived travel time of all passengers and regard it as a valid lower bound.

Table 6.1.5: Results of the average passenger travel times within the entire network.

KPIs	Values
Perceived travel time per passenger (min)	128.45
In-train time per passenger (min)	56.30
Adoption time per passenger (min)	18.70
Transfer time per passenger (min)	8.85
Transfer penalty per passenger (min)	7.20

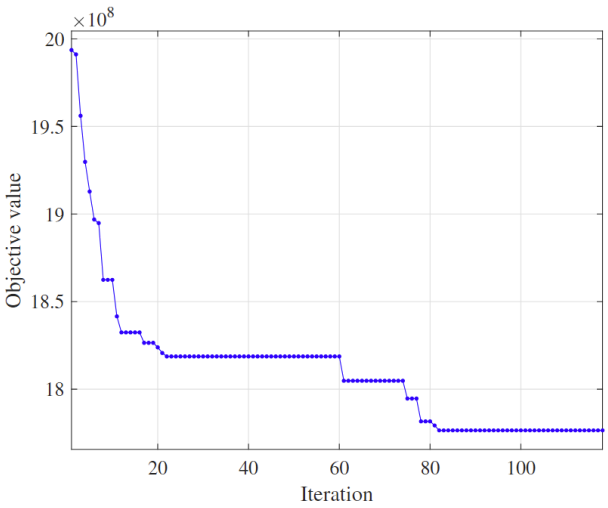


Figure 6.1.5: The convergence curve of our proposed algorithm for the instance based on the entire intercity network.

Next, we use the proposed algorithm to solve this instance. The detailed results are reported in Table 6.1.5. We can observe that the average in-train time per passenger is 56.30 minutes, the average adoption time is 18.70 minutes, the average transfer time is 8.85 minutes, and the average transfer penalty per passenger is 7.20 minutes. In addition, the convergence curve of our algorithm is presented in Figure 6.1.5. From these results, we conclude that our algorithm has good performance in terms of solution quality and computation time.

6.2 Robust station routing and strategic station planning

6.2.1 Background

The overall goal of long-term timetabling is the construction of a timetable for the operations on a railway network. Algorithms for this task usually operate on large networks (including whole

countries) and aim to improve the quality of the timetable across different quality metrics for operators and customers. However, due to the inherent complexity of this problem, algorithms for network timetable optimization are typically restricted to a macroscopic view where infrastructure limitations and potential train conflicts are only incorporated rudimentarily (e.g. through headway constraints). Therefore, these algorithms typically cannot guarantee the microscopic feasibility (and thus also the practical viability) of the generated timetables. This effect is particularly prominent in station areas, where many different lines intersect and the infrastructure exhibits a high degree of complexity.

As a second step in the planning process, it is, therefore, necessary to construct a local station routing which matches the network timetable. Beyond mere feasibility, an important goal in this step is finding a routing of high quality and attractiveness for railway operations. One central metric for the operational quality as well as the resulting passenger satisfaction is the robustness against delay propagation. While this naturally also depends on the disruption management and dispatching measures enacted during real-time train operations, a good initial routing plan, which takes delay robustness into account, may go a long way to enabling smoother operation.

During the annual timetabling process, the station routing task can be executed by experienced human planners, who may also fix infeasibilities through a combination of measures on the station and network level, for example by slightly modifying arrival or departure times or reassigning trains to different platforms. When comparing many different timetable-concepts in a longer-term setting and evaluating their robustness, however, the effort required for this quickly becomes a limiting factor.

Beyond the tactical timetabling step, a good station routing algorithm also has implications for the strategic task of station planning: Integrated clock-face timetables (e.g. the planned “Deutschlandtakt”) aim to facilitate better connection options between trains by concentrating arrivals and departures around the full and half hour marks. This has two major implications: First, it complicates the station routing task, as more train operations are concentrated in small periods of time. Second, the structural constraints of the timetable may – even after exhausting all options for finding a good routing – require modifications and enhancements to the infrastructure in order to enable the desired timetable concept. However, these are costly measures. Thus, it is important to offer support to infrastructure planners by determining limitations and evaluating both benefits and costs of possible modifications.

Against this background, DLR has expanded the research on robust station routing in MOTIONAL WP6. An algorithm was developed, which automatically finds train paths compatible with the arrival and departure times of a given network timetable and at the same time directly minimizes the expected delay propagation in the station caused by small random initial delays of trains. To do so, the delay propagation is modelled at a fine-granular infrastructure level, which additionally enables tracking on which infrastructure segments the delay is *propagated*, rather than only focusing on the occurrence of a delay. Using this, the algorithm generates infrastructure evaluation metrics that highlight potential bottlenecks and delay hotspots. For manually generated infrastructure variants, an evaluation procedure is proposed that analyzes the benefits realized by reducing the expected delay propagation. Additionally, a methodological approach is outlined to quantify and compare this benefit against the associated costs of infrastructure construction and maintenance per infrastructure variant.

We begin by describing the station routing optimization in the next subsection. Afterwards, we

present the methodology for the determination of delay hotspots as well as the assessment of infrastructure variants, which incorporates this station routing algorithm. We also describe how to monetize and compare delay robustness benefits and infrastructure costs. We apply the methodology to an exemplary test case on realistic data for the Dutch railway station of Arnhem. Finally, we give conclusions as well as an outlook on further research work to be carried out in WP7.

6.2.2 Optimal station routing minimizing expected delay propagation

This section gives an overview of the proposed optimization methodology for station routing. A detailed description including the mathematical model and further experiments has been submitted for publication (Widmann et al. 2024).

Delay robustness in station planning has received considerable interest in the literature. Many methods focus on the (weighted) direct buffer times between different train runs as an approximative metric, e.g. (Caprara et al. 2010; Dewilde et al. 2014; Burggraeve and Vansteenwegen 2017). However, this approach does not consider indirect dependencies between trains. Other methods proposed a delay propagation model that includes indirect dependencies, but either focused only on evaluating, but not optimizing the robustness of timetables (Delorme et al. 2009) or used recoverable robustness to give guarantees on worst-case scenarios (Caprara et al. 2014).

The goal of the optimization model is to directly incorporate and minimize the expected delay propagation caused by isolated primary delays. We propose a method that models delay propagation for small primary delays by simply passing the delay to the next train, without remedial measures like re-ordering or re-routing trains. Thus, the relevant quantity to model is the absorption capability between train pairs, defined as the smallest delay of one train that does not affect another train directly or indirectly. The difference between a buffer- and an absorption-based view is visualized in Figure 6.2.1: While there is no direct interference between train t_1 and train t_3, delay may still propagate indirectly through the dependencies caused by t_2. The value of the absorption is the length of the shortest path in the buffer graph.

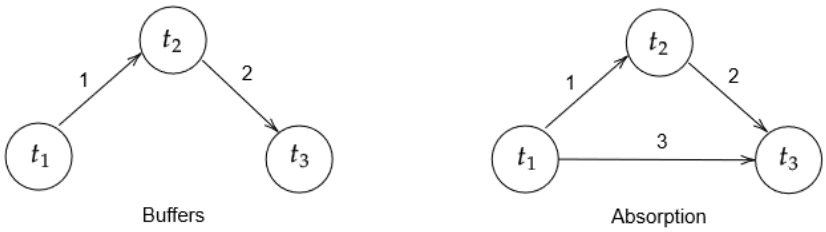


Figure 6.2.1: Difference of a buffer- and absorption-based view

Using this absorption, we can determine the expected delay propagation onto train 2 caused by a random primary delay D on train 1: It is the expected value $\mathbb{E}[\max\{0, D - a\}]$, where a is the absorption between the two trains. For the isolated primary delay D, we assume a modified exponential distribution capped by a maximal delay horizon H. Then, the expected delay impact of such an isolated primary delay on all other trains is the sum over the expected delays per train. The final objective is the mean impact caused by a primary delay on any one train (i.e. assuming that the isolated primary delay could affect each train with the same likelihood). In the following, we will describe how to build a mixed-integer linear program (MILP) that can optimize this

quantity.

As input for the optimization procedure, we utilize a network timetable which specifies arrival and departure times at the station as well as the entry and exit tracks of the trains in the analyzed station infrastructure. However, both the platform assignments as well as the precise routes through the station are not defined, yet. In the pre-processing phase, the algorithm first generates all possible physical routes through the station, eliminates implausible ones and determines the precise blocking times of trains running along them. To this end, suitable train simulation software generates running times that are as realistic as possible. This results in train path alternatives – combinations of a physical route together with the blocking times of a specific train run along it.

In preparation for the optimization model, we then split the station infrastructure into track sections which are separated by signals, switches, crossings or station borders (i.e. all elements at which the physical route or the blocking time of a train path may change) and enrich each train path with information about its used track sections and the associated blocking times, resulting in a list of track section reservations per train path. The train path alternatives, from which we want to choose the most suitable options, now form the input to the optimization model. To this end, we build a MILP model that ensures the feasibility of the train routing through a multicommodity-flow formulation combined with a delay propagation network.

Going into a more detailed description, we first focus on the feasibility component, which extends the ideas of (Caimi et al., 2011) by connecting the inbound and outbound paths: The train path alternatives are translated into a directed flow graph, with each track section reservation corresponding to an edge. We build up the graph starting from the “anchor points” of the track section reservations belonging to the planned stops at the platform. From there, we add edges by going forward and backward through the list of track section reservations of each train path. Two different train paths of the same train share graph edges as long as the track section reservations are identical, and diverge afterwards. In the resulting graph structure, the edges “branch out” from each platform track section reservations per possible train and platform combination. This has the benefit of implicitly making the choice of inbound and outbound path independent, which avoids the multiplicative number of variables when multiple inbound and outbound paths exist and reduces computational complexity significantly.

The resulting collection of graphs is merged into a flow graph by first linking a general source node with one source node per train, which in turn connects to the first track section reservation of each corresponding train path. The construction for the sink is analogous. Each edge is now associated with a binary variable, representing whether the corresponding track section reservation is chosen in the final routing, and standard flow conservation constraints are enforced at all nodes (except source and sink). This ensures that at most one path can be chosen per train and that this is a complete train path.

To ensure microscopic feasibility we employ the same clique constraints as (Caimi et al., 2011): For each physical track section, we look at the release time of all associated track section reservations over all train paths and form a clique for each of these time points, consisting of all track section reservations that are active at that time. We add constraints ensuring that only one of the binary variables belonging to each clique can be active at a time. (Caimi et al. 2011) prove that this formulation is valid and that it contains all maximal cliques, thus leading to a tight model. The objective function for the feasibility aspect simply maximizes the number of routed trains, with an objective value equal to the number of scheduled trains indicating that a microscopically

feasible routing for the proposed timetable was found.

While multiple train routings could be feasible, many will behave quite badly for small initial delays that occur in everyday operation. To rule this out, we look for a solution that minimizes the expected delay propagation caused by isolated primary delays, as described earlier. Thus, it is necessary to track the absorption between all pairs of trains for our objective. This can be accomplished by using a scenario-expansion approach, where each scenario tracks the absorption starting in one specific train.

For each scenario, we set up delay propagation constraints by associating each binary feasibility variable with an absorption variable, which are then linked in two ways: Within a train path, no absorption is considered, i.e., delay is simply passed on to the next chosen track section reservation. Conversely, if a delay would cause a conflict between two track section reservations of different trains, the original absorption plus the buffer between the trains is passed on, going from the track section reservation of the preceding train to that track section reservation of the following train which lies at the end of the block section preceding the conflict. This signifies that a train will wait at the last possible signal if it cannot reserve the next block section fully. Hence, this finer granularity allows us to model intermediate stops within the train path, compared to the formulation of (Caprara et al. 2014) that assigns delay to the full inbound and outbound path.

Note that this delay propagation model implicitly includes nonlinear constraints: The absorption at each edge should equal the minimal incoming absorption; and delay should only be transmitted along a link if the corresponding track section reservations are actually chosen – an implicitly quadratic constraint. The precise procedure for linearizing these constraints, as well as carefully choosing upper bounds for Big-M-constraints have high impact on the computational efficiency. We propose a problem formulation as one large MILP combining the feasibility and robustness components, which can be optimized efficiently for medium-sized stations using commercial solvers.

The result of this optimization step is a feasible, robust train routing which is consistent with the given macroscopic timetable. For validation and evaluation purposes, random primary delays are sampled according to the delay distribution. When using isolated primary delays, the average sampled delay propagation matches the objective. To simulate a more realistic case, a probability for (independent) primary train delays is assumed and multiple primary delays are sampled at the same time to determine their combined effect.

6.2.3 Evaluation of infrastructure modifications

The results so far give us the opportunity to find the optimal, most efficient way to use the existing infrastructure for a given timetable. This is an important step, as it improves the reliability of train operations without large associated investment costs. However, the results that can be achieved in this way may not be satisfactory for every scenario: In the worst case, it may not be possible to operate the desired timetable at all (in which case the algorithm will at least return a solution with as many trains as possible). Even if a routing with all trains is feasible, the expected delay propagation may not meet pre-defined reliability goals. Remedial actions can then only occur on the basis of modifying the timetable (which may not be possible due to mandatory constraints implied by the network timetable) or by adapting the infrastructure.

As construction work is expensive, it is important to be able to identify the most critical limitations

of the current infrastructure and evaluate possible alternatives. The representation and results of our optimization procedure can support planners in this aspect: The spread of delay is tracked at a high granularity in the track section model. In particular, to understand and find the bottlenecks in the infrastructure, emphasis should not necessarily be placed on where on the infrastructure trains are delayed, but rather where the *propagation* of delay occurs, i.e. those segments where the dependencies between trains exist and cause secondary delays.

These metrics, which are intermediate variables for our optimization objective, can be evaluated for the optimal routing, aggregated and displayed in an easily interpretable format. For the final routing, we consider one case per train, where it is delayed by our time horizon H, and track all occurring delay propagation according to our model. Then, as infrastructure criticality metric, we define the delay propagation per track section as the sum over all propagated delays starting in this section (across all cases). The resulting criticality metrics can be displayed in a heatmap over the infrastructure. For a visualization, see our exemplary application in the next section.

This information can be one supporting tool for infrastructure planners to identify key issues and derive sensible, possible infrastructure modifications. Due to the high individuality of each infrastructure layout and the multitude of spatial and geographical constraints, we do not believe it to be sensible to automate the process of generating or optimizing infrastructure variants. However, the evaluation of humanly generated variants once again falls into the scope of our methodology: We can repeat our methodology to determine optimal routings and corresponding robustness metrics. These metrics represent the expected delay propagation caused by small delays. By comparing them across variants, we can quantify how much delay propagation can be avoided by a proposed measure. As the small, everyday delays we analyze are unavoidable in current train operation, the predicted effect is expected to materialize in practice. Of course, there may be additional benefits which are not captured by our evaluation, e.g. robustness to infrastructure failures and disruptions, more options for train dispatching and a generally enhanced flexibility. However, our methodology can give a “lower bound” on the concretely realized benefit for the chosen timetable.

While we have described the methodology for a single timetable, in strategic planning it is to be expected that the timetable will change over the life cycle of the infrastructure. Naturally, if multiple different preliminary timetables are available (e.g. representing the forecasted operation at different points in the future), the presented analysis can be repeated and aggregated over these variants. In principle, it would also be possible to generate a family of timetable variants to analyze through slight variations of a generic expected train program. Such an analysis could aim to identify the flexibility of the infrastructure to operate varying network timetables. However, such a usage would likely require further improvements to the computational performance, as well as a methodology to randomly create sensible timetable variations. These developments are left for future work.

So far, we have focused on determining the robustness benefits that can be achieved through infrastructure modification. However, in order to evaluate the attractiveness of different variants of station layouts, both benefits and costs must be taken into account. In this case, it makes sense to monetize both in order to be able to compare them directly.
The variants differ in their investment and operating costs as well as in their life cycles. As the life cycles of railway technologies are very long and have to be amortized over years through the savings in delays, a dynamic investment calculation method is proposed that determines the Life Cycle Costs (LCC) as net present value.

An LCC approach is particularly useful for evaluating technological implementations with a long lifespan, such as technologies in the railway sector, which once implemented can determine the functionality of the infrastructure for several decades. This methodology has already been successfully applied in other Europe's Rail projects, for example in X2Rail-2 (Eckert et al., 2020).

It has the advantage that it relates payments that occur at different points in the life cycle to a defined point in time (e.g. $t = 0$) and thus makes the net present values of the different variants comparable. Otherwise, distortions would arise over such long periods of time if factors such as interest rates and inflation are not considered.

The LCC calculation considers not only the initial capital costs, but also all relevant costs during the life span of a product or technology, such as operating costs, maintenance costs, costs for replacement purchases, etc. When comparing different technologies, the calculation is carried out over at least the longest common life cycle. The reference period used for this analysis is based on the European Commission's recommendation for rail vehicles of 30 years (European Commission, 2014).

The value of money decreases over a period of 30 years due to inflation, compound interest effects and opportunity costs. In order to cumulate all costs incurring at different points in time over a long period of time (e.g. the service life of the railway infrastructure), the costs must be discounted and related to a specific point in time. The longer the time span of the life cycle under consideration, the more important the discounting of costs becomes due to the compound interest effect (Nellthorp, 2017).

For this analysis the Net Present Value can be calculated to the present time ($t = 0$) as follows:

$$NPV = \sum_{t=0}^{T} \frac{B_t - C_t}{(1+i)^t}$$

where B_t = benefits in year t, C_t = costs in year t, T = lifespan of the project, i = discount rate in year t. The result is the cumulated value of costs and benefits over the life span discounted to the date before the first investment is done.

The individual values for the equipment of the variants as well as values for interest and amortization periods may differ for each infrastructure company in a specific assessment. However, this does not hinder the methodology from being transferable across countries or application scenarios. In an example for Germany, values from DB InfraGO's catalogue of cost parameters (DB, 2016) would primarily be used to estimate exemplary orders of magnitude. These must be supplemented by assumptions based on expert estimates or empirical values.

The benefits that mirror these costs result primarily from the reduced delays, which must be valued with a specific amount. The values can be differentiated, for example, according to train type (passenger or freight, high speed or regional train, ...).

6.2.4 Test case

In the following, we demonstrate an example of the methodology for evaluating infrastructure

variants. We consider the station Arnhem Centraal in the Netherlands, whose infrastructure was modified in the 2010s to include – among other modifications – a tunnel in the southwest approach to the station. While we neither can nor intend to reconstruct the intricacies of the planning and decision process, we simply derive an instructive example from the general layout: The assumed starting point for the methodology matches the current station layout, but *without* the tunnel. As desired network timetable, we choose the public passenger timetable currently in operation. The detailed data about the current infrastructure was provided by ProRail and Nederlandse Spoorwegen, while the timetable was attained by transcribing the public passenger timetable available in the online journey planner for a daily peak hour in the 2024 timetable period.

We can now apply our routing optimization to determine an optimized station routing congruent with the network timetable. For these experiments, we assume primary delays distributed according to a modified exponential distribution capped at a delay horizon of 5 minutes. The results show that even without the tunnel, all trains in the timetable could be operated. For the optimized routing, we determine the delay propagation metrics for isolated and multiple delays. The results are included as the base case in Table 6.2.1. Furthermore, we evaluate infrastructure criticality as outlined previously and depict it in the station heatmap in Figure 6.2.2:

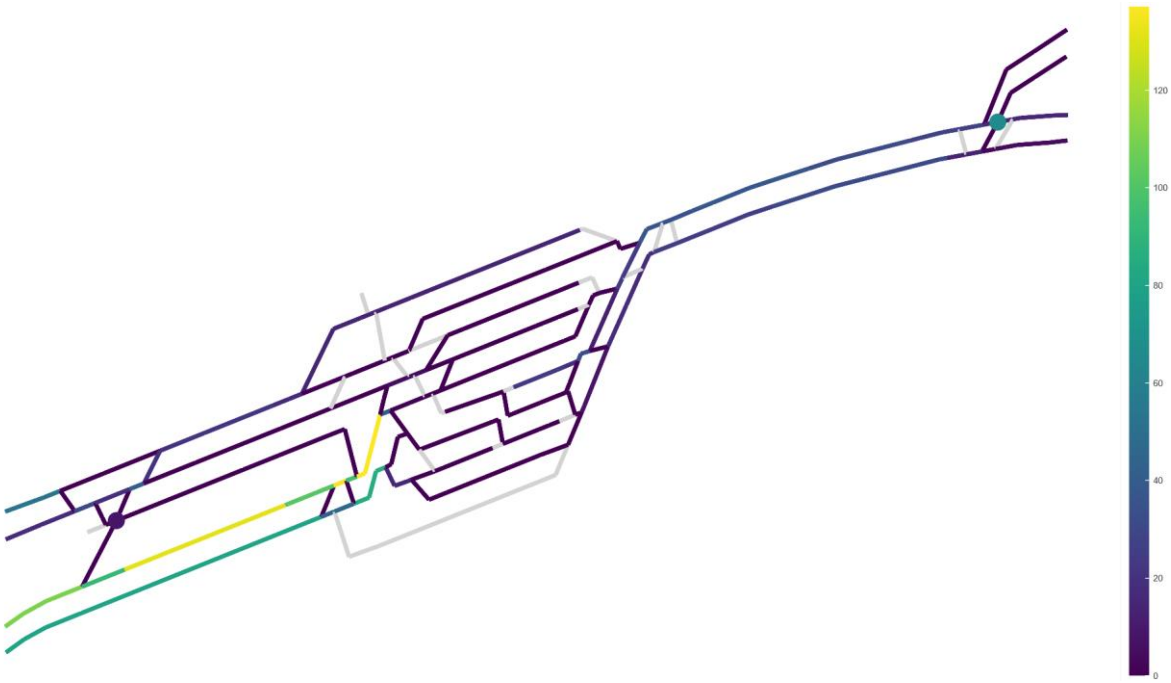


Figure 6.2.2: Heatmap of delay propagation for the optimized routing in Arnhem Centraal

We can clearly identify that the southwest approach to the station is the most critical part of the infrastructure. The only other moderate delay propagation occurs at the double-slip switch in the eastern approach, though at a much less significant level. Thus, focusing on the southwest approach, we see that the most critical segment actually lies closest to the platforms. Taking the whole infrastructure into account, it becomes apparent that this segment is the only way for trains arriving from a southwestern direction to reach the upper platform tracks. At the same time, it is also the primary way for trains to depart into the southwestern direction. Thus, it seems reasonable in this use case to explore infrastructure variants which enable alternative ways to reach the upper platforms.

The two potential variants introduced in this exemplary use case are the inclusion of an alternative pathway realized through additional same-level switches and crossings, as well as the aforementioned tunnel (which represents the currently existing infrastructure). We once again stress that this is an example meant to visualize the methodology and these variants should in actual application be generated by experienced infrastructure planners. We do not make any claims about the practical realizability of these modifications.

We repeat our routing optimization for the two considered variants. The time limit for each optimization run is set to 12 hours, which is acceptable as these evaluations are carried out only during long-term planning and are thus less time-critical. Then, we evaluate the average delay propagation caused by isolated or multiple primary delays through sampling. Additionally, to validate our methodology, we include an instance that uses the current infrastructure (identical to the "tunnel" instance) and fixes the platform assignments to those currently used in actual operation. The results are shown in the following Table 6.2.1.

Table 6.2.1: Optimization results for different infrastructure variants

Instance	Objective [s]	Sampled delay propagation [s]		Runtime [s]	Optim. gap	Objective after 3600s runtime [s]
		Isolated	Multiple			
Base case	77.1	80	616	8172	0.0%	77.1
Switches	67.6	70	544	43200	11.2%	68.0
Tunnel	45.7	46	383	43200	56.2%	48.2
Tunnel (fixed platforms)	48.8	49	410	–	–	–

We first note that the objective value of our optimization model approximately matches the sampled delay propagation caused by isolated primary delays. This validates the theoretical derivation that the objective encodes the expected delay propagation. Furthermore, we make the following observations on the model performance: While the base case is solved to optimality well within the given time frame, an optimality gap remains for the expanded infrastructure variants, which add flexibility and additional possible train path options and thus make the problem more difficult. This points towards a need for additional performance improvements (e.g. by improving the model formulation or employing a more advanced solution technique), which we aim to address in the further development in WP7.

Nevertheless, we note from empirical observations on this and other preliminary tests that it is in particular the proof of optimality as well as the last bit of robustness improvement that requires a long computation time. (Very) good solutions are obtained already early in the optimization process, as can be seen from the value of the best candidate solution after 1 hour of optimization. Furthermore, when comparing the optimized solution for the "tunnel" instance with the solution that uses the fixed platforms of the current public timetable, the optimized solution slightly outperforms the current timetable with respect to our optimization metric despite the large remaining optimality gap. As we can assume that the timetable designed by experienced planners is of high quality, this reinforces that the algorithm has found a good solution.

Comparing the variants, we see that both infrastructure modifications have a positive effect on

the expected delay propagation. The benefit achieved by building a tunnel is much more pronounced, reducing the average sampled impact of isolated primary delays from 80s to 46s, compared to a reduction from 80s to 70s in the variant with additional switches. Furthermore, these robustness results generalize to the case of multiple primary delays, which is arguably the typical case occurring in everyday train operations. Thus, while the interaction of multiple primary delays cannot be directly included in the objective due to the overall complexity of the problem, isolated primary delays prove a good approximative measure.

The general trend of this result is not surprising, as a tunnel naturally can eliminate more conflicts between train movements (which are now no longer passing each other on the same level) and thus reduces the dependencies between trains and the delay propagation more significantly. However, building a tunnel typically has much higher associated investment costs. This is where the methodology outlined in the previous segment can assist in comparing the monetary benefits and costs of the different variants. However, this analysis depends heavily on the worth of a minute of delay, as well as the infrastructure investment and maintenance costs. As these values are highly case-dependent and better known to the infrastructure companies involved in the planning, we refrain from computing them in our instructive example for now.

6.2.5 Conclusion and further work

In this contribution, DLR has worked on robust routing for railway stations congruent with a given network timetable. The problem is an important subproblem in the long-term timetabling process, to determine if (and how well) a network timetable can be realized in stations. We proposed an optimization model that directly minimizes the expected delay propagation caused by isolated primary delays of the trains. Reduced delay propagation should improve the smoothness of train operations and thus make them more attractive to railway users.

Furthermore, we identified an application of our methodology to strategic station planning to determine whether an (integrated clock-face) timetable can be operated, visualize which infrastructure components limit the robustness of the timetable and evaluate the performance of different infrastructure variants. We outlined a cost-benefit-analysis to juxtapose the robustness benefits against heightened infrastructure costs, which are important key performance indicators (KPIs) for decision-making in the strategic planning steps. The methodology was demonstrated on a realistic, instructive example derived from the station Arnhem Centraal.

As an outlook for WP7, we will try to identify opportunities for the connection of station and network level algorithms to generate timetables from scratch together with our partners in Demonstrator 8.1. This could, for example, take the form of an application of the tactical routing or strategic infrastructure planning process to optimized network timetables provided by our partners. An intended result of our demonstration will be to validate and improve the qualitative aspects of the local routings by providing them to expert timetable designers for review.

Other potential improvements to our methodology include the search for performance improvements to enable solving more and larger instances and shrinking the optimality gap. Furthermore, to avoid the current focus on a specific timetable in long-term planning, a direction for further work could be to generate multiple potential realizations of a preliminary network timetable (for an intended general line program), determine the resulting local routings and aggregate the results across timetable variants. Beyond the average value, the stability of these metrics across different routings could be a target for analysis.

6.3 Analysis of station capacity utilisation

6.3.1 Background

The capacity utilisation of a railway segment for a given timetable gives information on whether more trains can be inserted into the timetable or if the infrastructure is used as much as possible. The current standard for calculating capacity utilisation is timetable compression defined by the International Union of Railways (UIC) and outlined in their capacity leaflet (UIC, 2013). However, their recommendation for how to assess stations (nodes) is very basic and requires the decomposition of the station area into a switch area and a (platform) track area to be analysed separately, which may not be feasible for all types of stations. Another drawback of analysing several short sections of the railway is that the capacity utilisation will be underestimated compared to analysis of longer segments, thus providing too optimistic input on how many train paths can be added.

To overcome this shortcoming, Weik et al. (2020) presented a method to analyse stations in their entirety with UIC-based timetable compression. They also included a timetable-independent approach based on the Monte Carlo simulation of a representative ensemble of trains to investigate the capacity of the underlying infrastructure, with strategic capacity planning as a possible application. A variant of the method, where the train order at the station can be shifted if it leads to more efficient capacity utilisation, has also been presented (Johansson & Weik, 2021).

The infrastructure utilisation is only one aspect of the well-planned usage of a station. From the perspective of the passengers, which tracks the trains are assigned to are more important. Johansson & Peterson (2024) studied the track allocation at a railway station from the perspective of how many easy train changes at the same platform were enabled, how many conflicting train paths occurred for these connections and calculated the capacity utilisation for the original timetable and for two other simple principles for station track assignment.

6.3.2 Method

In this work, the aforementioned method by Weik et al. (2020) is applied to analyse the station capacity utilisation. In addition, the perspective of the passenger is accounted for similarly to the analysis by Johansson & Peterson (2024). Their analysis was made by simply manually counting the number of train changes, crossing train paths, etc., but here the method is improved by being implemented as a script to enable a faster and more accurate assessment, as well as making it possible to add functionality to refine the analysis. In this analysis, possible train changes where the passenger would be taking another train back in the same direction they just arrived from are not counted as a connection.

The implementation is done in Matlab and takes as input infrastructure and timetable data provided by the Swedish Transport Administration in the microscopic railway simulation tool RailSys (see, e.g., Bendfeldt et al. (2000)). The detailed modelling of the infrastructure and the timetable includes the station route, the block occupation, and the station track allocation for passenger trains with stops as planned by the Swedish Transport Administration when creating the timetable. The information is exported to Matlab for calculation of the station capacity utilisation, the total number of trains, and how many of them are stopping passenger trains. Moreover, the number of possible train changes is evaluated by counting the number of passenger trains that depart between 5 and 30 minutes after the arrival of a passenger train, and how many of these connections happen at the same platform, either with the trains using the same track or

the tracks that share a platform, is counted. Finally, of these possible passenger connections, the number of crossing train paths is also counted, i.e. the number of connections where the arriving train and the departing train at some point occupy the same infrastructure at the station. This measure is evaluated since it indicates a trade-off between easy train changes and the risk of conflicts and delays. If two trains are scheduled at a station with a small separation in time and to the same or adjacent platform tracks, they are to some extent using the same infrastructure at the station. If the first train is delayed, it can easily affect the second train and lead to secondary delays or announced track changes, making the connections more difficult for passengers. Moreover, from a dispatching point of view, it can be easier to minimise the number of crossing train paths by physically separating the trains more at the station, which creates a trade-off between easy dispatching and easy train changes.

6.3.3 Case study

The evaluation is performed as a case study of the Swedish railway station Halmstad C on the Swedish Western Main Line, see Figure 6.3.3. Halmstad C will be rebuilt starting 2028/2029 and the current level crossings, where passengers move between platforms, will be removed (Trafikverket, 2024). Analysis of the current layout and timetable of Halmstad C, represented by the timetable for a full weekday (24 h) in February 2022, is performed along with analysis of the planned new infrastructure and a possible future timetable (24 h) for 2040, which includes a forecasted traffic increase due to higher demand. The 2040 timetable can be considered realistic since it is based on the official demand forecast by the Swedish Transport Administration.

In the current layout of Halmstad C, the station has five tracks where track 1 is the leftmost track in Figure 6.3.3 and track 5 is the rightmost. Passengers access trains on tracks 1 and 2 from the platform between those tracks, while tracks 3 and 4 share another platform between them. Track 5 does not have a platform for passenger exchanges. To access the platforms, passengers must use level crossings from the station building side to the left of track 1. The bus terminal at the station, however, is located on the opposite side of the station, to the right of track 5, with no level crossing straight from the nearest platform to that side of the station. Passengers changing between train and bus must instead use the level crossings located at the southern end of the platforms to exit the railway station on the side of track 1, then walk north past the station building to a pedestrian bridge crossing all tracks and leading to the bus terminal. There is no access to the platforms from this bridge, as the platforms are too narrow to accommodate stairs or elevators.

In the station layout after the rebuilding, the platforms will be widened to allow the level crossings to be removed and replaced with crossings under and over the tracks with access to each platform. A platform will also be built for track 5 by the bus terminal. These changes are planned to make the station safer and easier for passengers to use, but will also result in a new layout of tracks and switches.

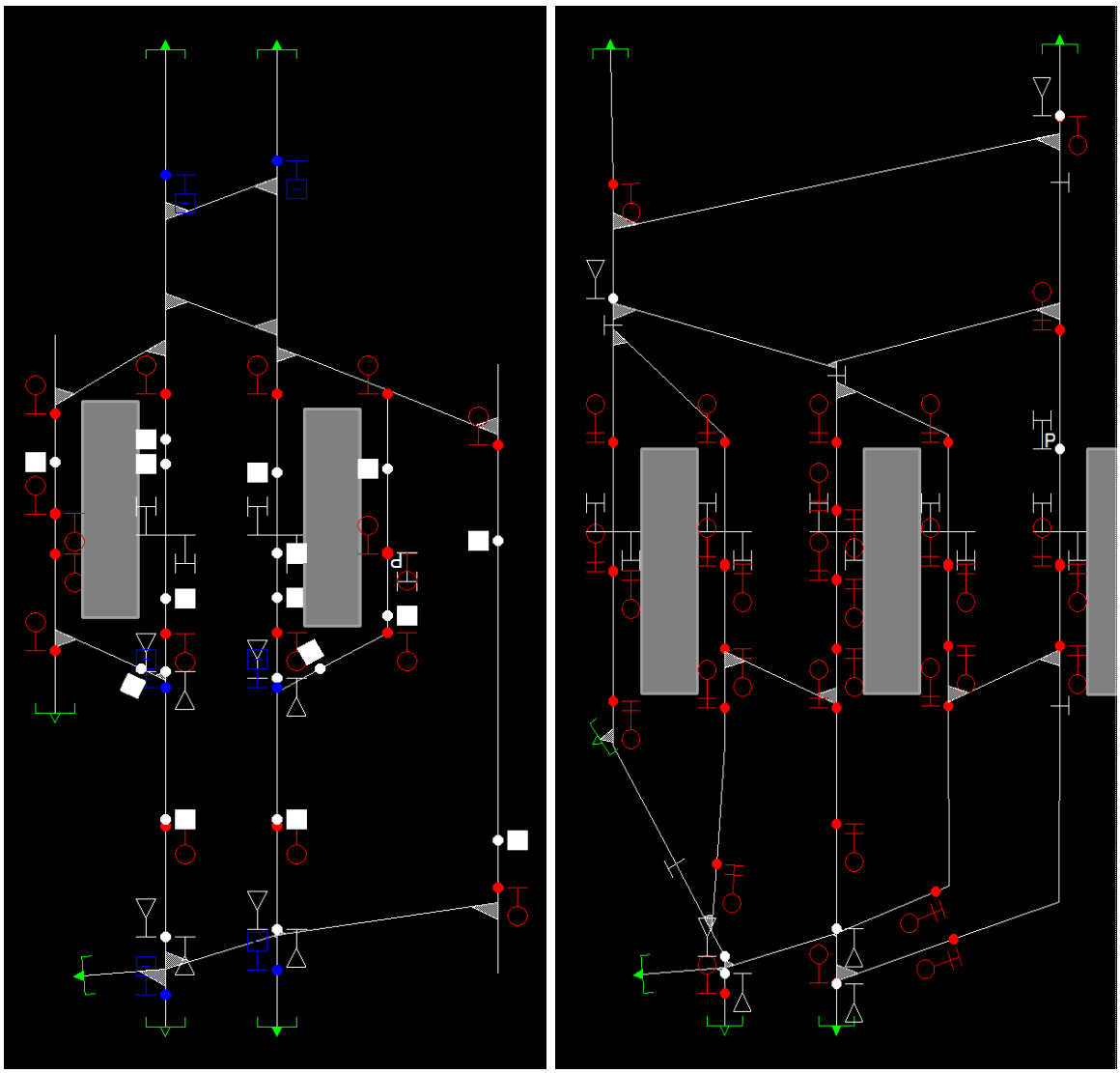


Figure 6.3.3: Current layout of Halmstad C (left) and layout after the future rebuilding (right).

6.3.4 Results

The results for the 2022 and 2040 timetables are summarised in Table 6.3.2. In the 2022 timetable, 81 trains were operated during the analysed day, while in 2040 the forecasted demand is more than doubled with 180 trains. The share of passenger trains is increased in 2040 where they constitute 93% of the total number of trains, compared with 88% in the 2022 timetable. Looking at the number of passenger connections, there are more than five times as many connections in 2040 than in 2024 while the number of passenger trains only more than doubles. This may indicate that the trains in the future timetable are better scheduled from the perspective of enabling train changes. While the number of possible connections increased more than five times, the number of connections with crossing train paths increased less, over four times compared to 2022, while the number of passenger connections from the same platform is 5.5 times higher in 2040. From a passenger perspective, the 2040 timetable seems to provide improvements to connection possibilities. However, the number of crossing train paths among the connections increased more than four times. As could be expected, this increase is correlated with a higher capacity utilisation of 50% in the 2040 timetable compared to just 18% in the 2022 timetable. Note that both the number of passenger connections with crossing train paths and the number of passenger connections from the same platform are overlapping categories and subsets to the total number

of passenger connections, thus they do not sum up to the total number of passenger connections.

Table 6.3.2: Results for the timetables from 2022 and 2040.

	2022 timetable	2040 timetable
Total number of trains	81	180
Number of passenger trains	71	167
Total number of passenger connections (5-30 min changing time)	92	487
Passenger connections with crossing train paths	64	275
Passenger connections from the same platform	46	253
Capacity utilisation [%]	18%	50%

6.3.5 Discussion

The analysis gives a comprehensive view of the capacity utilisation at Halmstad C today as well as after the planned rebuilding and increase of services. While the new infrastructure layout is already decided, the analysis indicates the feasibility of the forecasted increase in services, and of the allocation of the trains to station tracks, where the latter can give input to timetable planners who can adjust the track allocation in proposed future timetables. Worth noting is that the current level crossings at the station mean that passengers can change platforms relatively quickly, as long as they do not have to wait for a passing train. After the rebuilding of the station, this option will be replaced with a level-separated crossing, meaning that passengers will have to ascend and descend stairs or use the elevator when accessing and changing between platforms. This may increase the actual minimum time required to change between trains where a platform change is needed compared to the situation today.

As a general result, this study shows how capacity analysis can be applied in strategic planning when building or rebuilding stations. The presented methods can also be applied earlier in the planning process when a decision has to be made on which of a few different station infrastructure layouts should be built. If the infrastructure variants are modelled and some probable timetable variants, including station track assignment, are created, the station capacity utilisation can be performed along with the analysis of reliable train changes for the passengers. The same timetable can also be used in combination with different infrastructure variants, thus isolating the influence of the infrastructure. In terms of computation time, the Matlab scripts run in less than a minute on a standard laptop.

6.3.6 Future work

With the current implementation of the method, the analysis can easily be extended and further refined to better model relevant train changes. The first step will be to model the minimum required transfer times more accurately, so they differ depending on the considered pair of arrival and departure tracks. The current level of 5-30 min of transfer time to form a connection is only a first estimate. Further extensions could include considering changes to and from buses, along with evaluating how easy it is to reach the station building and exit to the city centre.

6.4 Conflict-based search algorithm for incremental timetabling

6.4.1 Background

The MOTIONAL project aims to develop new approaches to both short-term and long-term timetabling. Short-term timetabling typically concerns adjustments to existing timetables to handle e.g. planned maintenance work and new constraints and preferences from the train operators. In long-term timetabling (also called strategic timetabling), planners run a longer process lasting multiple years with the main purpose of investigating changes in the capacity of the network by adding and changing train services and extending and changing the infrastructure.

Typical optimization approaches for long-term timetabling start from a more abstract service specification, such as the number of trains per hour between pairs of major stations. The literature describes many cases of the fully automatic approach (e.g., Cacchiani and Toth 2018; Caimi, L. Kroon, and Liebchen 2017), where a program takes a service specification as input and produces a complete timetable as output. This approach has proven to have many challenges: (1) the planners that produce timetables usually have a lot of experience and domain knowledge which is difficult to transfer into a mathematical model, (2) the formal objective function is hard to define, and (3) solving from-scratch timetabling problems often becomes computationally impractical for real-world infrastructure sizes.

We approach the timetabling problem from a different angle. Instead of producing an “optimal” timetable from scratch, we focus on letting the user efficiently modify the timetable however they want, and then automatically restore feasibility. The process works as follows:

1. We start from an existing (possibly infeasible) timetable.
2. Route planners use a GUI to add, modify, or remove trains, and set preferences, resulting in a *reference* timetable representing the user’s goal.
3. Infeasibilities in the reference timetable are automatically solved by a routing and scheduling algorithm that find a feasible timetable that is close to the reference timetable.

This process relieves the route planners from tedious adjustments to a tentative timetable in order to make it feasible. We call this process *incremental timetabling*.

As described in Kloster et al. 2023, this approach avoids complex objective functions for e.g. defining the robustness, uncertainty in the passenger distribution or, in general, deal with stochastic constraints. Instead, it assumes that users, through expertise, will guide this process towards a good timetable.

On the other hand, the incremental approach does require a fast algorithm for finding a schedule that is as close as possible to the reference timetable, since this algorithm will be called repeatedly in an interactive user interface. Good solutions need to be produced within a few

minutes, at the most. This has prompted us to develop new algorithmic ideas for the "core" railway routing and scheduling problem with a fixed set of trains with partially fixed timetables.

6.4.2 Core scheduling algorithms with routing

The overall approach of incremental timetabling has also been explored by SINTEF in the MOTIONAL project for short-term timetabling (Section 7.1), there using an established MILP model as the "core" problem solver (Kloster et al. 2023). Learning from the literature on real-time railway traffic management using Alternative Graphs, it is typical that these types of solvers are reasonable scalable in terms of network size, but only when the routing possibilities are very limited (Mascis and Pacciarelli 2002a; D'Ariano, Pacciarelli, and Pranzo 2007; Pellegrini et al. 2015; Lamorgese, Mannino, and Natvig 2017; Leutwiler and Corman 2022).

For long-term timetabling, we are also looking into new ideas for the core scheduling solver, since the routing options for trains can be much more comprehensive in this context. Although the real-time traffic management literature suggests that time indexed MILP models may be more suitable for handling problems with comprehensive routing (Zwaneveld, L. G. Kroon, and Hoesel 2001; Harrod 2011), these approaches suffer from very limited scalability in terms of network size (Lusby et al. 2011; Reynolds et al. 2020).

A completely different branch of literature, concerning the joint pathfinding problem for multiple robots in a shared environment, has experienced very promising progress in the last decade. The problem definition called Multi-agent Path Finding (MAPF, see, e.g., Stern et al. 2019), and the algorithm known as Conflict-based Search (Sharon et al. 2015), have successfully been used in solved routing and scheduling problems for hundreds of agents on grids with obstacles. This problem has many commonalities with the railway routing and scheduling problem, especially when there are comprehensive routing choices. The section below describes how we have adapted Conflict-Based Search (CBS) to solve the railway routing and scheduling problem.

6.4.3 Conflict-Based Search for train routing and scheduling

6.4.3.1 Problem definition

We define the train routing and scheduling problem (TRSP) as follows. Let *T* be a set of trains, where each train $a \in T$ is defined by a directed acyclic graph of operations. Each operation has an lower and upper bound on its start time, a minimum duration, and occupies some part of the railway infrastructure (a track section). See Figure 6.4.1 for an example. A solution to the TRSP must specify the following:

- For each train, select a subset of the trains' operations that forms a path through its operations graph. We denote membership in this subset by the binary variable $x_a^o \in \{0,1\}$ for the operation *o* of train *a*. This defines the routing of the train, also called the *train path*.
- For each selected operation, specify the start time of that operation. We denote the start time $t_a^o \in \mathbb{R}$ for the operation *o* of train *a*. For operations for which $x_a^o = 0$, the corresponding t_a^o may remain undefined. This defined the *schedule* of the train.

For convenience, we will denote $t^{*o}_a$to be the end time of operation *o* of train *a*, which is defined to be equal to the start time of the immediate successor of *o* in the selected train path. A *feasible* solution to the TRSP must also satisfy:

- For each selected operation ($x_a^o = 1$), the corresponding start timet_a^o must be within the specified bounds for operation *o*. Also, we must have $t^{*o}_a - t_a^o \geq \delta$, where δ is the minimum duration of operation *o*.

- For any pair of operations o_1,o_2 from different trains a,b: if the set of infrastructure resources used by o_1 and o_2 overlap, the two cannot be executed simultaneously, i.e.:

$$(x_a^{o_1} = 0) \vee (x_b^{o_2} = 0) \vee (t^{*o_1}_a \leq t_b^{o_2}) \vee (t^{*o_2}_b \leq t_a^{o_1}). \quad (1)$$

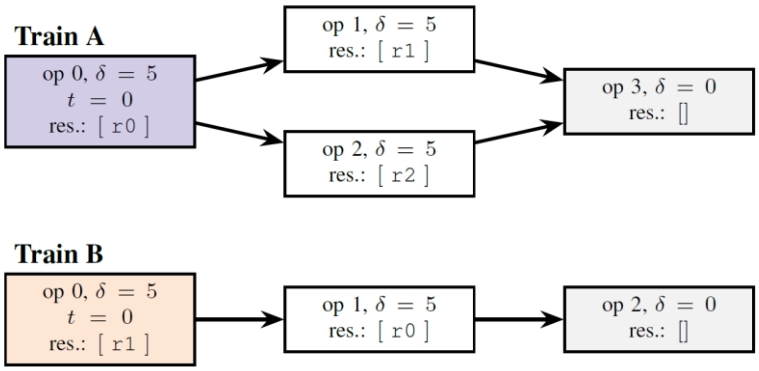


Figure 6.4.1: Two examples of train operation graphs. The boxes represent operations, labeled with a minimum duration δ, and a list of infrastructure resources r_0,r_1,... to which it needs exclusive access. The two left-most operations have lower and upper bounds both set to 0 on their start time t.

Resources in this context may represent locations on the mesoscopic level, such as station tracks and line tracks, or on the microscopic level, where each track circuits is a separate resource. The problem definition and the algorithm presented below is agnostic to this choice. In the long-term timetabling use case, we use resources representing mesoscopic locations.

To solve the TRPS, we can introduce binary variables $y_{a,b}^{o_1,o_2}$ for each choice of precedence constraint $t^{*o_1}_a \leq t_b^{o_2}$. Then, the model can be linearized into a so-called big-*M* model and solved using a MILP solver. The MILP solver will in effect be performing a branch-and-bound over binary choices *x* and *y*, and solving a tractable scheduling problem (the LP relaxation) for determining *t*. In broad strokes, this is the approach taken in Kloster et al. 2023 for short-term timetabling and in Mascis and Pacciarelli 2002b; Pellegrini et al. 2015; Lamorgese, Mannino, and Natvig 2017 for real-time traffic management. Since we know that this approach does not scale well with the number of routing decisions, we present the conflict-based search below as an alternative way of treating the four-way disjunction Eq. (1).

6.4.3.2 Conflict-based search

We adapt the conflict-based search algorithm of Sharon et al. 2015 to TRSP in the following way:

- First, plan each train as a solution to a *shortest-path problem with side constraints* on a time-expanded version of its operation graph. I.e., the graph is defined over nodes (*o*,*t*) where *o* is an operation and *t* is a starting time for that operation. Edges from an operation *o* to any successor operation *o'* in the operation graph are time-expanded into edges from each (*o*,*t*) to all (*o'*,*t'*) where $t' \geq t + \delta$. The side constraints are explained below.

- Combining the plans for the individual trains, check if these plans have any resource conflicts (i.e., check for violations of Eq (1)). If violations exist, we generate a disjunction of side constraints for the shortest-path problem.

- Run a branch-and-bound search with strong branching and probing to find the optimal choices in the disjunctions of side constraints.

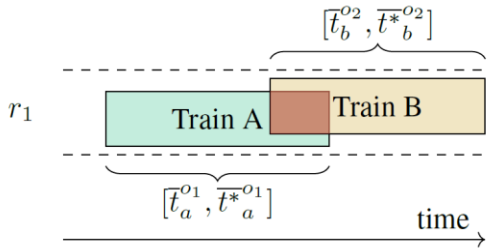


Figure 6.4.2: Two train operations, overlapping in time, and both requiring exclusive access to resource r_1.

This framework differs from the typical MILP approaches in that we do not use constraints of the type ${t^*}_a^{o_1} \leq t_b^{o_2}$, since they couple together the timing choices for two trains. Instead, we develop a different kind of basic constraints that involve only one train at a time. The key observation is the following. Consider the conflicting per-train solutions as shown in Figure 6.4.2, which violate Eq. (1). Now, consider one of the two alternative precedences between trains:

$${t^*}_a^{o_1} \leq t_b^{o_2}. \qquad (2)$$

We introduce an arbitrary constant $\lambda \in R$ and observe that the following is implied by Eq. (2):

$$({t^*}_a^{o_1} < \lambda) \vee (\lambda \leq t_b^{o_2}) \qquad (3)$$

If we do this for both of the two last disjuncts of Eq. (1), we get:

$$(x_a^{o_1} = 0) \vee (x_b^{o_2} = 0) \vee ({t^*}_b^{o_2} < \lambda_1) \vee (\lambda_1 \leq t_a^{o_1}) \vee ({t^*}_a^{o_1} < \lambda_2) \vee (\lambda_2 \leq t_b^{o_2}) \qquad (4)$$

for two arbitrary $\lambda_1, \lambda_2 \in R$. When we have a regular objective, i.e., each operation is scheduled to start as early as possible within its time bounds, we use $\lambda_1 = \overline{t^*}_b^{o_2}$ and $\lambda_2 = \overline{t^*}_a^{o_1}$.
The result of this exercise is that we now have a six-way disjunction, but if we regroup the disjunction, we see that each of the disjuncts are either path constraints or time bound constraints on an operation of a single train only:

$$(x_a^{o_1} = 0) \vee (t_a^{o_1} \geq \overline{t^*}_b^{o_2}) \vee ({t^*}_a^{o_1} < \overline{t^*}_a^{o_1}) \vee (x_b^{o_2} = 0) \vee ({t^*}_b^{o_2} < \overline{t^*}_b^{o_2}) \vee (t_b^{o_2} \geq \overline{t^*}_a^{o_1})$$

Left branch: path constraints and time bounds on train A Right branch: path constraints and time bounds on train B

The three left disjuncts of this constraint can now be grouped together and used as one of the *side constraints* in the shortest-path problem described above (and the same for the three on the right). This shortest-path problem with side constraints is a slight generalization of the Safe Interval Path Planning of Phillips and Likhachev 2011 where we allow arbitrary disjunctions of path and time bound constraints. The time-expanded graph is in principle very large, and the side constraints increase the computational complexity of the problem. In practice, the sparsity of side constraints combined with domination rules makes the problem very fast to solve using the multi-label Dijkstra's algorithm.

Grouping the side constraints into left/right branches, we organize the top-level search as a branch-and-bound algorithm. Note that instead of the four-way disjunction in Eq. (1), we have only a two-way disjunction in the branch-and-bound tree. The choice between re-routing and re-scheduling is delegated to the shortest path subproblem. We believe this feature to be the key feature of the CBS approach to TRSP and to explain the potential benefits over MILP approaches.

Note also that adding support for *temporary capacity restrictions*, such as a track being unavailable for some time interval, can be implemented by converting each restriction into a

constraint on each individual train, i.e., as a side constraint in the shortest path problem. The branch-and-bound search would not need to be extended in any way.

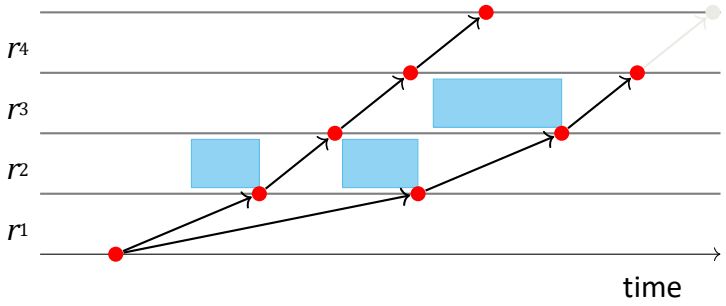


Figure 6.4.3: A generalized version of Safe Interval Path Planning is used to solve the shortest path problem with side constraints for the TRSP. Red dots indicate visited nodes in the time-expanded version of a train's operation graph. Note that the rightmost arrival node in resource r_4 can be pruned, as there are no further constraints in r_4 and the earliest arrival time suffices to find an optimal solution.

6.4.4 Preliminary computational results

As a preliminary assessment of the performance of the model proposed in the section above, we implemented a branch-and-bound with strong branching and probing over the sub-problems defined by the shortest path search for a single train, with disjunctive constraints on path and time bounds. The program was implemented using the Rust programming language running on AMD Ryzen 9 7900X 12-core desktop computer and using Gurobi v10.0 for a comparison with a MILP model. The problem instances were provided by Siemens Mobility and originally for real-time traffic management of a freight-dominated railway. Instances had 5-12 trains and an average of ~1400 operations per instance. We also tested an alternative graph MILP model with big-*M* for comparison. Note that MILP solver has built-in heuristics, while CBS is pure best-first aimed only at proving optimality.

Table 6.4.1 shows the computation times and the number of branch-and-bound nodes used to solve the instances using CBS and the big-*M* MILP. We see that the CBS-based algorithm shows very promising performance, solving most of the instances to optimality, where the Gurobi v10.0 MILP solver can only barely increase the bounds from the trivial root node bounds. We also observed that the solutions produced by Gurobi's built-in heuristics were of much worse quality than the optimal solutions produced by CBS, typically by a factor of 1.5x - 2.5x.

Table 6.4.1: Performance comparison on TRSP instances with a high number of routing choices.

Instance		Alternative graph MILP			Conflict-based search		
Trains	Ops.	Gap	Nodes	Time (s)	Gap	Nodes	Time (s)
6	443	0%	7594	12.1	0%	22	0.0
6	1754	82%	1	60.1	-	1501	60.0
6	1954	100%	1	60.0	0%	176	1.8
5	1854	100%	1533	60.0	0%	451	4.6
13	1437	-	1	60.0	0%	98	0.4
5	113	0%	0	0.0	0%	8	0.0
9	1718	83%	1406	60.0	0%	725	2.9
7	948	76%	1089	60.0	0%	131	0.4
8	1099	100%	4905	60.0	-	9814	60.0
8	2245	100%	7	60.0	0%	692	11.3
6	1416	93%	11018	60.0	0%	68	0.3
9	1975	90%	5679	60.0	0%	260	2.3

6.4.5 Remaining work

To bring this algorithm to productive use in long-term timetabling, we need to expand on the following:

- We initially assumed a regular objective, i.e., each train is scheduled as early as possible within its time windows. This assumption needs to be adjusted to an incremental timetabling approach, where we can also choose to go earlier than the scheduled time, at some cost. The choices of λ values in Eq. (4) would need to be re-evaluated, and the shortest-path algorithm could need to be substituted for another algorithm.
- The algorithm needs to be tested in an interactive context, using a system with a graphical user interface, and instance data suited for long-term timetabling. This context would have a lot more trains than in the experiments above, but we have also seen in Kloster et al. 2023 that the individual conflicts can be more independent than in a congested traffic management situation, and the problems are not necessarily slower to solve when there are more trains.
- In cases where the best-first search for the optimal solution does not terminate within the time limit, the algorithm at its current state of development does not provide any solutions. This can be mitigated by interleaving best-first search with a greedy depth-first search. A greedy depth-first mode would be likely to provide good quality solutions much faster than the time it takes to terminate the best-first search (which produces only an exactly optimal solution).

6.5 Usability of an optimization-based decision support system for annual capacity allocation

6.5.1 Background

The capacity allocation process can, roughly, be divided into 3 sub-processes: (1) different types of strategic long-term planning that is performed before the railway undertakings (RUs) have

submitted any capacity requests, (2) construction and establishment of the annual timetable based on the capacity requests submitted by the RUs, and (3) short-term (or ad-hoc) planning for adding or changing train paths and maintenance possessions after the annual timetable has been established. The focus of the work presented in this chapter is the construction and establishment of the annual timetable. However, some of the concepts presented are also relevant for short-term planning.

When constructing the annual timetable, planners modify train paths to resolve conflicts. There are many ways that the train paths can be modified to obtain a conflict-free timetable, but planners rarely have time to explore different solutions as there are strict deadlines. There are optimization algorithms that could be used to support the planners and use case UC-FP1-WP3-21 focuses on how to make an interactive usable optimization-based decision support system.

There have been several research projects on timetable optimization algorithms, but few are used by timetable planners in their everyday work. As an example, the Swedish IM has initiated many research projects on timetable optimization but has not yet successfully implemented any optimization-based support tool. One reason for this may be that the algorithms have been developed without considering their usability for the planners. Therefore, research focused on how to make optimization models useful for long-term timetable planners is needed.

6.5.1.1 Timetable planning in practice

Timetable planning must follow capacity management rules outlined by the EU. Currently, Directive 2012/34/EU is used, but this directive is expected to be replaced soon. Regardless of the directive, planners will have to, at some point, decide which capacity should be allocated to which train, and this brings along certain challenges. In this section we discuss some challenges currently experienced by planners. We expect that the challenges and knowledge obtained from interviewing planners in today's process to be mostly applicable also in future envisioned processes, albeit potentially with some changes or additional work needed.

One of the challenges when constructing the annual timetable is that the problem is so large that it has to be divided into pieces that different planners can work with. On a European scale, this may be the different countries being responsible for their part of the timetable, but also inside countries there are often many planners working together to construct the timetable. In Sweden, this division of labor is made in a geographical manner (see Figure 6.5.1), and train paths are passed between planners somewhat similarly to a relay baton. In fact, the train paths are said to be "handed over" from one planner to another. However, as opposed to a relay race where the next runner can focus on being ready to receive the baton and analyze the current status of the race, timetable planners are busy working with the "train path batons" that they have already received (or started with) while simultaneously trying to ensure that there will be capacity available for the remaining train-paths that have not been handed over yet. Large stations are often also handled by separate planners.

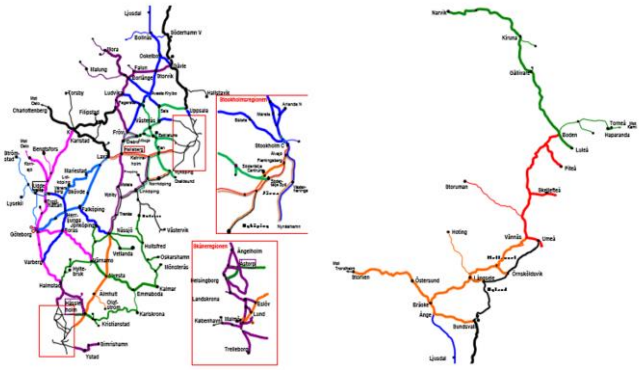
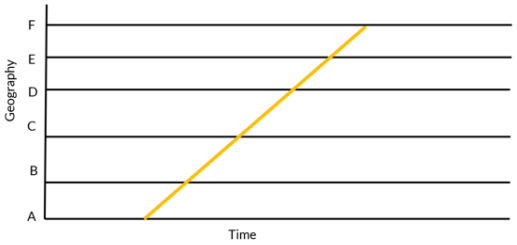
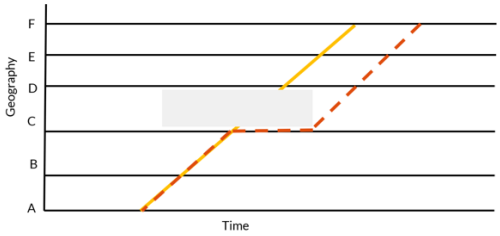


Figure 6.5.1: Each planner is represented by a color.

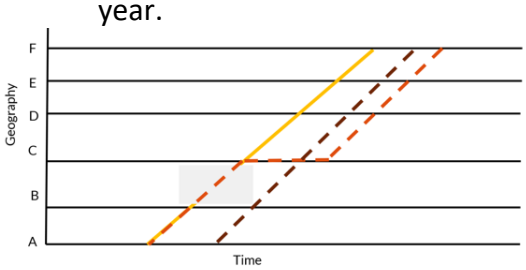
Another dimension that is hard to handle is the time dimension. Different trains will run during different days, and maintenance possessions will generally not be year-long, which means that each intended transport task will often be realized in different ways throughout the year. Often there is a "base train path" which is the one that is run most often, and then there are different variants of this train path where daily variations have been considered. The variants should preferably be as close as possible to the base train path at points of commercial interest (i.e. passenger transfers or freight terminals). Figure 6.5.2 shows an example of variants. Often, the planners focus on finding a base-variant for each capacity application first, and then focus more on the variants later in the planning process. It is also common that variants caused by maintenance possessions are handled in the ad-hoc process rather than when constructing the annual timetable.



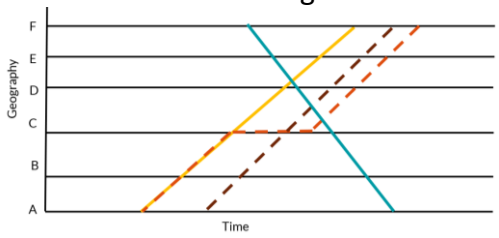
a) Base train path. Runs most days of the year.



b) Base train path and one variant that is run during a maintenance.



c) Yet another variant is needed for a different maintenance possession.



d) When adding a new train that interacts with all variants, the planners must decide how to handle this. They could, e.g., plan for a long stop for the new train that covers all variants or divide the new train into 3 variants as well.

Figure 6.5.2: Base train path and variants on a single line.

As already touched upon in the paragraph above, some planned times are more important than others. First of all, the arrival and departure times to points of commercial interest will be more important to the RUs than other planned times. We call these important points delivery commitments. Train paths could be changed in between delivery commitments without affecting the RU (as long as the robustness is not affected). In general, the core of the planning work when constructing the annual timetable is to find a feasible and robust timetable with train paths that fulfill the RUs capacity requests at commercially important points (i.e. delivery commitments) as well as possible. Note that we consider e.g. driver break times and vehicle turn around times as "commercially important" parts of the timetable, although they might be moved as long as the underlying requirements (break times, time needed for turn around) are fulfilled. Hand-over points, i.e. the geography-time point where a train path is handed-over to another planner, are another kind of "special points". In Sweden, hand-over points can be changed also after hand-over if needed, but it generally requires adaptation work from the subsequent planner(s).

Last but not least, capacity allocation is an authority decision. This means the planners need to explain why one solution was chosen instead of another. Consensus is often used as a way to find a timetable that is good from a socio-economic standpoint, where the underlying assumption is that if all commercial and public actors agree on a solution, it should be a good one. In some countries, there are also particular criteria that are used to decide between solutions if no consensus can be reached. The criteria aim to reflect the socio-economic value of different solutions, although there has been criticism against them (Broman et al., 2022).

6.5.2 Method

The focus of this study is usability aspects that come into play when a timetable planner working for an infrastructure manager wants to use optimization/automation to solve conflicts for a train, or a set of trains, in the annual capacity allocation process. The usability aspects were investigated using participatory design, hierarchical task analysis and semi-structured interviews. Further, while full implementation of all functions is not in scope of this usability study, some sort of "feasibility check" of the functions is included. To this aim, the M2-famework is used to provide an example of the functionality, illustrate different concepts and ways-of-working, or as base for explaining how the functionality could be implemented.

6.5.2.1 Participatory design

Participatory design, sometimes also referred to as "co-design", is a method and mindset used in usability research, user experience, and design, that emphasizes the active participation of users and other stakeholders throughout the design process. Participatory design aims to create solutions that better meet users' needs and expectations by integrating their perspectives and experiences directly in an iterative design process.

Participatory design is based on including different types of users to ensure that the varying needs of users are considered. For example, people with different job roles may have different needs, and a digital tool may therefore have to allow for different functions and visualizations to be useful (meet its goal) and user-friendly (be efficient to use) for the different people. In participatory design, the idea is that by including a diversity of users in the design process, more needs and perspectives can be considered. In a participatory design process, the idea is to involve and listen to the end-users, and let them influence the solution that they will later be expected to use. This should increase end-user acceptance and ownership of the solution. Openness, the opportunity to influence and shared responsibility during the design process also aims to create trust and

commitment to the change of working processes that may be needed when introducing new digital tools.

Different methods and techniques can be used to engage users and stakeholders. Typically, the needs are explored through workshops and interviews, in groups or individually, based on the users' daily work. Once a basic understanding of the work has been achieved, a first prototype is created which is then iterated further by allowing users to give feedback on what is being developed. By iterating prototypes, users can more easily relate the solution to their actual work tasks and understand how/if it matches their actual needs in the given organisation. By starting with simple prototypes, and then gradually increasing the level of complexity, development work can proceed more efficiently as less time is spent designing and developing technology that don't meet the users' needs and that therefore must be changed for the intended improvements to be achieved. Changes generally become more expensive and complicated to make the longer the development work has progressed. Therefore, the needs analysis phase, and early user and stakeholder involvement, is important to ensure the solution finally implemented is both useful and user-friendly.

As discussed above, participatory design has several advantages – the product or service become more tailored to the users' real needs, and acceptance and adoption rated can increase. Further, combining different perspectives and experiences from different job roles and people in the organization can lead to new and unexpected solutions. However, there are also challenges with participatory design, e.g., participatory processes can be time- and resource-intensive to organize and implement. It can also be a challenge for the person leading the work to ensure that all voices are heard, and that no person or group becomes dominating as this can skew the picture of needs. Different stakeholders may also have conflicting views and priorities, which can make it challenging to decide which direction the development work should take.

6.5.2.2 Hierarchical Task Analysis

Hierarchical task analysis (HTA) is a method from use-centered design where the tasks performed by users are systematically mapped and analyzed (Stanton, 2006). The method involves breaking down a complex task into its components and organizing them hierarchically, making it possible to see and understand each step of the process. In HTA, you start by identifying the main task and then break it down into subtasks and activities. Each subtask is further broken down until you reach a level of detail that can be sufficiently understood. Depending on the purpose of the analysis, different levels of detail may be appropriate. The hierarchical chart that the analysis results in helps to visualize the relationships between different tasks and provides an overview of different work steps.

The benefits of HTA include its ability to provide a clear overview of complex work processes, which facilitates the identification of improvement opportunities in the work process, but the analysis also provides a good basis for use-centred design. HTA can help create more intuitive and user-friendly systems by highlighting critical interaction points and user needs at each step of a task. HTA is particularly useful in areas such as system design, user interfaces, and training, where a deep understanding of user workflows is essential to design efficient and secure solutions. Part of the HTA for solving a resource conflict show in Figure 6.5.3.

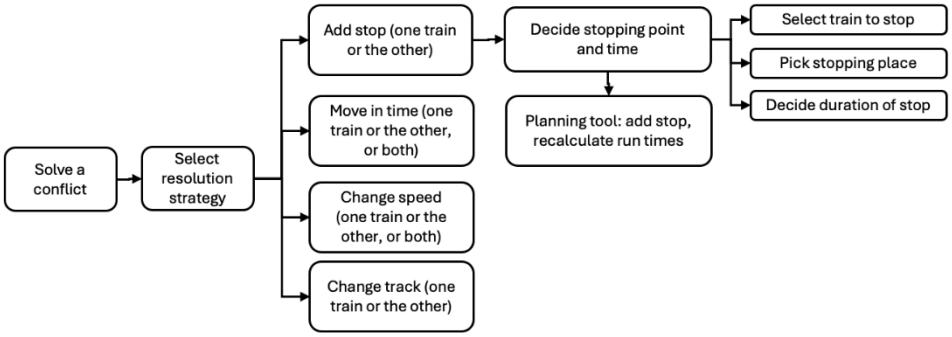


Figure 6.5.3: Part of the HTA for solving a resource conflict.

6.5.2.3 Semi-structured interview

Semi-structured interviewing is a qualitative research method that combines structured and open-ended follow-up questions to collect in-depth and detailed information from respondents (Lantz, 2013). This method is often used in various areas, including user-centered design, to gain a deeper understanding of participants' experiences, attitudes, and behaviours. In a semi-structured interview, the interviewer follows a predetermined list of questions or themes, but also has the flexibility to explore new topics and follow up on answers through impromptu questions.

Semi-structured interviews allow for a deeper exploration of topics compared to structured interviews, as it lets the researcher further investigate different, sometimes unexpected, aspects of the respondents' answers which can lead to otherwise missed insights. At the same time, it provides more structure than completely open-ended interviews, helping to keep the conversation focused and ensuring that important themes are covered. This balance between structure and flexibility makes semi-structured interviews particularly useful when the researcher wants to combine in-depth insights with a certain degree of comparability between different interviews.

To conduct an effective semi-structured interview, an interview guide with key questions and themes is required, but also that the interviewer is prepared to adapt to the respondent's answers and follow up with relevant follow-up questions. It is a great advantage if the interviewer has domain expertise to be able to identify and follow up on core problems with the respondent. The method also requires good communication skills and the combination of structure and flexibility makes it ideal for research projects where both depth and breadth of information are important.

6.5.2.4 The M2 framework

The M2 framework has been developed at RISE and can be used to automatically generate timetables with a variety of objective functions and levels of flexibility. For UC-FP1-WP3-21 in WP6, the framework may be used during later iterations of the participatory design process to illustrate different concepts and ways-of-working. The core of the M2-framework is a linear integer optimization model (see Gestrelius et al., 2015) with big-M constraints for e.g. precedence rules (similar to Kloster et al., 2023). M2 have many different objective functions that can be selected, e.g., minimizing the offset from applications, optimizing robustness metrics or minimizing the number of train paths that do not fulfil given criteria for commercial viability. There is also support for multi-objective optimization. When it comes to flexibility, time intervals that the trains must stay inside can be defined, and various decisions such as where trains should/shouldn't stop and/or interact can be fixed before the optimization is run. The optimization model is a macroscopic model but respects the number of tracks at stations and extra time needed for acceleration or braking in case of stops. Different train compositions can be handled, as well

as different requirements on, e.g., where trains may stop. The framework currently requires input data in a specific format (called TDEF), but M2 will be developed during MOTIONAL to also support additional data format(s). The M2 framework can also be used to export the timetabling problem in an extended railML format.

6.5.3 Research process and results

6.5.3.1 Participant introduction

All participants have been working as planners or coordinators in the long-term planning process or as capacity experts. The selection of participants was made by discussion with supervisors on suitable candidates, and the specific persons were invited to participate via e-mail. All participants were employees at Trafikverket at the time of the interview. The participants are introduced in Table 6.5.1.

Table 6.5.1: Participants.

	Description
LTP 1	Previous long-term planner in the south of Sweden (multi-year experience as planner), currently manager of timetable planners with relevant experience of supervising a team of planners. Previous experience as dispatcher.
LTP 2	Long-term planner in the west of Sweden. Multi-year experience as planner. Experience of sharing construction area with colleague. Previous experience as dispatcher.
LTP 3	Long-term planner in the north of Sweden. Multi-year experience as planner. Previous experience as dispatcher.
LTP 4	Long-term planner in the south of Sweden. Multi-year experience as planner.
LTP 5	Newly employed long-term planner.
CE 1	Capacity analysis expert. Knowledgeable in optimization.
CE 2	Capacity analysis expert. Knowledgeable in the current Planning System.
CE 3	Capacity analysis expert in strategic traffic planning, previous experience as timetable planner, works with TTR.
CE 4	Manager for strategic planning.
CE 5	Capacity analysis expert in strategic traffic planning, previous experience as timetable planner.
CE 6	Capacity analysis expert in strategic traffic planning, previous experience as timetable planner.
NC1	National coordinator. Responsible for station planning and ad-hoc planning.
NC2	National coordinator. Responsible for annual timetable.

The iterations performed and planned during WP 6 and WP7 are outlined in Table 6.5.2, including which participants took place in each iteration during WP6. White columns are iterations that have already been performed, and grey columns are iterations that will be performed during WP 7.

Table 6.5.2: Iterations.

Iteration 1	Iteration 2	Iteration 3	Iteration 4
Input: interview guide	Input: HMI sketches and PPT-prototype	Input: Interactive prototype and examples	Input: Updated interactive prototype and examples
Output: Understanding on	Output: Information on usefulness and	Output:	Output:

ways of working, list of tasks or challenges where optimization may be useful	suggestions on how to improve the presented functionalities, prioritization of suggested functionalities, understanding of potential algorithmic problems.		
Semi-structured interview with planners focusing on their current work: LP1, LP2, LP3, LP4 [1]	Semi-structured interview with planners focusing on quality metrics that may be useful in their current work: LP1, LP4, LP5 [1]		
Semi-structured interview with staff from the expert unit "Railway Capacity Center" on optimization functionalities that may be beneficial: CE1, CE2, CE3, CE4, CE5, CE6	Semi-structured interview with staff from the expert unit "Railway Capacity Center" on the quality metrics and their usefulness [1]	Semi-structured interview with staff from the expert unit "Railway Capacity Center" on optimization functionalities in the interactive prototype and examples that may be beneficial	Semi-structured interview with staff from the expert unit "Railway Capacity Center" on optimization functionalities in the interactive prototype and examples that may be beneficial
	Presentation of HMI sketches and discussion with WP 6 parties	Presentation of the interactive prototype and examples and discussion with WP 6 parties	Presentation of interactive prototype and examples and discussion with WP 6 parties
Semi-structured interview with staff from the coordination unit "National capacity optimization" focusing on their work and quality metrics that may be beneficial: NS1, NS2[2]			
HTA based on steps taken in the currently used planning tool, CE 2			

[1] Work also financed by Trafikverket grant TRV 2022/69851.
[2] Work also financed by Trafikverket grant TRV 2022/69851.

6.5.3.2 Iteration 1

Interview guide and result synthesis

The overall aim of Iteration 1 was to get an overview of their work over the course of the annual allocation period. The interview guide consisted of a short introduction to the project(s), a set of questions that the respondents were asked to "keep in the back of their mind" as well as an online white board with (1) a timeline of the annual allocation process, (2) a normal day at work and (3) communication graph. The timeline of the annual allocation process was based on the 2024 process deadlines outlined in the Network Statement from the Swedish Transport Administration and has been included in Figure 6.5.4. The normal day at work was simply a line that started with "arrive to work" and ended with "leave work". The interaction and communication graph is included in Figure 6.5.4. Some examples of communication/interaction were included in the communication/interaction graph to be used as a starting point for discussion. In the actual graphs used the examples were written in the boxes, but to increase readability they have been included as separate text in the figure. During the interview, post-it notes were added to the pictures with tasks, communication needs, comments and challenges.

During the interviews with timetable planners all pictures were used, although most time was spent on the annual overview followed by the communication and interaction graph. The day timeline did not stimulate many ideas. When interviewing the staff from the national capacity optimization department and planners from the expert center, only the annual overview timeline was used.

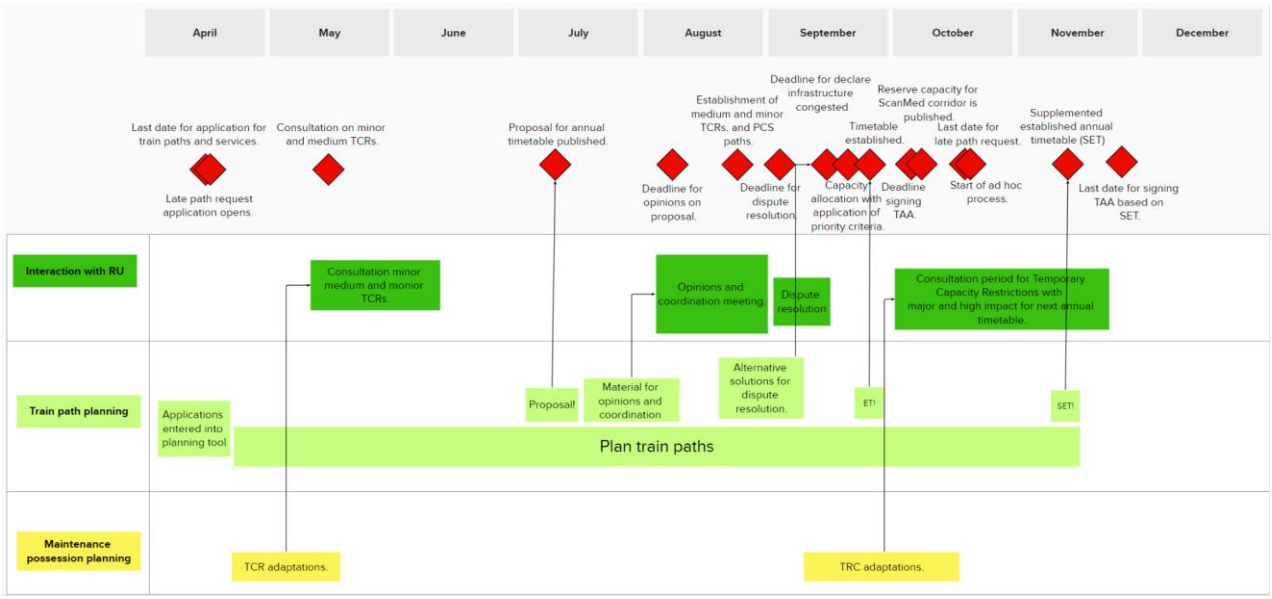
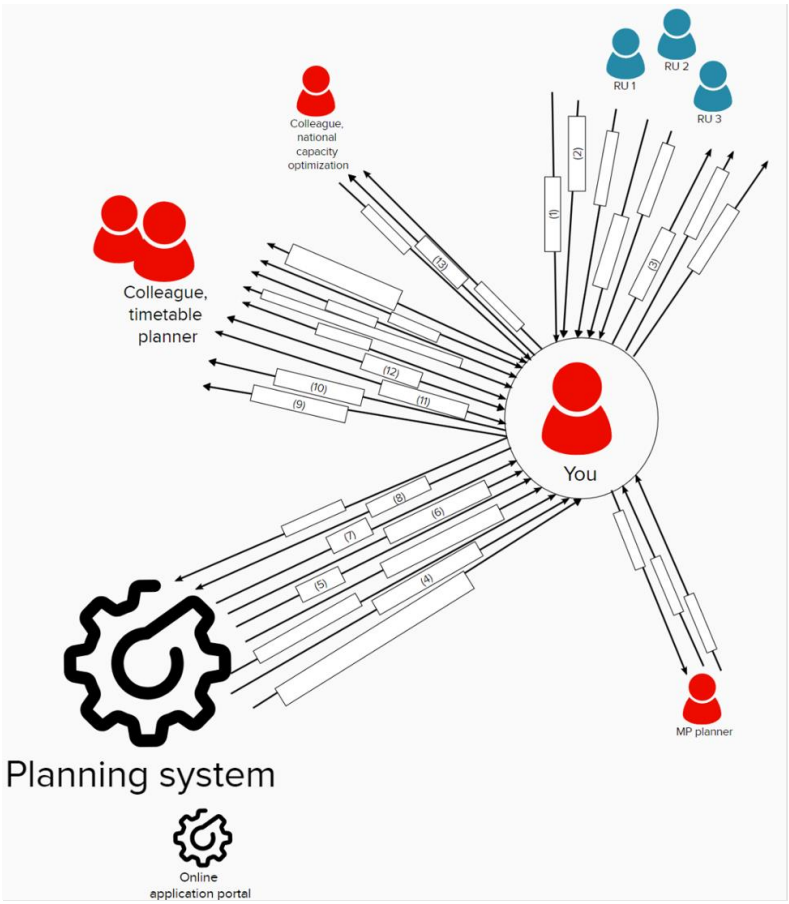


Figure 6.5.4: The annual process timeline used during the semi-structured interviews. Translated from Swedish by the authors.



1) Why did you plan like this?
2) Could you plan like this instead?
3) What is the intention with this application?
4) Information about train (including specific application details?)
5) Topology
6) All train paths and variants
7) Conflicts
8) Change train path like this.
9) Change in train path that has been handed-over?
10) What's the intention behind X?
11) Where is train path X?
12) Train path hand-over.
13) Number of finished train paths.

Figure 6.5.5: The interaction and communication graph used during the semi-structured interviews. Translated from Swedish by the authors.

To synthesize the output from the different interviews the post-it notes from each interview group were copied into the same time-line. They were then clustered based on content, and each cluster was summarized in one post-it notes. The summary post-it notes were used to identify areas where decision support with optimization may be useful.

HTA

To understand the planning process and specific needs while performing the planning task in the digital planning tool currently used, a hierarchical task analysis (HTA) was performed. The HTA was constructed based on information obtained from a meeting with a person who had access and experience with the timetable planning tool (TPS). During the meeting, the different steps needed to plan a timetable were shown in the tool and captured in the HTA flowchart. An additional meeting was scheduled to further elucidate how certain tasks were performed, and finally the flowchart was sent to the respondent who suggested some further changes to the HTA, to ensure that the task was captured correctly. The final HTA after both iterations can be found in Appendix C.

Results from Iteration 1

Interviews with timetable constructors in Iteration 1 provided an overall view of the way of working. The interviews with staff from the expert unit "Railway Capacity Center" on optimization functionalities resulted in a list of identified challenges where algorithms may be useful.

During the interviews, different "stages" of the planning were identified (see Figure 6.5.6). These

are not formal stages, but rather reflect how many planners seemed to work.

First there is some sort of clean-up or preparation stage (marked with a 1. in the figure). This is when planners look at the capacity requests to identify any "odd" applications and try to get a sense of the problem as a whole. Then follows the "base planning" stage, where the base-variants for all trains are planned (2.). There is some sort of overall prioritisation of which train paths should be planned first that is provided by the national capacity optimization department. Finally, the planners focus on "fine detail planning" – this is mainly train variant planning, but could also include quality enhancing work (3.). Last but not least, some planners also have to construct alternative solutions to be discussed with RUs (4.). This may be done throughout the planning process but is mainly required when preparing for dispute resolution meetings. Note that the borders between phases are not clear and well-defined, and a planner may very well do some stage 1 tasks later in the process than what is shown in the figure.

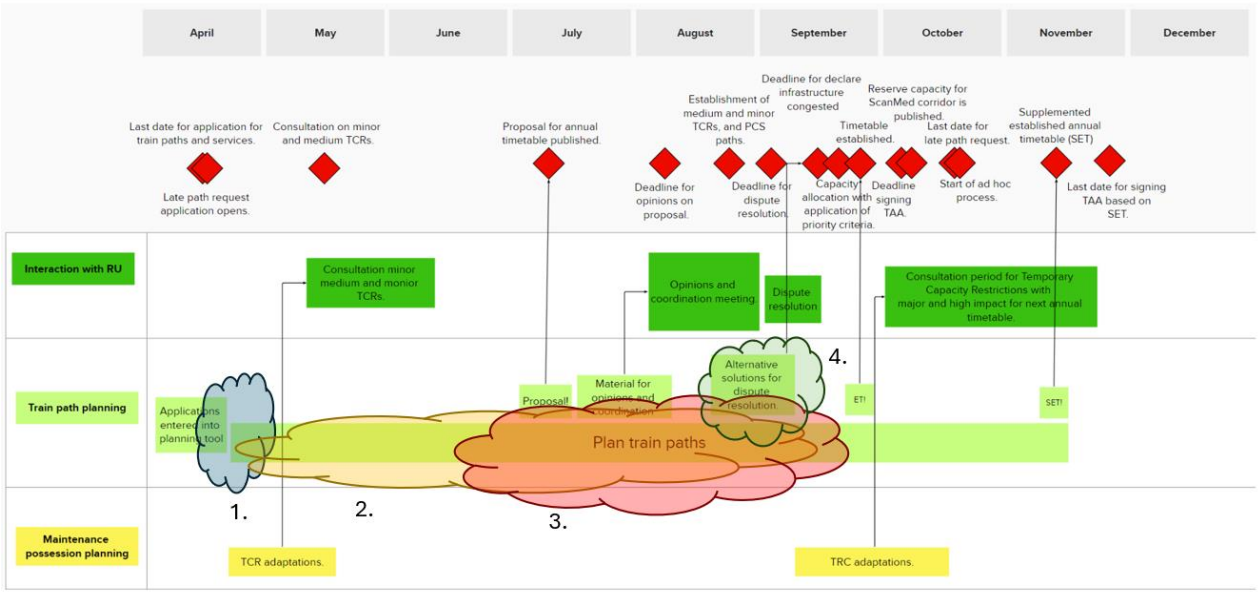


Figure 6.5.6: Different planning stages.

The focus during the interview with staff from expert unit "Railway Capacity Center" was to identify tasks where the planners could benefit from optimization support. These tasks, as well as the main goal of the optimization method that could support the task, are presented below. Note that in a final version of a decision support tool many tasks may very well be implemented in one "step", i.e. clean-up, provide a good starting point and provide a starting point with a reduced number of conflicts.

Clean up data in preparation stage

Task: Clean up input data
Support tool function 1: Identify applications that do not look as expected
Support tool function 2: Generate train paths that fulfil some basic requirements

After the application deadline, all applications are automatically read into the planning tool. At this point, it would be useful if the tool could indicate which train paths are different than expected. This would require that the tool has some way of defining the "expected" characteristics which requires more data analysis focused methods than the optimization algorithms covered in this WP.

Another clean-up activity is ensuring that the train paths defined by the applications are sensible and follow certain base rules, and if they do not – to change them so they do follow these base rules. The base rules include physical feasibility rules but could also include e.g. allocating robustness time supplements to the different parts of the trains' trips. A good route should also be generated for each train path.

Solve "easy" conflicts in preparation or base planning stage

Task: Solve "easy" conflicts
Support tool function 3: Generate a starting point timetable with a reduced number of conflicts

When all applications cannot be fulfilled, the planner is responsible for deciding which operator gets which capacity. However, if all delivery commitments and planning rules can be fulfilled, then the problem can to some extent be considered "easy". It would therefore be useful if the planning tool could solve as many conflicts as possible without breaking any delivery commitments or planning rules. That is, if the tool could provide a partial solution to the total timetable planning problem, where the remaining conflicts are conflicts that require some delivery commitments, or planning rules, to be broken.

This is a functionality that it may be hard to find a good algorithm for as the concept of "easy" can be hard to pin-point mathematically. Therefore, it's important that examples are provided for this functionality in later iterations to test if proposed algorithms behave as intended.

Working with conflicts that require due consideration

Task: Solve conflicts that require due consideration, i.e. solve "hard conflicts".
Task: Improving the quality of a timetable.
Support tool function 4: Allow the planner to influence which solutions the tool generates.
Support tool function 5: Support the planner when investigating different solutions.
Support tool function 6: Allow certain planning rules to be broken
Support tool function 7: Force the algorithm to make certain decisions
Support tool function 8: Allow train paths to remain flexible to some extent to allow for later added trains to get better train paths

As opposed to the "easy" conflicts, there will also be conflicts where delivery commitments cannot be fulfilled. These require the input of the planner, and different ways of changing the optimization problem before running the algorithm again were suggested, e.g., let the planner:

1. Change the priority of trains.
2. Move trains around (out of and into the problematic area).
3. Cancel a train.
4. Move a set of trains.
5. Change the flexibility of different trains.

A problem that was identified for function 4 is that a train that may seem to not be part of the conflict may also need to be changed in order to solve the conflict. To ensure the planner remains in control of the changes, it should be possible to somehow decide which (parts of) train paths can be moved to resolve a conflict or to at least present all the changes that were made to train paths to resolve the conflict.

Further, timetable planners do occasionally break the planning rules for a variety of reasons. This must be possible also when using the decision support tool with optimization. That is, it must be

possible to tell the optimization to disregard certain rules. Similarly, sometimes certain details of the plan must be fixed, e.g., a train meeting must take place at a specific place, or a train must be allocated to a certain track in a station. Then it must be possible for the planner to enforce this.

Finally, when train paths are manually planned it is laborious to make changes to other (already planned) train paths just to see if this opens up new solution opportunities. However, this can be done when using an optimization-based support tool. It is important that this flexibility gain can be used by the planners in a good way, and to avoid unnecessary fixing of trains. While always allowing full flexibility would result in a timetable with the best quality measures, some sort of fixing will be needed, both to decrease the solution space and to let planners work on different parts of the timetable without disrupting each other. An initial idea is to fix the train paths at delivery commitments, and potentially also at hand-over points between different planners, while allowing all other times to vary.

Long-distance trains

Task: Planning long-distance trains.
Support tool function 9: Support for adding long-distance trains early on with some level of flexibility

Long-distance trains are often hard to handle in the planning process, and they tend to get a lot of time supplements. Therefore, support for doing some sort of base planning for long-distance trains early on, while still conserving enough flexibility to avoid unproportionally negative impact on other trains, is wanted. This is closely related to the point on avoiding premature fixing of train paths. Note that this type of problem is also relevant for border-crossing trains.

Train path variants

Task: Reducing the number of train path variants.
Support tool function 10: Support for integrating different train path variants
Support tool function 11: Plan new variants close to an existing base train path

Train path variants are created when, e.g., a train is affected by a maintenance work during some of its operation days. It's preferable to keep the number of train path variants down as each variant must be planned separately. There are many functions that could support a planner when trying to reduce the number of variants. The tool could, e.g., suggest which variants should be created, integrate variants into one train path as soon as possible and automatically plan variants running close to their base-train.

Track allocation planning at stations

Task: Find good track allocations at stations
Support tool function 12: Support for good track allocation at stations

Stations are increasingly often where the hard-to-solve conflicts occur. Good support for track allocation planning at stations is therefore vital. This includes the same type of rules as for timetable planning in general, i.e., the allocation should be physically and commercially feasible, and robustness should be considered.

Maintenance possessions

Task: Plan for maintenance possessions in combination with train paths
Support tool function 13: Support for planning maintenance possessions and train paths

simultaneously

Maintenance possessions and trains paths both take up infrastructure capacity. Ideally, the planning support tool should support planning of both trains and maintenance possessions simultaneously. However, currently this is not done in the Swedish planning process. Rather, maintenance possessions that have a large impact on the traffic are established before the annual capacity allocation starts, and are viewed as unmoveable. Maintenance possessions that have smaller impact on the traffic are planned after the establishment of the annual timetable, in a process called “revision planning”.

6.5.3.3 Iteration 2

Based on the material and the knowledge gained from Iteration 1 on how timetable planners perform their daily work and the identified needs for decision support, a set of user interface proposals were developed in PowerPoint. During this iteration two different aspects were considered: (1) How can a timetable planner understand the quality of a timetable/train path and formalize what needs to be changed? And (2) what algorithmic approaches may be feasible for the challenges identified in iteration 1? Next iteration should be focussed on how the planners, partially based on the more formalized way of describing timetable quality given by this iteration, could interact with algorithms to work with some of the identified challenges.

HMI sketches and Power Point prototype

For the second iteration, HMI sketches and a Power Point prototype of the suggested functions and their connection to existing or new tasks in the HTA was produced. The updated HTA can be found in Appendix C, and below are some screenshots of the most important functionalities. Before showing the mock-up to the respondents, the interviewers explained that the visualization of the functionalities was not intended to be the final version and that the restrictions imposed by Power Point had greatly affected the design. However, the mock-up did demonstrate the intended functionalities. The respondents were told that feed-back on both design and function was most welcome.

In the earliest stage of the second iteration, simple hand-drawn conceptual images were used to illustrate possible functions. For example, type of metrics, metric granularity, the effect of a planning action on adjacent trains and how the changes would improve chosen planning parameters were functions that were conceptualized.

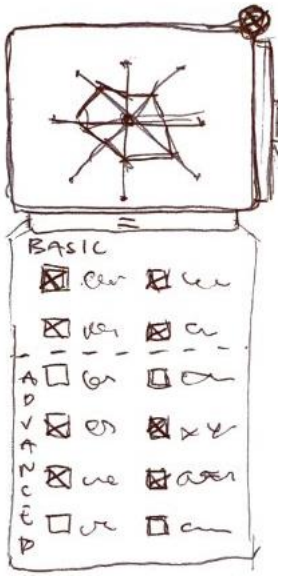


Figure 6.5.7: Example of HMI-sketch used to mediate discussions with the planners.

To support the planners' understanding and formalization of timetable and train path quality, functionalities aimed at measuring this and visually presenting the measurements to the planner were suggested. The suggested functionalities include:

- A radar chart intended to help planners decide which train to work with next, and also to compare solutions and trains when it comes to planning progress/level of completion.
- A traffic light tile board intended to give a quick overview of a train paths/timetables quality and problems.
- More detailed information for analysis of different quality aspects, and
- A capacity restriction (or maintenance possession) calendar with a visualization of the spread of train path variants.

Each component is described in more detail below. Views for comparing two trains, or two solutions were also developed. They are outlined at the end of this section.

The radar chart (Figure 6.5.8) shows six metrics in the same graph for a train: Number of unplanned trains with higher priority, technical time added to the train, conflict minutes (before, after and in the current construction area) and level of completion in terms of miles. A second version was also suggested with a slightly different set-up.

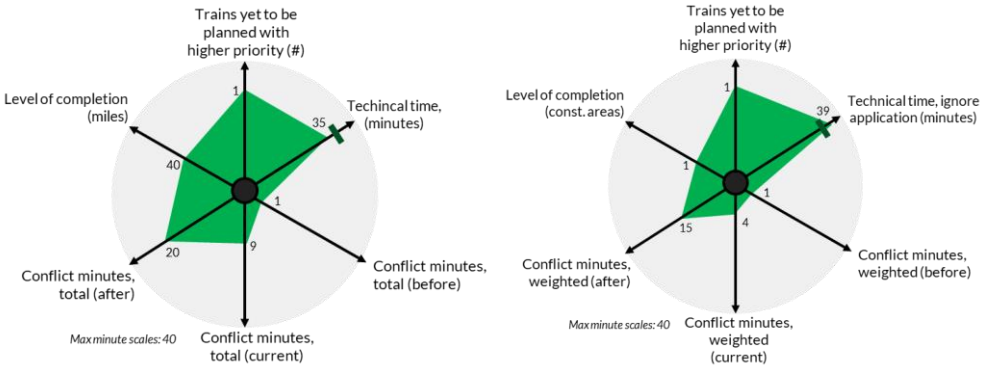


Figure 6.5.8: Two versions of the radar chart

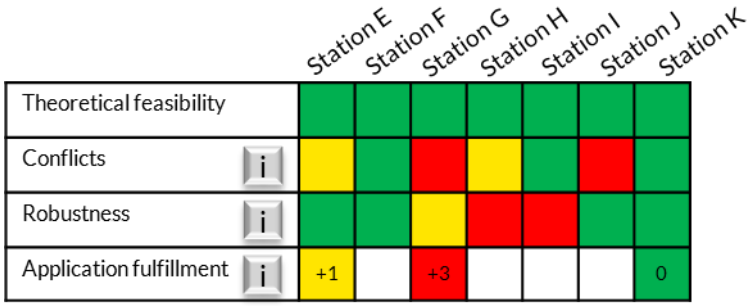
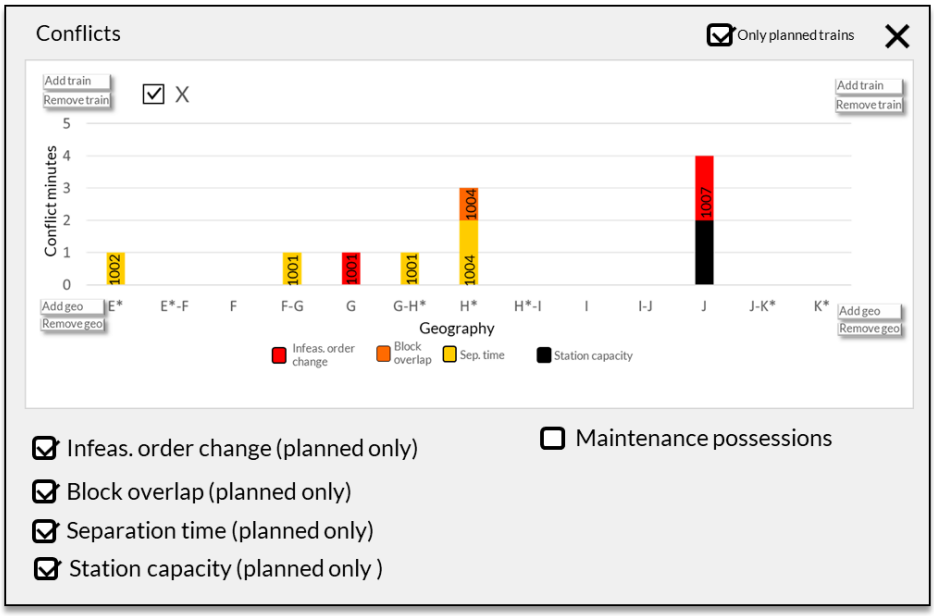


Figure 6.5.9: Traffic light tile board

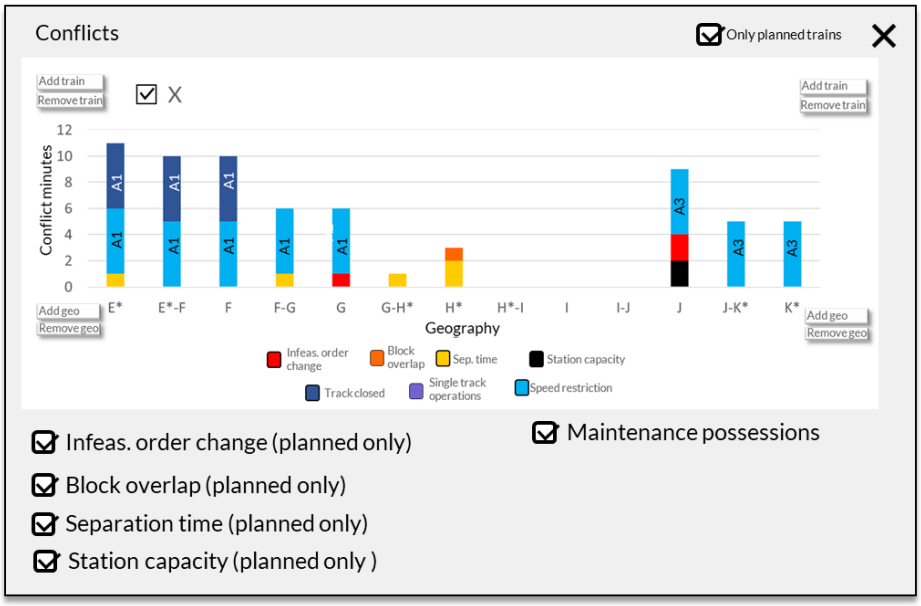
The **traffic light tile board** is a concept that is designed to make use of human pattern recognition ability to quickly give an overview if certain criteria are fulfilled or not. The concept originates from alarm tile design in nuclear powerplant control rooms (Brown et al., 2000).
The concept can accommodate hundreds of datapoints in a single at-a-glance overview, enabling the planner to see if the criteria for parameters such as theoretical feasibility, conflicts, robustness, and application fulfillment has been fulfilled or not across a geographical area. In Figure 6.5.9, three levels of criteria fulfilment are used. Fulfilled and OK (green), in need of attention (yellow), and severely violated (red). The “traffic light” categorization (when to use green/yellow/red) can be adapted as needed. More detailed information for each quality aspect can be accessed from the tile board. Currently, the additional information is always presented in a form of different charts, although the final visualization of the additional information depends on what needs to be communicated.

When it comes to the **additional information** for the different quality aspects, the conflict bar charts are shown in Figure 6.5.10. The first graph shows four different conflict types: conflicts where train order is changed at a resource where this is not possible, conflicts where two or more trains occupy the same block (but do not change order), separation time conflicts, and when station capacity is not respected. The second bar chart also include conflicts with maintenance possessions.

The train number of the conflicting train is shown in the bar chart. The user can choose to exclude trains that have not yet been planned. This is important as including conflicts with trains that have not yet been planned can muddle the picture when trying to figure out which conflicts ought to be solved now as the conflicting train has already been planned, and which ones may be solved at a later stage when the conflicting train is being planned. The included geography can be adapted to the planners need (extend or shorten the visualized stretch). If the included geography is larger than the planner’s own construction area, the construction area will be indicated with a green square.



(a) Conflicts with other trains.



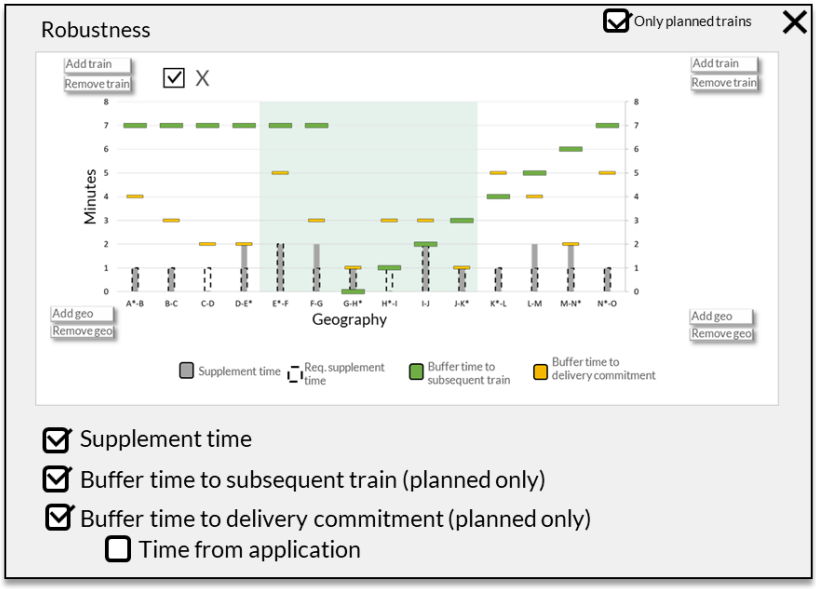
(b) Conflicts with other trains and maintenance possessions.

Figure 6.5.10: Conflict bar charts.

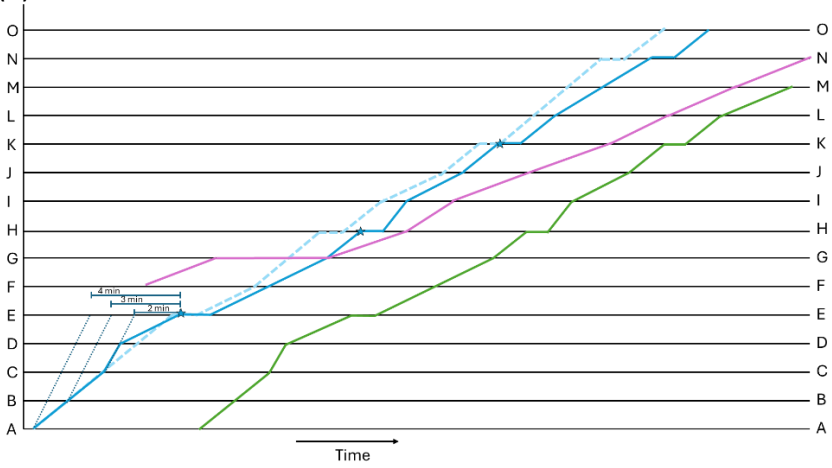
Robustness is visualized using a “bullet chart”, that shows minutes on the y-axis and geography on the x-axis, as shown in Figure 6.5.11(a). The bullet chart includes four different metrics that can be compared in each bar. Each bar shows robustness time supplement (grey), required robustness time supplement (dashed), buffer to subsequent train on the geography (green), and buffer to delivery commitment (yellow). The required supplement time is the time specified in planning rules. There is one bullet graph per stretch included in the view. The visualization enables the planner to see how the robustness of the train, in terms of supplement time and buffer time, develops over the train’s planned trip. In this figure the geography is larger than the planner’s construction area, as knowing what supplements that have been added to the train before and after may affect the decision of how much supplement time need to be added at stations inside the current planner’s construction area.

Figure 6.5.11(b) shows the graph corresponding to the bullet chart. The blue train is train “X”. The

solid line is the currently planned train path, while the lighter dashed line shows what the train path would look like with the required supplement time. The dotted darker lines show when the blue train could arrive given that it drives at its maximum speed. The purple and the green trains are the subsequent trains at different geographies.



(a)



(b)

Figure 6.5.11: (a) Robustness graph with multiple robustness metrics. (b) The corresponding timetable graph.

The chart for application fulfilment shows how far away the current train path is from various application details at the relevant geographies (see Figure 6.5.12). An application detail is e.g. wanted arrival/departure times, associations that should be of a certain length or a track requirement. A track requirement is e.g. that the train must be allocated to a track with a platform at stations where the train has a passenger exchange. The letters under the geography names specify what type of commercial activity takes place at each station.

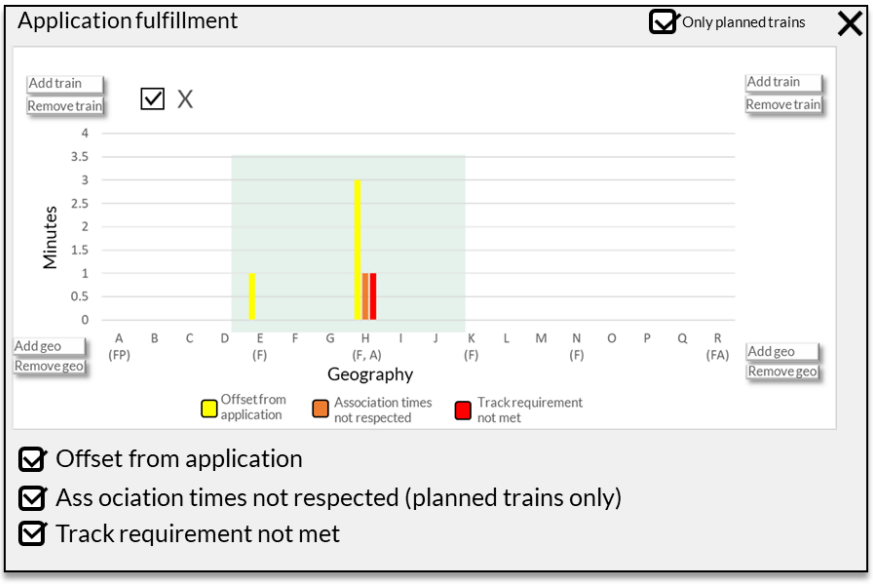
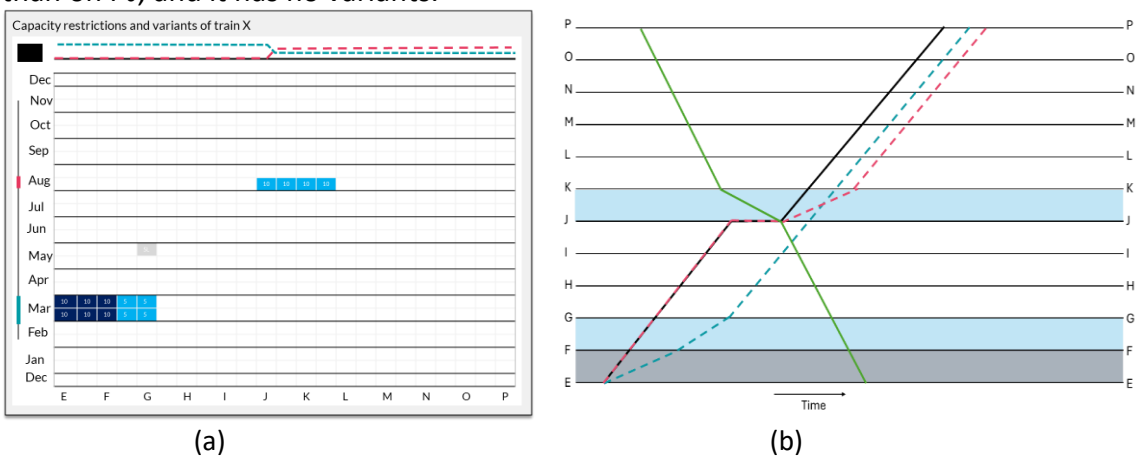


Figure 6.5.12: Graph to visualize application fulfilment

Finally, Figure 6.5.13 shows the capacity restriction calendar with a visualization of the spread of train path variants. The calendar has time of year on the y-axis, and geography on the x-axis, and maintenance possessions are shown as squares at the time/geography where they have an effect on the train. The number inside the square shows the predicted or planned additional time that is needed to regulate the conflict with the maintenance possession. To the left of the time of year axis is a black line showing the train path variant that runs most days (which we will call "main variant"), and coloured rectangles showing when different variants are planned to run instead of the main variant. At the top of the calendar the different variants and their spread in time are shown per geography. The situation captured in Figure 6.5.13(a) is shown as a graph in Figure 6.5.13(b). The green train runs from April to December, and therefore does not interact with the teal variant. The maintenance possession time is added to the green train on the link K-J rather than on I-J, and it has no variants.



(a) (b)

Figure 6.5.13: Maintenance possession calendar and spread of train path variants.

To **compare** two trains, the same visualization objects are used, but with two trains instead. Figure 6.5.14 gives an overview of the quality scoreboard when comparing two solutions. Colours are used to differentiate the two solutions from each other. Note that more than two solutions could be included in the comparison. The number of trains planned per priority category is plotted instead of the number of unplanned trains that have a higher priority than the current train. Also, the number of variants is included in the radar chart.

The visualization for comparing two solutions is basically the view for one train, but with two trains represented with different colours akin to how different solutions are presented.

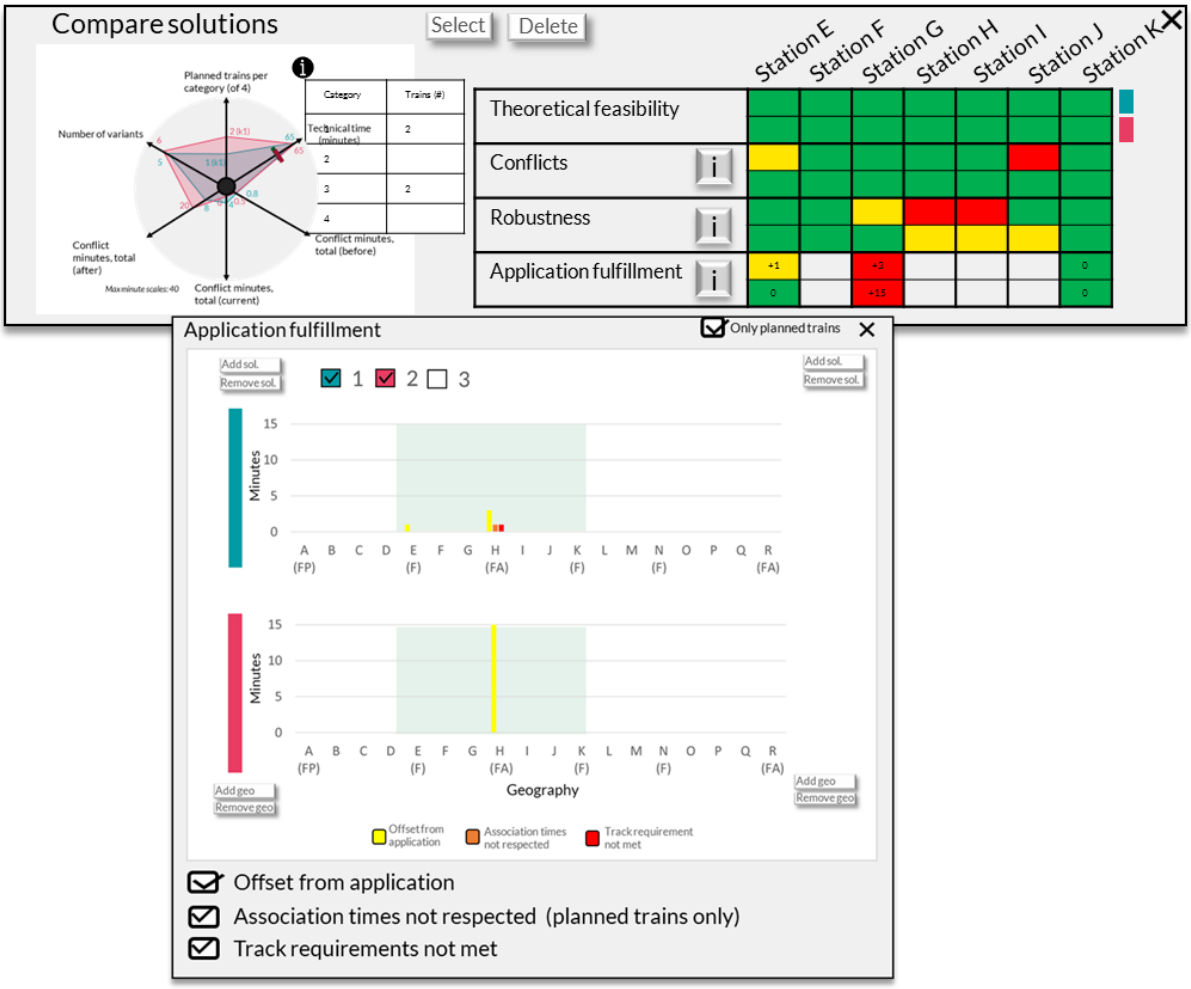


Figure 6.5.14: Comparing two solutions.

Results from iteration 2

In this section, summaries of the feedback received from the participants on the HMI-prototypes during workshops and interviews are described.

The radar chart was appreciated for its ability to visualize several relevant parameters in the same graph. However, in this iteration the scales were not normalized which made the scales difficult to interpret. For example, what is seen as a "good" value can vary between metrics which means that scales need to be adapted for each metric individually to make the radar chart easy to read. The same feedback was received for the function where two trains can be compared by plotting two trains in the radar chart.

The **traffic light tile board** was valued for the possibility to get a quick overview of several important metrics for the planner's area of responsibility. Especially, the ability to get an overview of how applications are fulfilled, and the ability to see the effect of changes in timetables on quality metrics was appreciated. One of the planners expressed that the use of a metric tile board could become an "aha-moment" for planners, since it is challenging in today's system to get an overview and compare before/after a change how it affects a quality metric, such as e.g. application fulfilment. A question raised was how to represent the geography in the tile board in the best way

since all stretches may not need to be visible at once. Also, a discussion point is how to set the threshold criteria for what should be a green/yellow/red tile. Another positive comment about the tile board is the opportunity to create a "pedagogical tool" for what is a good enough timetable. If the tile board is all green, the timetable can be seen as a good timetable. The challenge is of course to implement the criteria for each colour sufficiently well. The more detailed graphs available when clicking the quality metrics in the tile board was also discussed in-depth.

The **conflict bar graph** with train and station conflicts was perceived as easy to understand. However, as conflicts can readily be spotted in the timetable graph as well the added value of the conflict bar graph is mainly to help planners identify conflict type. When interviewing capacity experts, the questions of how conflict minutes should be calculated, and which broken rules should be considered as conflicts, were also brought up. In general, the **robustness bullet graph** with multiple robustness metrics was perceived as complex by many participants. It contains condensed information in each separate bar, which is multiplied across the geographical x-axis. Nevertheless, the robustness metric is deemed important by the planners, but it is clear that it is not very intuitive, and training will most likely be needed before this type of more complex visualization would be adopted. The **application fulfilment bar graph** was valued for a detailed view that can be used to communicate with applicants since the bar graph is easier to interpret compared to using a train graph. The possibility to see how association times are respected (or not) was also seen as an addition to what is possible to see in today's system. How well association times are respected was commented as "a very important metric to customers", but without support the planner can't see what effect a change in the timetable has on this metric. The visualization of **maintenance possession calendar and spread of train path variants** was valued for the possibility to see variants over geography and time. A function that was expressed as important by several participants is the need to reduce variants by merging them. This is difficult today since there is little support to identify which variants are close to each other and could possibly be merged. The functionality presented here does not support the actual merging but could be used as a decision support in seeing how far apart two trains are and facilitate the choice of variants to be merged.

The ability to **compare two solutions** by visualizing two sets of quality metrics alongside each other challenges the current work practices since there is little time to do such comparisons today. According to one of the planners, comparing solutions allows the planner to see what has been achieved and see if the solution created is a better solution. It also supports the planner of not having to remember what a solution looked like, when creating a new one. Such functionality of creating alternatives is available in the current system (sandbox function) but is perceived as too time consuming to use. The proposed visualization would facilitate direct comparison between solutions and could presumably save time.

During the interviews different findings of more general character were also made that relate to the planners' daily work. Several participants mentioned that easily interpretable visualizations that complement the train graph would be a very good tool in communication with applicants since a train graph does not show timetable quality per se and it is not possible to show an applicant the trade-offs that had to be made to arrive at a good train plan. By having the type of decision support presented here, the planners would have a tool to show how and why certain choices have been made. Further, the participants with management roles also pointed out the value of easy-to-use information to make tacit knowledge available to planners with less experience. Visualizing the basis for decisions also makes it possible to manage what criteria should be used for certain decisions, making the planning process more uniform across a team of

planners. Another benefit of visualizing quality metrics that emerged from the interviews is improved transparency of what the planner has done, what type of solutions have been chosen and what trade-offs have been made to achieve a solution that is feasible for the customers. At the same time participants stress the need to proceed with care in how to use advanced digital tools, since the tools also create changes that may need education and support from the organizational perspective. Also, planners warned about the risk of overestimating the importance of the quality metric signals without understanding the underlying problem.

In general, we can conclude that the seemingly easy to use and understand traffic light tile board could be a good start for implementation. If it is adopted by planners in their daily work, it could create an avenue for introducing more advanced visual decision support using quality metrics later on.

Algorithm discussion

In this section initial ideas for algorithms for the different functions are discussed. Note that a full description of all algorithms is not in scope of this section, the aim is rather to provide some sort of insight into whether optimization can be used as part of the support function, and to give an outline of a potential optimization solution. When applicable we will refer to other sections in this report where algorithms that could be used as support are presented.

Before going into each function individually, some notes of a more general nature are appropriate.

First, timetable optimization problems are generally NP-hard. However, in practice, good-enough solutions can be found fast enough to still make optimization a gainful decision support tool. The size of the problem at hand matters and generating a timetable for an entire country will take much longer time than generating a timetable for a specific line. In the discussions below we assume that some sort of heuristic or decomposition method can be used to find a good enough solution fast enough given that the function can be adequately modelled as an LP or MILP problem. Another way to limit the problem of long solution times is to limit the problem scope. For example, when moving a train in the southern part of a country, maybe only trains that run in this area need to be included in the problem? This may of course influence the overall quality of the timetable (unless we can prove that the trains running in the north do not affect trains in the south), but the extra work and time needed to include both the northern and southern trains may not be worth the potential quality increase. Further, if different planners are responsible for different parts of the timetable, changing trains outside of your area of responsibility may require some extra communication work and collaboration with other planners. Deciding on how to limit the problem when performing certain tasks is therefore an important implementation aspect for many functions in the list – both from a tractability and from a usability point of view.

Secondly, while some of the functions are more or less straight forward from an algorithmic point of view, more research is needed for all of them from a usability perspective. Many of them also require more research from an implementation perspective, i.e. when it comes to choosing the most appropriate models, parameters and/or algorithms. Further, the timetable problem is inherently multi-objective, so handling many objectives simultaneously will be necessary.

Support tool function 1: Identify applications that do not look as expected

Given that the applications can be viewed as points in a multi-dimensional space, density-based methods may be useful for the task. As the density of the applications is unlikely to follow an identifiable probability distribution, parametric density-based methods are likely not applicable.

Rather, more complex non-parametric methods, or a nearest neighbour method, may be more suitable. Regardless of which method is chosen, research is needed to define a good distance metric for the applications, as this will greatly affect the effectiveness of the anomaly detection. Other machine learning methods such as isolation forest or some neural network method may also be tested.

Support tool function 2: Generate train paths that fulfil some basic requirements

This is a fairly straightforward optimization problem as conflicts are not considered and the problem, therefore, can be divided into many sub-problems where each sub-problem only includes one train (or a group of trains connected via associations), and all sub-problems can be solved in parallel. Depending on the complexity of the basic requirements a greedy algorithm may even be appropriate. For more complex basic requirements, i.e., requirements that allow for the robustness supplements to be moved around to some extent, the overall problem will most likely become a set of linear programming problems, or potentially a set of mixed integer linear programming problems. The latter would be true if, e.g., ignoring associations should be allowed but at a high cost.

Support tool function 3: Generate a starting point timetable with a reduced number of conflicts

This is a support function that requires more research not only from a usability/understandability perspective but also from an implementation and tractability/solution time perspective. Potential methods are decomposition methods like the ones proposed in Chapter 7, or some MILP-based heuristics where batches of trains are added to the timetable based on their priority (see Gestrelius et al. (2015) for an example). Note that this is a function that is intended to be run once in the beginning of the process and therefore long execution times can be accepted (even counted in days), but the returned timetable must be usable.

Support tool function 4: Allow the planner to influence which solutions the tool generates

If an existing MILP-framework with suitable variables exists, all the functions suggested (1. Change the priority of trains, 2. Move trains around, 3. Cancel a train, 4. Move a set of trains, 5. Change the flexibility of different trains) can easily be implemented by changing, adding or removing constraints. The M2-framework is an example of such as framework, and the work presented in Kloster et al. (2023) also allows for these types of interactions with the underlying optimization model.

Support tool function 5: Support the planner when investigating different solutions.

This function requires the planner to be able to save different solutions, and to analyse how they differ. The analysis can be supported by different quality metrics (see Figure 6.5.13 for an example) or by visualizing which changes have been made in the graph view. To gain more knowledge on how planners can be supported during this analysis, more usability research is required.

Support tool function 6: Allow certain planning rules to be broken

If all planning rules are implemented in the MILP-framework as constraints, allowing certain planning rules to be broken simply translates to removing certain constraints or making them soft. Once again, the mathematical side of this function seems quite straight forward, but the usability aspects require more research.

Support tool function 7: Force the algorithm to make certain decisions

If the decision that should be enforced can be included as a constraint (or a set of constraints), the

mathematical side of this function is quite straightforward. It is likely that most of the decisions that a planner may wish to enforce can be handled by adding/changing constraints. In the M2-framework decisions that can be enforced are e.g. where trains should stop/meet/overtake, train orders per geography and exact arrival and departure times at geographies.

Support tool function 8: Allow train paths to remain flexible to some extent to allow for later added trains to get better train paths

Allowing train paths to be moved around is at the core of timetable optimization models. This function is therefore well-supported by optimization in general. However, to avoid overloading the planner with changes to be analysed, some sort of fixing of (parts of) train paths will most likely be needed. The question of how to work with flexibility vs. fixing is therefore an important usability aspect. See e.g. Gestrelius et al. (2023) for an example of working with different degrees of flexibility.

Support tool function 9: Support for adding long-distance trains early on with some level of flexibility

Long-distance trains are particularly problematic – planning for them and fixing them early on can be too costly for other trains, while planning for them late may result in the long-distance trains getting unacceptably long trip times. This is a problem that can be, at least partially, solved by ensuring that trains retain some level of flexibility after they have been planned and "fixed". Therefore, the problem with long-distance trains could be alleviated by using optimization support and retaining some level of flexibility as discussed above. However, adding some extra buffer time to whichever trains that are planned early on (especially if it's the long-distance trains) may also be necessary to find a good overall solution. Another interesting question to research is to what extent the finally planned arrival time of long-distance trains can be predicted at border crossings, be it between countries or between different geographical timetable planning areas, as this could be used to help planners save space for long-distance trains at appropriate places in the timetable.

Support tool function 10: Support for integrating different train path variants

Each additional variant is an extra train path that needs to be planned (albeit some of the variants may be very short train paths). Reducing the number of variants therefore reduces the timetable planning problem size. If a support tool can help planners to partially or completely integrate train path variants, the problem size can be reduced. There are many ways that this functionality could be implemented, e.g. as overnight runs or as planner-initiated tests to see if certain variants can be integrated. From an optimization perspective some sort of distance metric between the variants to be integrated is needed. This metric will then be added to the objective, either as a single objective where other quality deteriorations are limited as part of the constraints, or as part of a multi-objective cost function or framework. See e.g. Nachtigall and Voget (1996) and Schlechte and Borndoerfer (2010) for different ways to handle multi-objective timetable problems.

Support tool function 11: Plan new variants close to an existing base train path

The optimization modelling needed to for this is similar to the one discussed above – some sort of distance metric needs to be included and weighted against the other objectives.

Support tool function 12: Support for good track allocation at stations

Stations can be handled in some sort of master/slave framework when included in a timetabling problem, see e.g. Lamorgese et al. (2017). Track allocation at stations can also be optimized without changing the arrival/departure times, as presented in Section 6.2. In practice, track

allocation in stations is sometimes handled by separate planners. Therefore, keeping the track allocation support somehow separate, but when needed, interacting with the rest of the timetable planning problem makes sense. For example, it would probably be beneficial with a separate view for allocating tracks at stations.

Support tool function 13: Support for planning maintenance possessions and train paths simultaneously

Simultaneous planning of maintenance possessions and train paths is a research area all in itself, see e.g. Liden (2015) for an overview. The process for planning maintenance possessions is sometimes also separate from the planning of train paths, where possessions are either fixed before the train paths are planned or train paths are planned first and then adapted to make space for possessions during parts of the year. A first step for including maintenance possessions is therefore to ensure that the already established possessions and their restrictions are respected by the optimization model. A second step would be to include support for replanning train paths based on new maintenance possessions. A final step would be to actually consider the traffic and maintenance possessions at the same time. Preferably, the strategic planning of maintenance should also be properly supported and this initial plan for maintenance possessions could be used to set limits on where the possessions should take place, thereby reducing the problem size.

6.5.4 Conclusions

Based on interviews and hierarchical task analysis, 7 challenges have been identified where some sort of data analysis or optimization support would be useful. These challenges were broken down into 13 support tool functions. Algorithmic approaches that can be used to implement these support tool functions have been identified. HMI-sketches and prototypes on how to communicate the quality of a train path or a train timetable have also been suggested. The intention is that the quality HMI-sketches should be extended to also include optimization interaction functions in the next participatory design iterations.

The study of current work practices shows that introducing more automatic planning support will change the everyday planning work. Notably, currently the planners make very detailed changes to train paths and spend a lot of time entering these changes into the system. There is little time for working with the overall quality of the timetable, although the planners do consider multiple quality aspects in their everyday work. An easy way to gauge the quality of a train paths is currently lacking, and the traffic light tile board was well received by the participants. However, more complex quality information, such as e.g. the robustness bullet chart with multiple metrics, was perceived as hard to understand and use. The prototype visualizations were found to have both pros and cons, and improvements can be made by further iterating the proposed ideas.

Other benefits of introducing visualization of quality metrics were also discovered during the interviews. Firstly, the idea of using visualizations as a mediator for discussion with applicants was an unexpected finding, where planners can more easily show and reach an understanding for trade-offs made between customers in the planning process. Secondly, the visualizations could potentially make tacit knowledge of how to create good solutions more transparent and directly available to planners with less experience, improving the learning curve from novice to expert. Thirdly, making the criteria of a “good enough” train plan more transparent could contribute to planners working in a more uniform manner, facilitating setting standards for how to achieve good planning solutions. Such findings further support that improved digital support can also change work practices.

If more automatic, optimizing, planning support functions are introduced, the everyday work will move away from entering detailed planning data to steering algorithms in a wanted direction. This can be done via direct interaction with the timetable graph (e.g. pulling the train path to a new wanted position), but could also take the form of interacting via more formalized instructions (e.g. increase these robustness metrics). Either way, understanding the strengths and weaknesses of a proposed solution, and formalize a hypothesis of how these weaknesses may be avoided, will become important.

Introducing optimization-based support tool functions should reduce the amount of time planners spend entering various plan change details, and thereby free up time for working more with the quality of a given timetable and/or reduce the time it takes to construct an annual timetable. However, it is important that the current planning tool (without optimization functionalities) meets a certain level of usefulness. For example, the usefulness of being able to compare different solutions greatly increases if each solution can be saved so that the planner in the end can chose the solution (s)he prefers. Also, the introduction of the new functionalities has to be carefully designed and managed in order to reap the intended benefits.

7. Algorithms for short-term timetabling

Task 6.3 deals with optimization algorithms for short-term timetabling. Short-term timetabling refers to adjusting the timetable during the year. There are two major reasons to do this:

1. A Temporary Capacity Restriction (TCR) forces the timetable to be adjusted, because the regular timetable cannot be operated anymore. Sections 7.1-7.3 deal with different algorithms that can be used om this situation.
2. Additional and/or changed demand requires an adjusted timetable. This situation is discussed in Sections 7.4.

Sections 7.5 and 7.6 deal with general methods that can be applied to adjusting the timetable.

Each section starts with describing the background, then give a detailed problem description defining the use case in detail. In addition, we give a detailed description of the methodology to tackle the use case. Finally, we validate the developed algorithms on representative problem instances (TRL 4 validation). The results are reported at the end of each section.

7.1 An efficient MILP-based algorithm for handling TCRs

7.1.1 Background

Our work in Task 6.3 is a collaboration between SINTEF Digital (Norway) and Hitachi Rail (Italy). The aim is the development of an algorithm for conflict detection and resolution in a short-term timetabling application. The algorithm will be able to operate both at network level and station level. It will be able to deal with different Temporary Capacity Restrictions (TCRs) in a specific region of the Italian railway network, the complex Genoa node (see Section 7.1.2 for more details). In this collaboration, SINTEF is tasked with the scientific and practical development of the algorithm, while Hitachi is tasked with providing both the data and an interface to their operational systems as well as validating the solutions produced by SINTEF's algorithm (i.e., whether a certain new timetable satisfies all operational constraints). Figure 7.1.1 highlights the vision of an automatic decision support tool that train planners can use interactively to test new timetables in response to TCRs. The work done in WP6 (described here) and the work to be done in WP7 represent the first steps towards this goal.

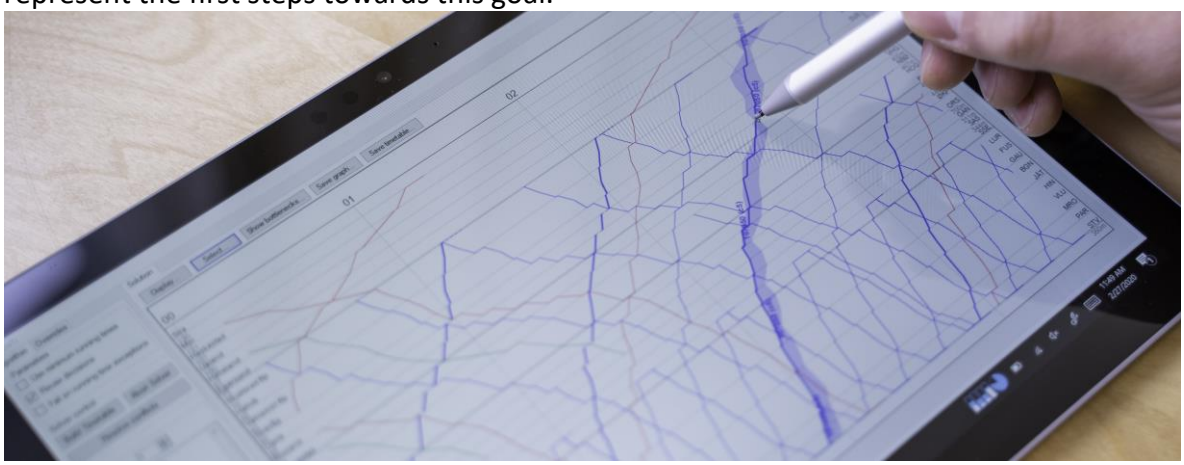


Figure 7.1.1: The vision of a decision support tool for short-term timetabling, based on a previous prototype made by SINTEF. We consider essential the interactive input of train planners.

7.1.2 The Genoa node

7.1.2.1 Area of the demonstrator

The SINTEF-Hitachi demonstrator will be configured in the Genoa SCCM area, where Hitachi's systems are currently in operation supporting railway lines with a variety of features including single and double track lines, the presence of alternative routes, and stations of significant complexity.

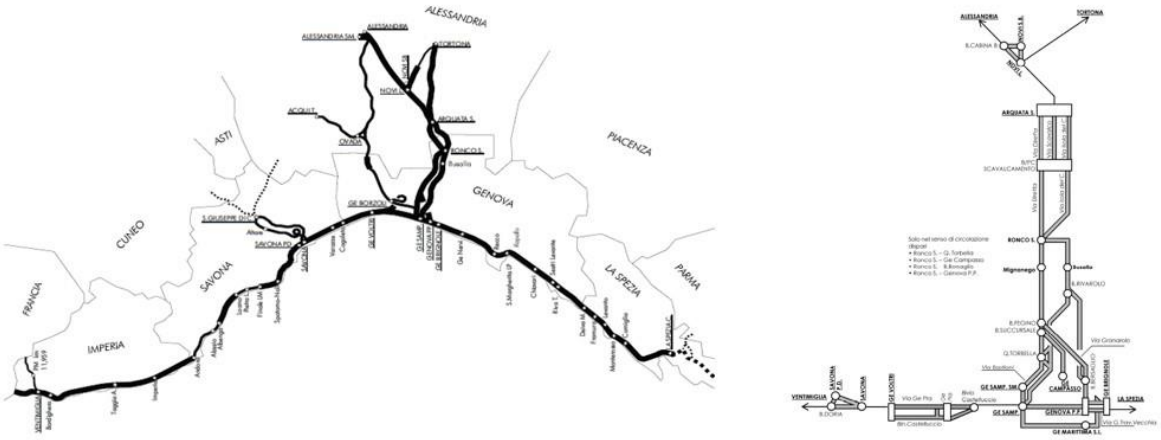


Figure 7.1.2: Double/single tracks and alternative routes

7.1.2.2 Demonstrator data

The data that will be used by the demonstrator will be static configuration data (e.g., network configuration data and train running times) and a reference timetable, both based on real data. Other data needed for the demonstrations (for example, the TCRs) will be created ad-hoc and will not be traceable to real events.

7.1.2.3 Use cases

The scenarios considered in the demonstrator will be use cases of possible perturbation, such as track or station TCRs, but also the addition or modification of new tracks on request. In the use cases, the significant area and timetable parts for the single demonstrations of the functionality will be taken into consideration. Figure 7.1.3 contains the proposed use cases. Each use case will be provided with predefined scenarios that can be modified through the demonstrator HMI. In principle, the algorithm described in Section 7.1.3 should be able to support all use cases, but some of them may require more software development work than anticipated and could be modified or ignored.

Test Group	Test description
Track possessions	Possession on one track of a double track
	Possession on both track of a double track
	Possession on a single track
	Possession on both track of a double track with available alternative routes
	Long possession on both track of a double track. Trains must be suppressed
	Long possession on a single track. Trains must be suppressed
Station possessions	Possession on a through platform
	Possession in a platform used for stopping or overtaking
	Possession in all platforms of a small station
Speed restrictions	Speed restriction on both track of a double track
	Speed restriction on a single track
Mixed TCRs	Possession on one track of a double track with speed restriction on the other

Figure 7.1.3: Use cases proposed

7.1.3 Description of the algorithm

7.1.3.1 An incremental timetabling algorithm

The algorithm developed in Task 6.3 builds on SINTEF's strong history of algorithms for train scheduling. In particular, our work on the concept of *incremental timetabling* described in Kloster et al. 2023 (started on a previous project and concluded in MOTIONAL) represents the core algorithmic framework over which we built the necessary customization to support the additional complexities found in the Genoa node.

When looking at the workflow of train planners, it is clear that most of their time is divided in two main tasks: (1) making significant changes to a timetable in order to accommodate planned maintenance work, new train services, etc. and (2) making small adjustments to the arrival and departure times of trains to make sure a timetable is completely feasible. It turns out that the second task is usually much more time consuming than the first one, while having the least impact on the overall quality of a timetable. The process that we call *incremental timetabling* intends to relieve train planners from this second task, so that they can focus their effort in making the most important decisions of the first task. The idea is simple: during the first task, train planners build a so-called reference timetable. This timetable is typically infeasible from the operational standpoint, but it takes all major constraints and preferences into consideration. This means that only minor changes are expected to be required in order to gain operational feasibility. This is where our conflict detection and resolution algorithm comes into place.

The timetabling algorithm is described in detail in Kloster et al. 2023. Here we summarize the main features. The railway network is divided in resources, either track sections or stations. We use a decomposition technique first described in Lamorgese and Mannino 2015, so that the feasibility within each resource is treated independently of the feasibility across resources. In particular, we assume there is an oracle function that, given the arrival and departure times of all trains in a certain resource, can determine whether there is any conflict within the resource. In other words, the oracle represents a conflict detection function for a single resource.

The timetabling algorithm is built on a big-M MILP model for train scheduling (see, for example, Castillo et al. 2009), whereby we assign a scheduling variable to each resource to represent the arrival time on that resource. Then, we can use linear constraints to model several operational

constraints:

- *Minimum running time constraints*: these can be simply modelled by stating that the departure time of a train from a resource is greater than the arrival time on that resource plus the minimum running time.

- *Precedence constraints*: these are constraints that involve more than one train, and they are used to model the occupation of a track/station. For example, if two trains want to traverse the same track section, then either train must wait for the other train to exit the track section before entering. Similarly, in a station, one can use linear constraints to model the number of trains and which platform they are using, creating precedence relationships between pairs of trains.

- *Safety margin constraints:* these are used to model all safety margins that need to be enforced when two trains perform certain operations too close to each other. For example, in certain small stations with single tracks on both directions, a train may enter the station no later than a few minutes before another train reaches the same station from the opposite direction.

- *Periodicity constraints*: these constraints deal with the very typical "periodic" departure times of frequent passenger train services. An example is when all trains going to the same destination need to depart from the first station (and many of the stations in between) at regular intervals of the hour (for example, every 20 minutes at hh:10, hh:30, hh:50). These constraints are not currently considered in this project, but they can be included later if necessary, as described in Sartor et al. 2023. This shows the flexibility of the model in dealing with many different types of constraints arising from real-life applications.

Given the complexity of the real-life problems, it is typically impractical (if not basically impossible) to consider all possible constraints from the start. Instead, our timetabling algorithm uses a well-known technique called row-and-column generation to make sure only the necessary constraints are taken into consideration. Only the minimum running time constraints are always added to the MILP model, which is then solved according to the following iterative process:

1. Solve the current MILP model
2. If infeasible, the whole problem as a whole is infeasible. Return.
3. Check whether the solution violates any of the remaining constraints.
4. If not, the solution is feasible and optimal. Return.
5. Otherwise, add to the MILP model the violated constraints.
6. Repeat.

Because the initial MILP model is quite small, and because the number of violated constraints at each iteration is also small, each iteration can be solved quite quickly using an off-the-shelf MILP solver (we use Gurobi). This technique has been proven to be quite effective in this kind of problems since the relationships between pairs of trains are typically quite sparse (for example, a safety margin constraint in a station involving a pair of trains departing several hours apart from the same station is very unlikely to be violated and play a role in determining the optimal solution).

The objective of this model is peculiar. Most of the contributions to the literature of timetabling algorithms focused on finding an "optimal" timetable, but it is never easy to mathematically define the optimality of a timetable, and there is too much knowledge and experience in human train planners that cannot be easily modelled as linear equations. We consider instead a different approach. We let train planners define a so-called reference timetable. This is basically a rough timetable that satisfies all major constraints (planned maintenance work, preferences of train operators, etc.), but with some small timing inconsistencies that need to be resolved. The reference timetable can also be the current official timetable. This is of course already feasible, but it may become infeasible due to a TCRs. The objective of the algorithm is then to find a new timetable that is feasible (satisfies all practical operational constraints) and that is as close as possible to the reference timetable. This gives more control to train planners, who can decide what they consider to be a reference timetable, letting our algorithm sorting out all other "small details". Here, the term "as close as possible" means that we minimize the sum of the total deviation of the new feasible timetable from the reference timetable at a selected set of stations (typically, only the major stations). This deviation can also be weighted with train priorities, so that deviations of freight trains become less important than deviations of passenger trains, for example.

We consider this to be the "core" timetabling algorithm that we built over several years of research and polished it further during the first months of WP6. All the remaining mathematical details can be found in Kloster et al. 2023. The rest of this section is dedicated to explaining how this algorithm has been further extended to support all practical constraints and features required by the use cases provided by Hitachi Rail.

7.1.3.2 Exact station platforming

The timetabling algorithm requires an *oracle* to decide whether specific collections of trains can be present in a station at the same time. From the perspective of the overall algorithm, all the oracle does is give a yes/no answer, but to compute this answer, the oracle needs to model the details of track allocation and solve the associated assignment problem. We refer to the oracle used for the Genoa node as the *track allocation solver*.

When invoked by the timetabling algorithm, the track allocation solver is given a station and the collection of trains that visit the station during the same period of time. The trains do not necessarily occupy the station all at the same time, but they may all influence each other, since the station is never empty between the first arrival and the last departure. The track allocation solver then needs to gather the following information:

- Track compatibility. The subset of station tracks that can be used by each train is found. For the Genoa node, there are several factors that determine this subset:
  - The type of train: Freight or passenger
  - The previous and next line track in the train's route
  - The type of visit: Route start, route end, actual stop or just passing.
- Conflicting pairs. Two trains cannot use the same track if they are in the station at the same time, or within the minimum safety margin for departure/arrival on the same track.

The problem can now be formulated as a graph colouring problem: To each node (train), assign a compatible colour (track), such that no edge (conflicting pair) connects two nodes with the same colour.

The track allocation solver first attempts to solve the colouring problem heuristically. The algorithm assigns a track to one train at a time. At each step, it selects the train that has the fewest track options left. If the train's track from the reference timetable is available, that track is

assigned; otherwise, a track that leaves the most options for the remaining trains is assigned.

The heuristic algorithm is fast and succeeds in most practical cases where a legal assignment exists. However, a second algorithm is used in cases where a heuristic assignment is not found. This algorithm constructs an integer programming model of the colouring problem, which is solved using the same solver as for the main MILP.

7.1.3.3 Modelling trains only as needed

Most of the time used by the timetabling algorithm is spent solving the main MILP. This MILP is solved in each iteration, and each time it gets more complex, as constraints are incrementally added. Thus, reducing the size of the MILP model is an important avenue for speeding up the algorithm.

In MOTIONAL, we have improved the algorithm by only including trains in the MILP as needed. Recall that we are doing incremental timetabling, where a reference timetable is available, and deviations from this timetable are discouraged. This means that many, if not most, trains will have the same schedule in the final solution as in the reference timetable. This fact allows us to omit trains from the MILP and improve the solving speed while still arriving at the same optimal solution.

The concrete rule is that trains are included in the MILP only after being involved in one of the conflicts detected by the timetabling algorithm. When a conflict is found, constraints are added to the MILP to avoid repeating that conflict in the next MILP solution, and the trains involved must be modelled so that the variables to be constrained actually are present. If a train does not take part in any conflict found during solving, it is not added to the MILP at all.

In the first iteration, no conflicts have yet been found and no trains modelled, so the MILP starts empty. The schedule in the first iteration is equal to the reference timetable. Any conflicts found in this schedule cause the first trains and constraints to be added to the MILP. In later iterations, the schedule is taken from the MILP solution for the trains that are modelled, while the remaining trains keep the reference timetable.

The conflicts found in the first iteration can be put in three categories:

- Inconsistencies for a single train in the reference timetable, e.g. a train going faster than the minimum running time.
- Conflicts between a train and use case-specific elements, e.g. using a track at a time that it is closed.
- Inconsistencies between train in the reference timetable, e.g. insufficient separation between two follower trains.

In a further improvement of the algorithm, we exploit the fact that conflicts of the first two types do not involve constraints between trains. Before starting the algorithm proper, we compute a *default solution*. This solution is found by solving a MILP that models all trains with no inter-train constraints (i.e. the first-iteration MILP if we were not modelling trains only as needed). We then reset to an empty MILP and start the iterations proper as described above. However, when looking for conflicts, for any train that is not modelled, we use the schedule from the default solution instead of the reference timetable. This means that only conflicts involving two or more trains will cause trains to be modelled in the MILP. Any conflict for just one train is solved in the default solution and does not reappear in the main part of the algorithm.

7.1.3.4 Alternative routes

At the start of the MOTIONAL project, the timetabling algorithm was mainly focused on solving

single line problems. Some networks with multiple lines were in use, but with limited complexity. Consequently, extensions were needed to handle the more complex network around Genoa. The dependence of track compatibility on the previous/next line track mentioned above is one example of this. A larger challenge that had not been modelled before, is the presence of alternative routes. The initial algorithm assumed that while the problem may have trains that run on different lines, each train has a single fixed route to follow. This is not the case in the Genoa problem: trains may choose between alternative routes from one station to the next in the timetable. Often, there is a choice between a newer high-speed track and an older, more winding track. Consequently, the model and algorithm has been extended to handle this.

The core of modelling alternative routes is including additional decision variables in the MILP. Naively, one could add a separate binary variable for each possible visit of a train to a station or track, that expresses whether the train actually uses that station/track. However, many of these variables must take the same value to express a consistent route, and furthermore, some parts of the route have no alternative and must be run. The actual model divides the possible routes for a train into *segments*, each of which consists of a part of the train's route that must be used or not in its entirety. Segments start and end at the points where alternative routes split off or merge, as well as at the start and end of the entire route. Each segment has an associated binary variable that expresses whether the train uses the segment or not.

The constraints modelling the schedule of a single train are modified to take alternative routes into account. Firstly, constraints are added to constrain the segment use variables to express a single consistent route. Mathematically, this is modelled as a flow problem. The first and last segments for a train must be used, i.e. have a flow of one, and flow must be conserved at every segment junction: whenever the train uses one segment that ends at a point in the rail network, it must also use exactly one of the segments that start there, and vice versa.

Other constraints in the model are altered to ensure that they only apply when the train actually uses the relevant part of the route. This concerns the constraints relating the exit time from one segment to the entry time of the next, as well as constraints added for avoiding conflicts that are discovered. For instance, a constraint expressing "train A must exit track T before train B enters it –or– train B must exit before train A enters" is extended with extra terms expressing " –or– train A does not use the track –or– train B does not use the track". This is achieved using big-M terms, similarly to other conditions already expressed in the MILP.

7.1.4 Preliminary computational results

The model and algorithm are run frequently as a part of development, to verify that each completed part works correctly on data that are similar to the actual use cases. The network used is shown in Figure 7.1.2. It contains several lines with 74 stations and 148 connecting tracks at the coarse level, further split into 145 stations and 332 tracks at the more detailed level. The reference timetable covers a timespan of about 8 hours, for 318 trains.

We run the solver on two versions of this data. In the first version, we simply aim to remove any conflicts in the reference timetable, which is not conflict free. Solving this takes 28s and 7 iterations. 79 trains are modelled, being involved in a total of 148 conflicts.

In the second version we add a station possession as per the use cases. All tracks in the station Finale Ligure are unavailable from 07:00 to 08:00, requiring some trains to avoid that time interval. The solving time is similar: 29s for 6 iterations. As we now start from a feasible initial timetable, less trains and conflicts are modelled: 15 trains and 80 conflicts. Figure 7.1.4 shows the most interesting part of the solution, where the four trains that should pass the station while closed, have been delayed. The reference timetable is represented by black lines, and the solution in red.

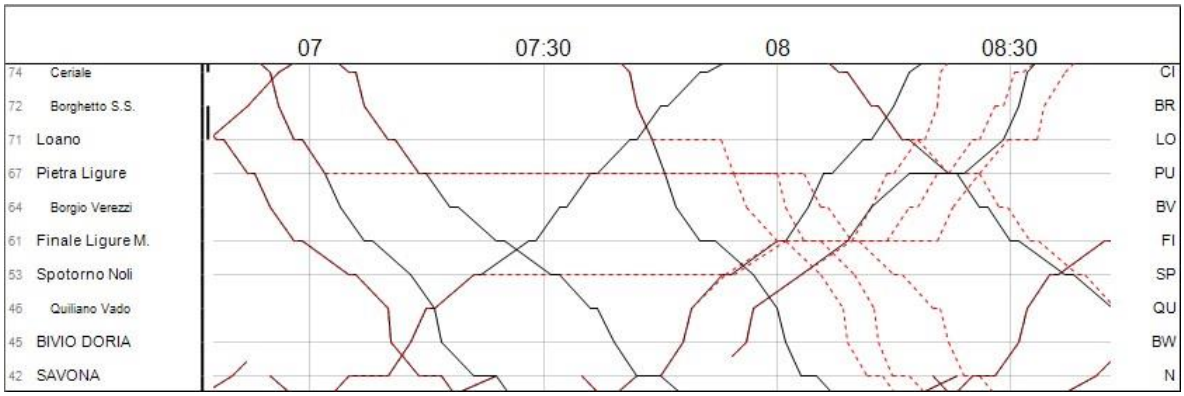


Figure 7.1.4: Optimized solution for a station possession

The results of this collaboration comfortably reached TRL4, and we are on track to reach TRL6 by the end of WP7.

7.1.5 Future work

These preliminary results will be improved through the work in WP7. Among the things we plan to do are the following:

- The model and solver will be extended to cover the remaining use cases, including track possessions and speed restrictions.
- Checking the accuracy of the mathematical model compared to the real-world rules that it should model is an ongoing process. The optimized solutions will be scrutinized for discrepancies and the model updated as required.
- In some stations, the compatible tracks can depend on whether one train is overtaking another train or not. This is an additional complexity that requires the track assignment solver to be extended.
- The algorithmic machinery for modelling only subsets of trains is general and can be reused for further algorithm improvements. For example, there are often situations where the modelled trains can be divided into independent groups, where no conflict has yet been found between trains from different groups. This means that the iteration could be performed faster by solving multiple independent MILPs, one for each group, as this is usually faster than solving one big MILP. Another possible improvement is to create smaller MILPs that focus on discovering all required constraints for a subset of trains in the challenging parts of the problem. This often requires multiple iterations that are slowed down by modelling additional irrelevant trains.
- An extra challenge of alternative routes is the fact, mentioned above, that station track compatibility can depend on the route chosen. This introduces an extra dependency between the main MILP and the track assignment solver. Updating the solver to handle this correctly is ongoing work.

7.2 An algorithm for adjusting train timetables in case of TCRs

7.2.1 Background

Temporary capacity restrictions (TCRs) are frequently occurring in railway systems due to poor infrastructure or maintenance work and prevent trains from operating as planned, since tracks or paths might be closed or have speed limit restrictions. For major infrastructure restrictions known in advance, a new timetable for the time period concerned is often made. This rescheduling is

currently time-consuming and done manually. Therefore, for smaller TCRs or TCRs that appear with short notice, it is not possible to manually reschedule the timetable. Instead, trains are delayed, and some might be cancelled, and it is up to the traffic dispatchers to solve the conflicts as good as possible.
To increase timetable quality in the case of a TCR, timetable planners need decision support to assist them in creating a modified, conflict free and robust timetable. The aim of the method we develop is to aid the planner in the process of rescheduling also during smaller TCRs or TCRs that appear on short notice. We intend to provide a tool that makes it possible and time efficient to reschedule during such TCRs.

The case-specific situation is that the TCR is known one to a few weeks in advance and will be ongoing for one to a couple of weeks. A new timetable during a TCR of this magnitude is valuable for the train operating companies and the dispatchers, but there is seldom time for creating one today. The specific TCRs considered here are a closure of one of the tracks at a double-track line together with a speed restriction on the remaining track, and a speed restriction on a part of a single-track line.

7.2.2 Method

We use a mixed-integer linear programming (MILP) approach, (based on a big-M formulation), as often used in previous literature. Our work is based on the MILP, and the approach of incremental timetabling, presented in Kloster et al. (2023). In their work, the purpose of the method is to quickly add, or modify, individual train paths. When making such changes to an existing timetable, planners must often spend a lot of time manually making several small adjustments to make the timetable feasible. So, with the incremental method used by Kloster et al. (2023) the aim is to quickly solve the problem of finding a feasible timetable, thereby giving planners time and possibility to try, and evaluate, several tentative timetables. By doing so, the knowledge of the planners steers the solution by incrementally improving the quality of the timetable as the work progresses.
Now, we will use this method to modify several train paths (that need to be modified) when a TCR occurs. When a TCR is imposed on an existing timetable, this will in most cases lead to the timetable becoming infeasible. The purpose of our algorithm is to find a modified feasible timetable that is as close as possible to the existing target timetable.
However, depending on the severity of the TCR, the infrastructure manager might need to choose a set of trains in the original (annual) timetable to cancel to be able to (or make it easier to) find a feasible solution, thus creating a new target timetable. Which trains to cancel should be chosen based on area and line-specific criteria and furthermore consider socio-economic aspects. How this selection of trains to cancel is done, is not part of the algorithm itself, but rather the “knowledge of the planners” that come into play in the iterative process where the planners draw conclusions from one evaluated tentative timetable, trying to find an improvement by creating another one to evaluate.
This means that for each target timetable the planners construct (by removing a subset of trains from the original (annual) timetable), a modified feasible timetable that is as close as possible to the target timetable, is created. Feasible means here, just as in any train scheduling problem, that the created timetable meets all operational rules; free-running, precedence, safety margins and connections. Kloster et al. (2023) describe how all of these are modelled in the MILP, however in the use-case for which this algorithm is intended, only traffic on a line is considered and consequently we do not need to consider any connections.
Many timetable rescheduling methods focus on real time, i.e., rescheduling must be fast

(dispatching problem). In the problem considered here, more computational time is available, however, the planner should be able to, quite quickly, evaluate several (or at least a few) alternative sets of train cancellations. Therefore, the algorithm must be fast enough to be useful in such a decision support tool. Since we want to find solutions that are "close" to the current timetable, in order not to introduce unnecessarily large changes to the timetable, we start from the existing timetable and make changes to it instead of creating a new timetable from scratch.

7.2.2.1 Assumptions

To keep the problem as small as possible and since it is suitable for the geography of the specific use case we are interested in, the problem is modelled macroscopically. This requires some assumptions which are:

- We do not consider any block-sections, instead macroscopic time separation, modelling blocking times and sufficient margins, is used.
- A track between two consecutive stations is only considered as one resource. This means that several trains can occupy the resource at the same time if sufficient arrival time separation is upheld. This also means that a track resource is always followed by a station resource, which in its turn is followed by a track, and so on, as illustrated in Figure 7.2.4.

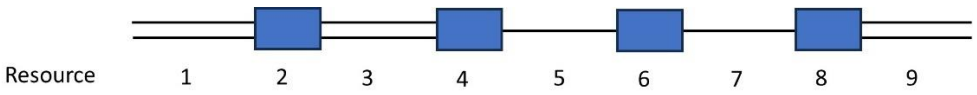


Figure 7.2.4. Conceptual model of the simplified railway line structure.

- Track-resources are either single-track or double-track. I.e., in the considered case, no track section has capacity >2.
- Directions are specified. I.e., on double-track sections the direction of each track is given.
- Capacity in stations. Stations can have any capacity greater than or equal to one. No paths through stations are considered. Hence, all tracks out of a station can be reached from all tracks coming into the station. And we allow this without considering any potential blocking of crossing paths. In this way, station capacity is reduced to simply describe the number of trains that can occupy a station at the same time.
- Connections between trains are not considered.

One practical extension to the model of Kloster et al. (2023) is needed. Different speed profiles, (for different train types), are necessary for different stop patterns. Meaning that the running time through a track-resource will depend on whether the train stops at any or both stations, or just runs through at full speed. To model this, additional variables, and constraints, are needed. This type of addition to the MILP has previously been described by e.g. Bach et al. (2019) and Gestrelius et al. (2015). Bach et al. (2019) model it in a slightly different way than what we need. In their work it is assumed that the extra run time needed if stopping at both stations is the sum of the extra times needed if only stopping at one of the stations individually. In our real-world data, we see that this is not true in practice. Instead, we use the approach of Gestrelius et al. (2015) where no relationship between different run times is assumed.

7.2.2.2 Interactions

Given the assumptions, only some train interactions are necessary to consider when constructing the MILP model.

1. Trains going in opposite directions on single track segments cannot occupy the same track resource at the same time, hence one train must precede the other. This is also true in stations with capacity = 1, regardless of the direction of the trains.
2. Two trains following each other can occupy the same track at the same time but must be separated by some headway. Furthermore, since trains are assumed to only run on at most one track in each direction, trains must be prevented from overtaking each other on track segments.
3. For stations with capacity > 1, trains can overtake each other and can be present in the station at the same time. However, the station has a limited capacity and thus only a limited number of trains can be present simultaneously.

7.2.2.3 Algorithm

Even after these assumptions, which are made to keep the problem as small as possible, the problem will grow quickly with the number of trains. To solve the problem efficiently, we will follow the method presented in Kloster et al. (2023) and add constraints when they are needed. Most constraints will control interactions between trains that can only occur if a substantial change is made to the timetable. Since our objective is to find a timetable close to the original one, these situations will likely not appear. Hence, there is no reason to include these constraints from the beginning.
Therefore, we initially formulate the MILP only including free-running constraints. Each iteration then consists of the following steps, as described in Kloster et al. (2023):

1. Solve the current MILP.
2. If infeasible, stop.
3. Check whether the solution contains any conflicts, i.e., violates any operational rules.
4. If not, the solution is feasible and optimal. Stop.
5. Otherwise, augment the MILP with constraints to solve the identified conflicts.
6. Repeat

The MILP formulation is implemented in Julia/JuMP and solved using Gurobi.

7.2.2.4 Model and implementation

This section provides a brief overview of the model. More details of the model and implementation are provided in Appendix D.
The initial model only considers “free-running constraints”, i.e., when there are no interactions between trains. This is the base of our model, that the algorithm starts with in Step 1. In Step 3, violations are found and constraints needed to resolve these violations are added.
To augment the MILP in the described way in each iteration of the algorithm (see Section 7.2.2.3), we need to find any conflicts (violations of rules) to the interactions mentioned in Section 7.2.2.2. For more detailed descriptions of why these rules are needed, we refer to Kloster et al. (2023). Based on the identified conflicts we add the constraints needed to prohibit the conflicts. Appendix D describes the original MILP-formulation in detail as well as all constraints added for all types of conflicts.

7.2.3 A small test case

In this deliverable we are considering a section of the cross-border line (Malmö - Oslo/Alnabru) as a small test case. In the following sections, we describe the test case and how we apply and evaluate our method on this test case.

7.2.3.1 Test case description and data preparation

The input, i.e. the macroscopic infrastructure and annual timetable for the whole of Sweden, is stored in a database. Since geographical locations of all infrastructure objects are available in the database, the data can be handled through a geographical information system, for example to visualize or select parts of the infrastructure for the replanning. This is illustrated in Figure 7.2.5.

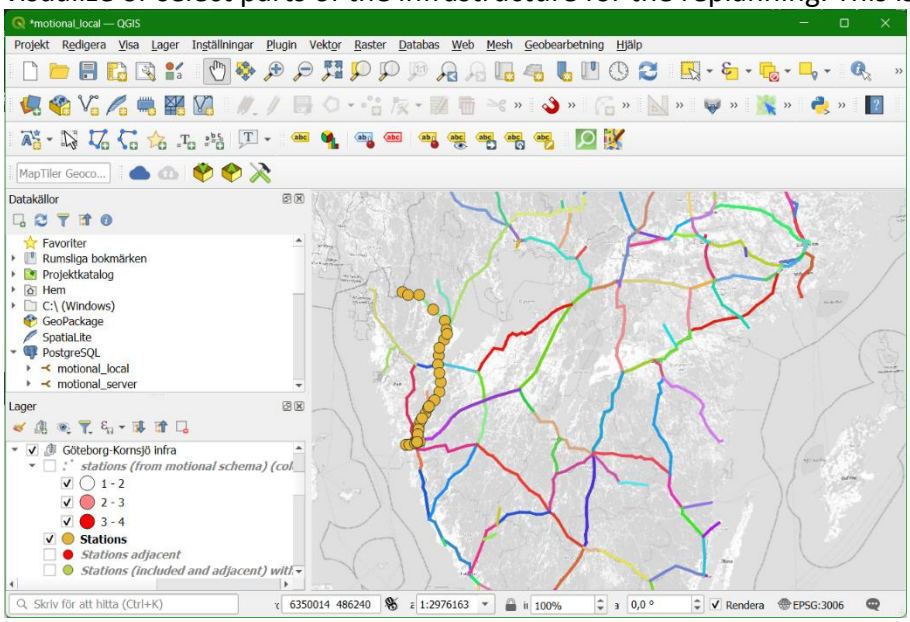


Figure 7.2.5. Illustration of selecting infrastructure data in the in a geographical information system QGIS with a background map from © OpenStreetMap contributors.

The segment of the cross-border line that is considered in this test case is the line from Gothenburg (Göteborg Marieholm) to the Swedish-Norwegian border (Kornsjö). This segment consists of 47 resources: 24 stations and 23 track segments. The TCR considered is the closing of one track on the 8 km double track segment between the stations Trollhättan and Velanda Södra and a reduction of the speed limit to 70 km/h on the remaining track. The geography of this test case and TCR are illustrated in Figure 7.2.6.

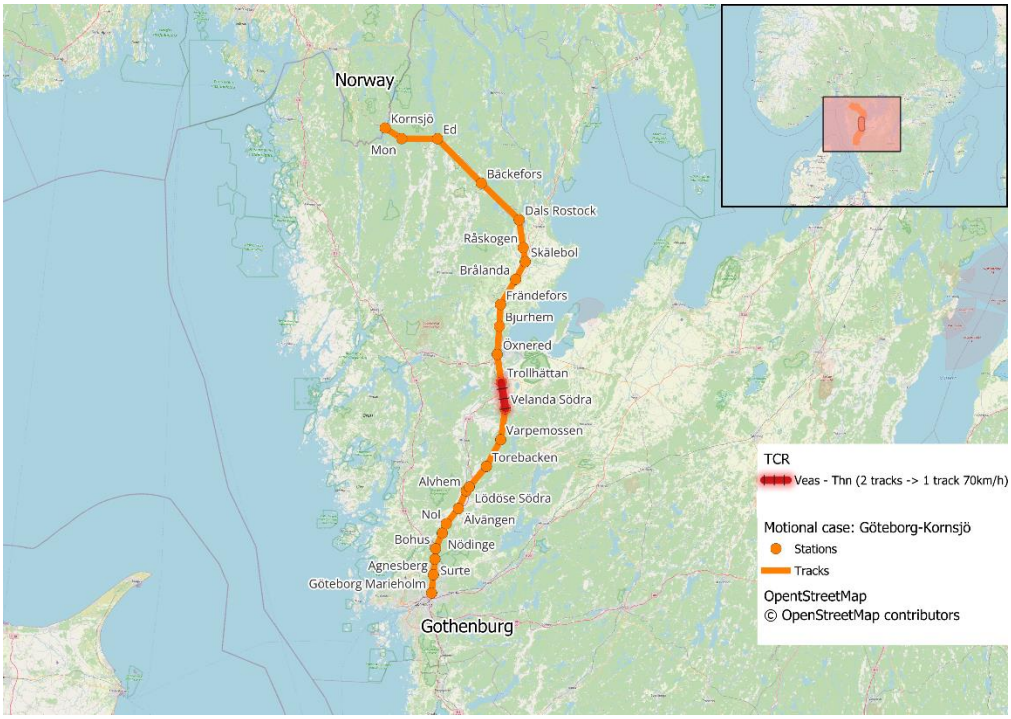


Figure 7.2.6. Illustration of the test case based on the line between Göteborg Marieholm and Kornsjö with a double track TCR between Velanda Södra and Trollhättan. Background map from © OpenStreetMap contributors.

The timetable of one running day, May 5th, 2021, is considered in this test case. It contains mixed traffic of passenger, freight and service trains with a total of 233 trains with varying train paths. Out of these, 105 trains are passing the track segment where the TCR occurs. The input (original) timetable is illustrated in Figure 7.2.7. During the TCR, this original timetable is not conflict free. A part of the input timetable with illustrated conflicts is presented in Figure 7.2.8.

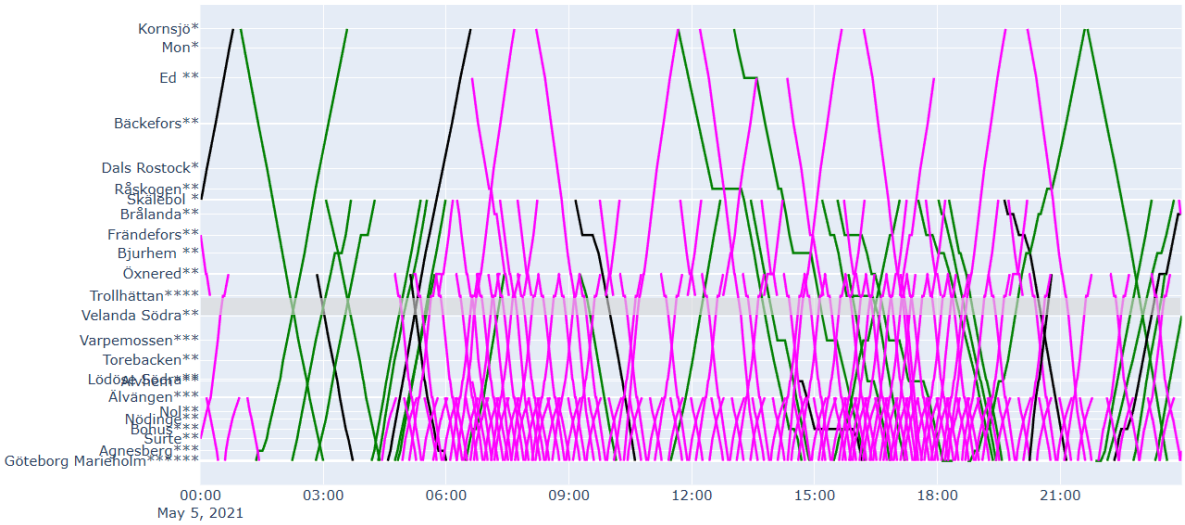


Figure 7.2.7. Timetable graph of the original timetable on the Göteborg Marieholm - Kornsjö line. Passenger trains in pink, freight trains in green, and service trains in black. The section with TCR between Trollhättan and Velanda Södra is marked in gray.

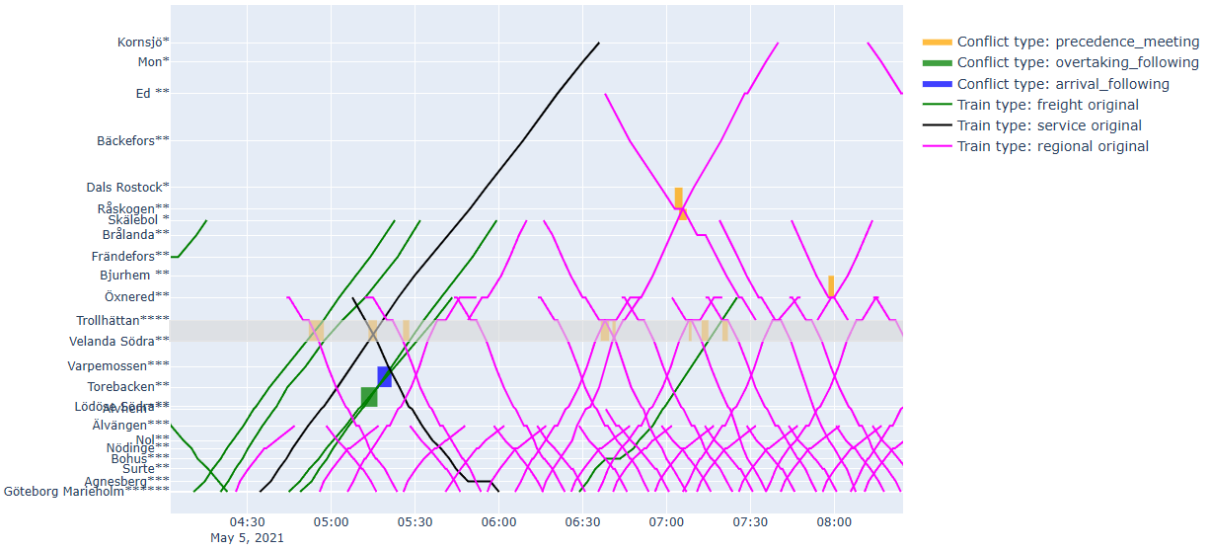


Figure 7.2.8. Example of conflict markings on timetable graph on a part of the original timetable. Passenger trains in pink, freight trains in green, and service trains in black. The section with TCR between Trollhättan and Velanda Södra is marked in gray.

Some model parameters are adjustable in the model to control how trains are allowed to be scheduled in the output timetable. In this test case, the parameters are set to be generous but reasonable and a base for future work. Specifically, the parameters are set to allow trains to be scheduled up to 1 hour earlier and 3 hours later at any resource and at most have their running time doubled over their full train path. Furthermore, trains going in the same direction are set to be separated by at least 1 minute on arrival at every resource. Trains going in different directions using a single capacity resource are set to be separated by 3 minutes. The full list of model parameters used in the model are listed in Appendix D in Table B.1.
In the test case, the objective is set to minimize the difference from the original timetable normalized over the length of train paths. The base model (free running) for the test case contains 5,192 timetable points to schedule and in total 15,343 variables (10,384 continuous and 4,959 binary) with 33,282 constraints.

7.2.3.2 Results

The algorithm produces a conflict-free output timetable in 10 iterations, taking roughly 50 seconds (varies from 30-70 seconds) on a laptop with AMD Ryzen 5 PRO 5650U @ 2.30 GHz and 16,0 GB RAM. Approximately 98 % of the time is spent in the solver (Gurobi Optimizer version 11.0.2 build v11.0.2rc0 (win64 - Windows 11.0 (22631.2))) when running with an absolute gap tolerance of 100 and a relative gap tolerance of 3 %. A segment of the output timetable is illustrated in Figure 7.2.9.

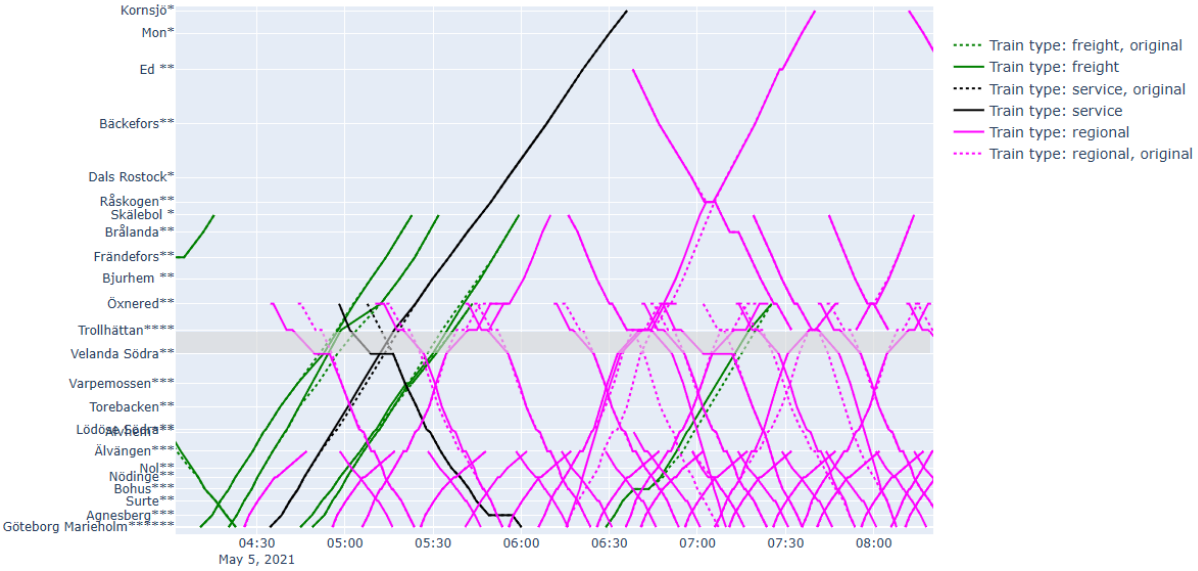


Figure 7.2.9. Segment of final timetable with changes. Dashed lines are original times. Passenger trains in pink, freight trains in green, and service trains in black. The section with TCR between Trollhättan and Velanda Södra is marked in gray.

Table 7.2.3 describes the number of identified conflicts of different sorts in each iteration. As seen from the values, there are many conflicts found during the first iterations, and quite few during the later iterations. In direct relation to this we can from Figure 7.2.10 see that the number of binary variables increases most in the first iterations and quite little in the later iterations. Overall, the number of binary variables increases approximately 10% during the ten iterations. (From ca. 4959 variables to 5447.) In the same way, of course, the number of constraints increases, see Figure 7.2.11.

Table 7.2.3. No. of identified conflicts, and objective value, in each iteration.

Type of conflict	**it1**	**it2**	**it3**	**it4**	**it5**	**it6**	**it7**	**it8**	**it9**	**it10**
Headway	8	19	25	13	8	3	1	1	4	0
Overtaking	2	7	6	5	2	0	1	1	1	0
Crossing opposite	68	15	6	4	1	2	0	0	0	0
Crossing same	0	0	0	0	0	0	0	0	0	0
Capacity	0	5	3	4	0	0	1	0	0	0
Total	**78**	**46**	**40**	**26**	**11**	**5**	**3**	**2**	**5**	**0**
Obj_value	2870	5681	6051	6137	6205	6236	6212	6209	6196	6197

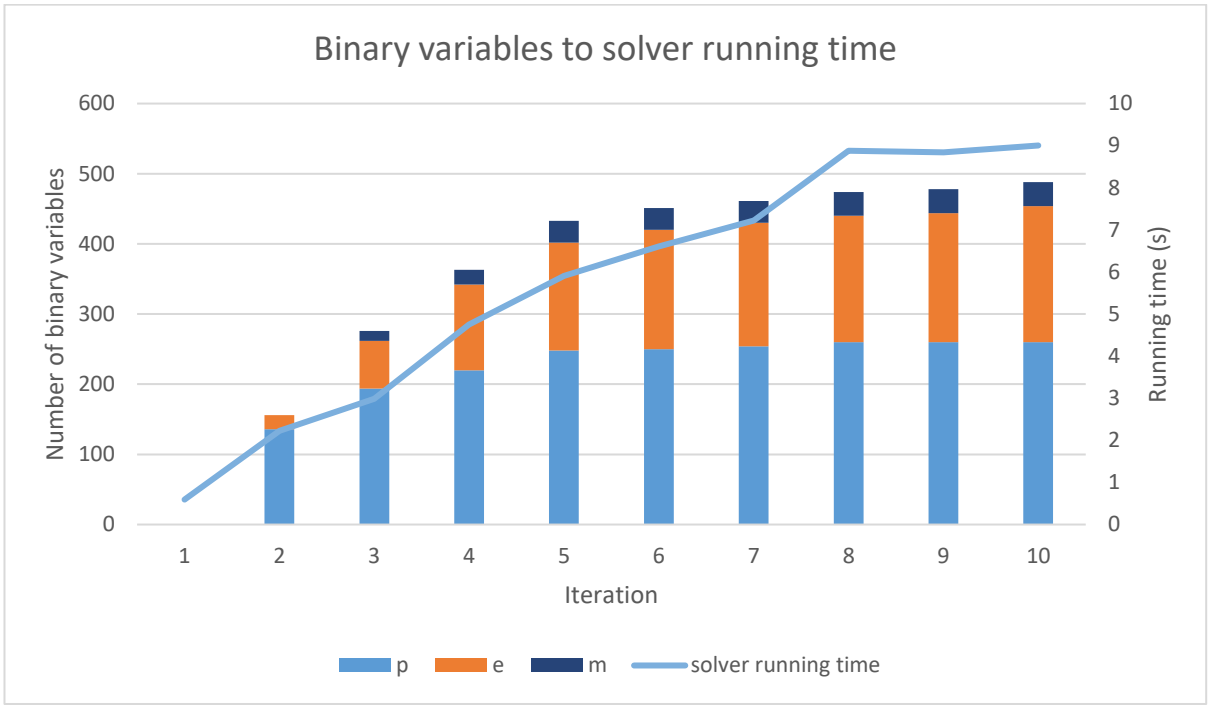


Figure 7.2.10. Number of binary variables compared to solver running time per iteration.

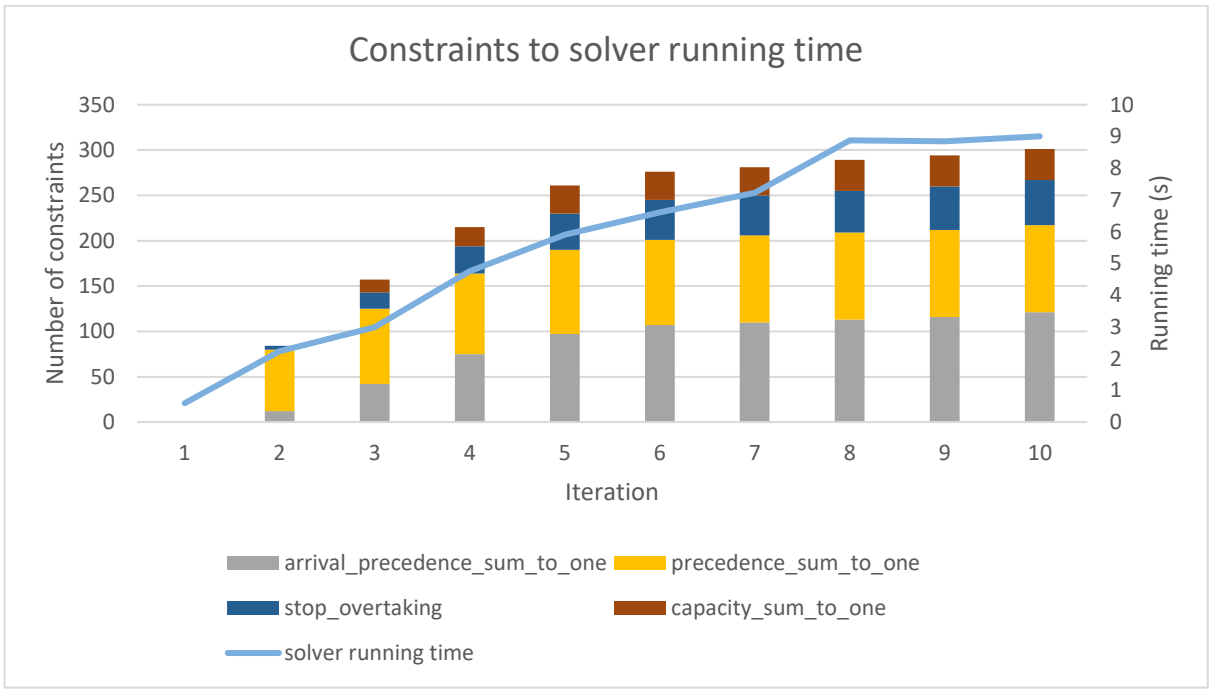


Figure 7.2.11. Number of constraints to solver running time per iteration.

Regarding the solution time to run the algorithm for the ten iterations needed in this test case, we can see two interesting things from Figure 7.2.12; it clearly shows that finding conflicts and adding constraints is done very quickly, while most of the time needed is to actually solve the problem, and that each iteration requires a few seconds to complete. We know that it is possible to improve the implementation to decrease the computation time needed (for the solver). This is something we will do in the future.

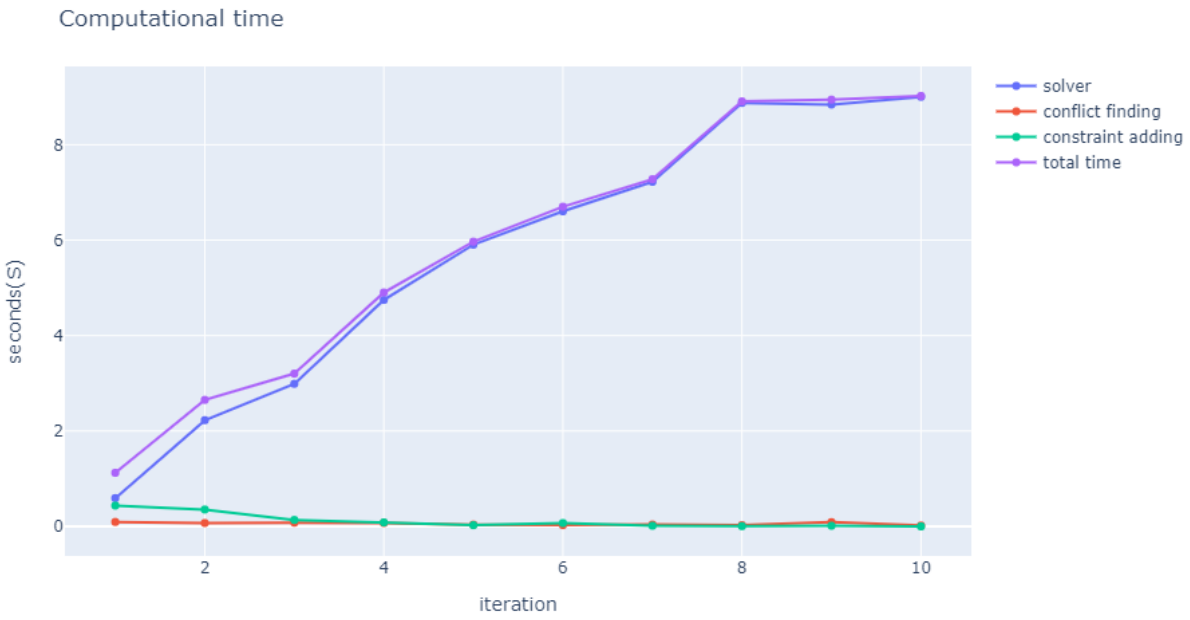


Figure 7.2.12. Computational time over algorithm iterations.

7.2.3.3 Conclusions and future work

The results from the small test case show that the presented method is useful for the considered type of TCR. The method has shown to produce a feasible timetable within acceptable computational time. However, there are still improvements to be made.

We have found that changes to the objective function and parameter settings should be evaluated to get output timetables that are more desirable in practice. This will be done in the continuation of this project. The produced timetables from our algorithm will also be evaluated from several KPIs and evaluated by simulation. There are also implementation improvements that can be made to speed up the computations.

7.3 Algorithm for constructing a nation-wide adjusted hourly timetable

7.3.1 Background

Preventive maintenance of the railway infrastructure necessitates the closure of some parts of the networks for a few days, forcing adjustments to the annual timetable. In the Netherlands, the scheduling of the maintenance works starts 6 to 12 months before the operations, followed by the actual adjustments of the timetable until 8 weeks before the operations, after which the rolling stock and crew schedules are adjusted. The problem of finding an adjusted timetable arises quite often: almost every weekend has maintenance works at multiple locations. The timetable adjustment process is a cooperation between the infrastructure manager ProRail and the passenger and freight operators.

The Dutch railway timetable is periodic with a period of one hour. The timetable adjustment process considers the basic shape of a day's timetable, assuming that it arises by repeatedly carrying out a one-hour timetable. Therefore, the problem setting of this research features one-hour timetables.

The timetable adjustment process is composed of two main parts. The first part decides which maintenance works should be carried out on which days. These decisions must respect both the availability of the engineering equipment and crews, and the by-train reachability of the major population centres. The first part focuses on capacity planning, without working out the trains' adjusted departure and arrival times.

The second part of the process produces a conflict-free adjusted hourly timetable (AHT) in which the services of the annual timetable may be adjusted by being fully cancelled, partially cancelled, re-routed, and/or shifted in time. In addition, bus services are ordered if a temporarily out-of-service railway link has no reasonable detour option in the remaining railway network.

The AHT is valid for the duration of a given set of infrastructure maintenance works. This means that most weekends need multiple AHTs since many works finish by Saturday evening and many others start on Sunday morning. Moreover, the maintenance works are scattered throughout the country and tend to affect the flow on multiple corridors of a highly interconnected railway network (see Figure 7.3.1). Therefore, it is desirable to consider the entire country's AHT, rather than splitting up the problem geographically.

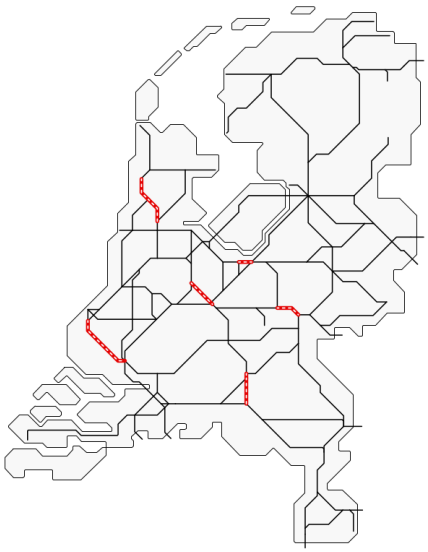


Figure 7.3.13. An example for a set of infrastructure segments in the Dutch railway network that are unavailable on a particular day due to preventive maintenance.

In this work we focus on supporting the second part of the process. The goal is to design an AHT that facilitates the passenger and freight flows on the available infrastructure. It is preferred to minimise cancellations and to adjust the departure and arrival times as little as possible. The latter goal helps limit the impact on the rolling stock and crew schedules as well as on connections with other modes of public transport.

There is no decision support tool available to help computing the adjusted timetable; it is an entirely manual process. Considering the large number of AHTs, the planning department is under constant time pressure which gives room for working out the details of one AHT, without being able to explore alternative ideas.

This research aims at solving problem instances that are relevant for real-life decision making. In the highly interconnected Dutch railway system, this necessitates considering the entire network, including all passenger and freight services. Our approach is designed for the AHT problem on a macroscopic level. We do admit that the solution may not be conflict-free on a microscopic level. We are not aware of any approach that would allow for optimising the entire country's microscopic railway network. This limitation can be mitigated in two ways. First, macroscopy is likely to be sufficient in the early stages of the process where the possible maintenance possessions are scheduled or where a first sketch of an AHT is constructed with sufficient time to resolve the microscopic conflicts. Second, our future research will investigate the prospects of post-processing tools that transform a macroscopically feasible timetable into a microscopically feasible one.

7.3.2 Outline of the proposed method

This research builds upon the model of Van Aken et al. (2017a,b) for the macroscopic AHT problem. Their model decides on the cancellation of some services and on the departure and arrival times of the remaining services. The objective is to minimise train cancellations as well as the deviation from the annual timetable's departure and arrival times. We extend this model by proposing the Network Adjusted Hourly Timetable (NAHT, see Section 7.3.4) model. We enhance the model's scope to include decisions on the trains' turning patterns at their terminal stations. Moreover, we propose a solution method that enables us to scale up the AHT calculations for problem instances of practically relevant size and complexity.

In addition, we propose the Macroscopic Track Usage (MTU, see Section 7.3.5) model which is inspired by the work of Van Lieshout (2021). The MTU model takes the outcome of NAHT, and determines the necessary number of tracks, thereby identifying the stations where the NAHT solution exceeds the available track capacity.

In this work we propose an iterative and interactive approach for the AHT problem where we combine the NAHT and MTU models with the insights of field experts, see in Figure 7.3.2. Each iteration consists of two stages. In the first stage, we solve the network-level timetabling problem (i.e., NAHT) ignoring the stations' limited platform capacity. In the second stage, we solve the MTU model to identify the track bottlenecks of the NAHT solution. Then the field experts suggest modifications of the NAHT model's specification to resolve the capacity bottlenecks. The modifications can include the cancellations of some services, re-arranging the turning patterns and/or requiring a certain time separation between the departure and arrival events. The modified NAHT problem specifications apply to the next iteration's calculations. The interactive algorithm terminates when all track capacity violations are resolved.

While an integrated optimisation model with network- and station-level aspects is desirable from a theoretical point of view, this research chooses for an iterative approach for pragmatic reasons. Our goal is to find AHT solutions of a good quality in a few minutes, even if the solutions may not be optimal. However, the integrated model is unlikely to be suitable for this goal because NAHT problem instances can be very challenging to solve even without track capacity. The addition of track capacity constraints to NAHT escalates both the memory consumption and the computing times.

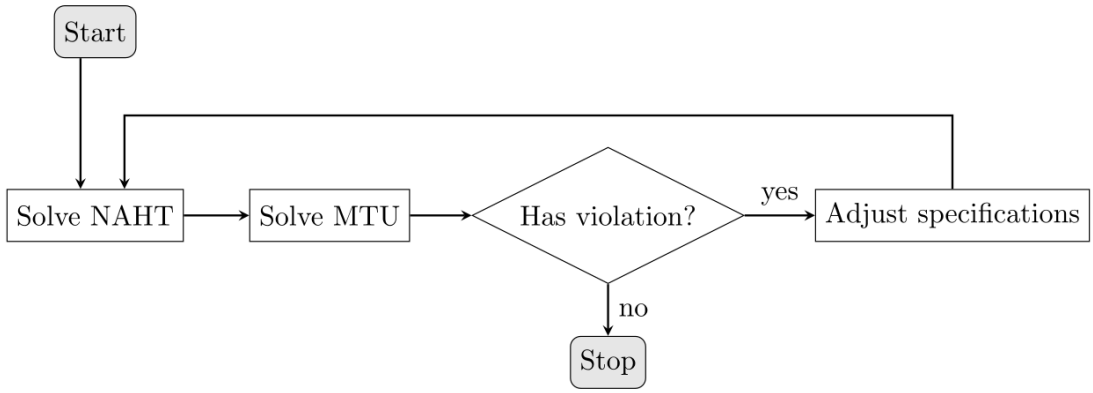


Figure 7.3.14. The structure of the proposed iterative framework.

7.3.3 Problem formulation

We consider instances of the adjusted hourly timetabling (AHT) problem in the following form. Given is the available infrastructure, consisting of nodes and segments between the nodes. In our

application, the nodes are the railway stations and the junctions with a given number of available platform tracks and non-platform tracks. However, we disregard the nodes' internal routing limitations. Each infrastructure segment connects two nodes and has a given number of parallel tracks.

We are also given the list of passenger and freight services, as well as the nominal hourly timetable specifying the departure and arrival times of each service at each node. In periodic timetabling, the departure and arrival times are minutes of the clock-face. That is, a service may enter a segment at 58 minutes of the hour and leave it at 2 minutes.

The maintenance possessions are given as a list of infrastructure limitations: some segments and nodes have fewer available tracks available than in the nominal timetable. A segment can be blocked fully or partially.

The AHT problem modifies the nominal hourly timetable by the following decisions:

- Services can be fully or partially cancelled. For each service, some of its visited nodes are marked as a possible cutting point. Then the parts between any neighbouring pair of cutting points can be cancelled. In particular, it is possible to cancel the middle part of a service, and to keep operating the parts on either side of a possession. The non-cancelled portions of a service form contiguous parts. The starting events (or finishing events) of a service are the first (or last) events of its contiguous parts.
- The nominal departure and arrival times of the services can be adjusted.
- Services can be re-routed by selecting a route from an explicitly given list of candidate routes. This option is particularly important for freight services that need to be able to reach the harbours and industrial sites.
- Whenever a passenger service has a finishing event at a node, this event will be connected to the starting event of another service at the same node by a turning.

The adjusted timetable needs to fulfil various requirements. The travel times and dwell times of each service must lie between given lower and upper bounds. Headway restrictions ensure a safe time separation between services that use the same infrastructure elements.

Our models connect at each infrastructure node the services' finishing events to the starting events by choosing the turnings by a perfect matching. The selected turnings must be sufficiently long so that the drivers have enough time to walk from one end of a train to the other end. The turnings allow for a steady flow of rolling stock and crews when the one-hour timetable is carried out repeatedly.

The solution must not exceed the nodes' track capacity, both for the platform tracks and for the total number of tracks. Track capacity is consumed by dwelling services, by passing services as well as by turnings.

In our application, freight services are merely reserved paths that are going to be filled in a few days or weeks before the operations. In contrast, the adjusted hourly timetable is computed several months earlier. The nominal timetable contains many freight services that share the time slots on a large part of their routes; obviously, at most one of these paths will be used in each concrete hour. Our models relax the conflicts between freight services. Intuitively, our models ensure that the passenger services are macroscopically conflict-free, and that each freight service can run in the time slots between the passenger services. In addition, freight services are not included in turnings because they do not form a strictly repeating hourly pattern.

The optimisation objective is to minimise penalties for train cancellations and for shifting the nominal event times. In essence, we penalise the deviation from the nominal timetable. According

to our objective function, the nominal timetable is the unique optimal solution of the AHT problem if there is no maintenance possession.

7.3.4 Network Adjusted Hourly Timetable (NAHT) model

The NAHT is an extension of the work of Van Aken et al. (2017a,b) which is in turn an extension of the Periodic Event Scheduling Problem (PESP). For the detailed description of the NAHT model, we refer to Appendix E.

The NAHT model contains decision variables to indicate whether a service is cancelled on a portion of its paths. Moreover, the model decides on the scheduled departure and arrival times and chooses the turnings between finishing and starting events.

The model's constraints express that the non-cancelled services fulfil all travel time, dwell time, turning time and headway requirements. These constraints all appear in the form where the time difference of two events lies between a certain lower and upper bound. In addition, the NAHT model can re-route services by choosing (at most) one route form given lists of re-routing possibilities.

The objective function of the NAHT model minimises the sum of the penalties for undesirable features such as the cancellation of a service, and a shift of an event. The time shift penalties are proportional to the magnitude of the shift. We consider different penalties for forward and for backward shifts. The cancellation penalties are proportional to the cancelled geographic distance. In addition, the model penalises the creation of too many contiguous parts of the services as a result of multiple partial cancellations.

7.3.5 Macroscopic Track Usage (MTU) model

The station-level MTU model takes the outcome of the network-level NAHT model and determines whether the stations' track capacity is sufficient. Capacity violation takes place if the dwelling, passing and turning activities consume too many tracks of a station. Platform track capacity often turns out to be a problem at railway nodes just outside the maintenance possession that become turning locations due to cancellations. These nodes can be small stations with few platform tracks. The proposed MTU model computes how many platform tracks and non-platform tracks are minimally needed to operate the timetable. For the detailed description of the NAHT model, we refer to Appendix E.

The MTU model for a given station considers the arrival and departure events of the services as scheduled by the NAHT model. Track occupation is caused by the dwelling or the passage of a service or by the turning between services. In both cases, an arrival event is linked to a departure event. Successive track occupations on the same track can be viewed as a follow-up relation from a departure event to an arrival event. The dwell, turning and follow-up relations form a graph representation of the problem, see Figure 7.3.3.

The main idea behind the MTU model is the observation that the track usage can be described as a set of directed cycles in the graph. The arcs along the cycles alternate between track occupation (i.e., dwell, passage or turning) and follow-up. The MTU model uses these cycles to calculate the necessary number of tracks.

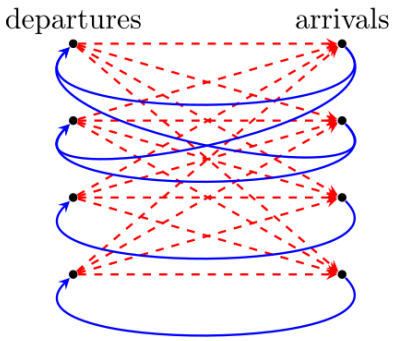


Figure 7.3.15. The directed graph behind the MTU model. The dwelling, passing and turning arcs are represented by solid blue arrows, the follow-up arcs by red dashed arrows.

7.3.6 Iterative framework with the NAHT and MTU models

We design an interactive and iterative framework to find good solutions to the adjusted hourly timetabling problem, see Figure 7.3.2. In each iteration, the network-level NAHT model is solved without station capacity limitations. Its outcome serves as input to the station-level MTU model to identify track capacity violations. A human decision maker analyses the violations, if there are any, and suggests modifications to the NAHT model specifications of the next iteration. The feedback actions in our experiments include (*i*) the forced cancellation of a service on some parts of its path; (*ii*) the adjustment of the turning patterns; and (*iii*) the adjustment of departure and arrival times.

An example of such feedback considerations is shown in Figures 7.3.4 and 7.3.5. Each coloured rectangle indicates a track occupation, starting with an arrival event and ending with a departure event. The track occupations of station 'Alm' in the first iteration has some turnings that are longer than 30 minutes (see Figure 7.3.4). The solution requires 5 tracks, however, the station has only 4 tracks. The human decision maker can suggest using alternative turnings at this station that better fit to the departure and arrival times. The adjusted turnings involve the circled events. Figure 7.3.5 is the track occupation in the second iteration. The shorter turnings save two tracks.

7.3.7 Hot starts for the NAHT model

The NAHT model belongs to a family of optimisation problems that often turns out to be difficult to solve. We observe this difficulty in instances with partially blocked infrastructure segments. The commercial mixed integer linear programming (MILP) solvers (such as CPLEX or Gurobi) struggle to find good feasible solutions and tight bounds.

A common speed-up technique for MILP solvers is to compute feasible solutions, and to provide these solutions as hot starts to the MILP solver. Hot starts often have a dramatic improvement on the solution times of large and complex optimisation problems.

We generate hot start solutions by imposing heuristic restrictions on the NAHT model's solution space. We are motivated by the intuition that the maintenance works are unlikely to lead to cancellations or time shifts in services in other geographic parts of the country.

For our experiments, we use 5 hot start calculator models. Each hot start calculator has its own definition for what it means for a service to be affected by the possessions. The hot start calculators amount to solving the NAHT model subject to the additional restriction that the unaffected services keep their nominal timetable without any cancellation or time shift. Thanks to the smaller solution space, we expect the MILP solver to find good solutions quickly for the restricted hot start models.

In our numerical tests, we compare two methods for solving NAHT problem instances. The Direct Method (DM) amounts to solving the NAHT model by a commercial MILP solver without hot starts. The Hot Start Method (HSM) amounts to solving the 5 hot start calculation models, followed by solving the unrestricted NAHT model.

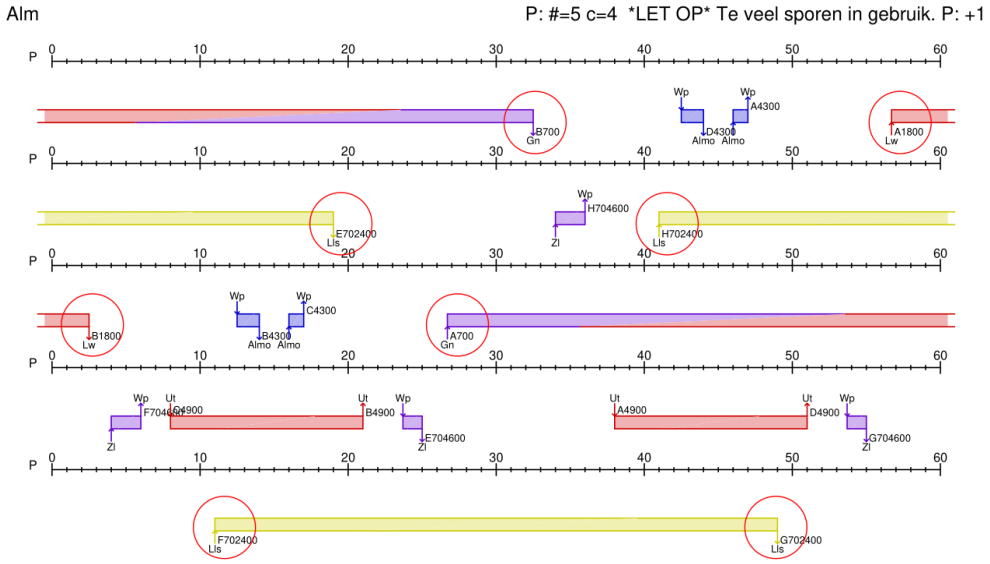


Figure 7.3.16. The track occupation of the first iteration's NAHT solution, using 5 tracks, thus violating the station- 9 -capacity of 4.

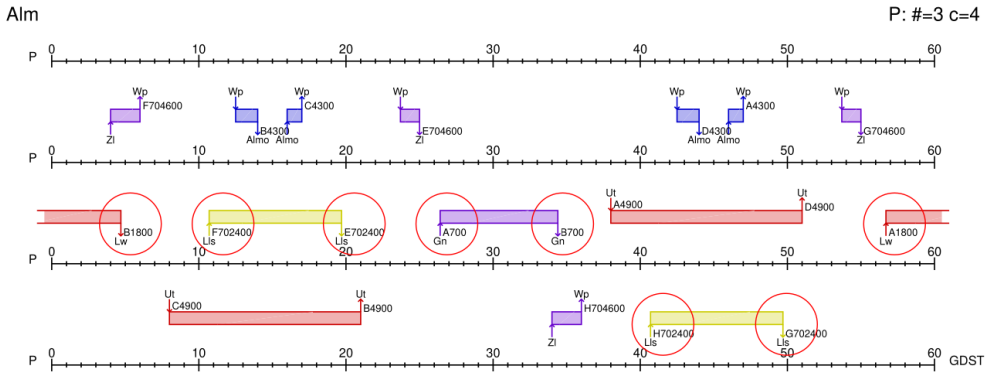


Figure 7.3.17. The track occupation in the second iteration with updated turning possibilities, using only 3 tracks.

7.3.8 Computational tests

In this section we summarise our computational results. For more details we refer to Appendix E. We test the proposed solution approach on problem instances that are derived from the real-life 2024 timetable of the Netherlands. The test data consist of 292 passenger services of NSR and 108 freight services, altogether with over 9500 departure and arrival events on a network of 293 stations and 207 junctions. These services form the nominal timetable.

We consider 9 test instances that arise from the nominal timetable by selecting one or more segments for a maintenance possession (see Table 7.3.1). The selected segments have a reduced number of available tracks. The maintenance possession scenarios are synthetic, they do not correspond to real-life cases. We selected the possessions in such a way that non-trivial computational or conceptual challenges appear in solving the NAHT nodes, in resolving the

capacity violations, or both. The possessions have a varying impact on the nominal timetable. The number of directly affected services ranges from 16 to as high as 97. The coefficients in the objective function are given in Table 7.3.2.

The practical value of the proposed algorithms is partly determined by the computation times. An efficient workflow necessitates calculations to be performed in a few minutes. Therefore, we limited the computation time of the NAHT model. Both the Direct Method (DM) and the Hot Start Method (HSM) have a time limit of 15 minutes in every iteration. We note that the MTU model is solvable to optimality for all stations in a total of about 20 seconds.

Table 7.3.4. Characteristics of the 9 test instances.

A	One segment (Mrn-Klp) reduced from 2 tracks to 0.
	Directly affects 16 services.
B	One segment (Rtz-Rlb) reduced from 4 tracks to 0.
	Directly affects 37 services.
C	One segment (Gdpa-Wp) reduced from 4 tracks to 0.
	Directly affects 41 services.
D	One segment (Asb-Ac) reduced from 4 tracks to 0.
	Directly affects 46 services.
E	One segment (Asdz-Shl) reduced from 4 tracks to 0.
	Directly affects 52 services.
F	One segment (Har-Klp) reduced from 2 tracks to 1.
	Directly affects 12 services.
G	One segment (Ah-Nm) reduced from 2 tracks to 1 track.
	Directly affects 22 services.
H	Three segments with reduced track capacity: Gvc-Ledn from 4 to 2, Bkl-Asb from 4 to 2, and Ed-Ah from 2 to 1.
	Directly affects 66 services.
I	Four segments with reduced track capacity: Ah-Nm from 2 to 1, Ehv-Btl from 4 to 2, Ut-Utvr from 8 to 6, and Utvr-Htn from 4 to 2.
	Directly affects 97 services.

Table 7.3.5: Penalty coefficients in the objective function of the NAHT model.

Cancellation of a freight service	1000
New starting event of a passenger service	100
Cancellation of a passenger service (per kilometre)	10
Forward time shift for passenger services (per event, per minute)	1

Backward time shift for passenger services (per event, per minute)	2
Forward time shift for freight services (per event, per minute)	0.01
Backward time shift for freight services (per event, per minute)	0.01

7.3.8.1 Tractability of the NAHT model

In our first tests we wonder to what extent the network-level NAHT model is solvable for large-scale instances. We apply both DM and HSM on test instances 'A' to 'I'. In these experiments, we do not attempt to resolve any track capacity violation.

The numerical results are presented in Table 7.3.3. DM solves instances 'A' to 'E' to optimality; we note that the actual solution time is between 10 and 20 seconds. This observation is in line with the intuition that possessions with fully blocked segments lead to easy optimisation problems. The resulting optimal solutions have very few time shifts or cancellations on top of the mandatory cancellation, indeed. Applying HSM does not have any benefit since HSM and DM find the same optimal solution.

Table 7.3.6. Comparison of the solutions obtained from DM and HSM. The first row of each instance indicates the total objective value as well as the contribution of passenger service cancellations, freight service cancellations and time shifts. The subsequent rows contain the proven lower bound on the objective value, the number of (fully or partially) cancelled services, the total geographic length of the cancelled passenger services, and the total amount of shift in minutes.

Instance	DM				HSM			
	Total	Pass.	Freight	Shifts	Total	Pass.	Freight	Shifts
A	3860 ≥3860	3860 #=16 266 km	0 #=0	0 0 min	3860 ≥3860	3860 #=16 266 km	0 #=0	0 0 min
B	7940 ≥7940	2940 #=32 174 km	5000 #=5	0 0 min	7940 ≥7940	2940 #=32 174 km	5000 #=5	0 0 min
C	12648 ≥12648	7648 #=36 485 km	5000 #=5	0 0 min	12648 ≥12648	7648 #=36 485 km	5000 #=5	0 0 min
D	21360 ≥21360	7360 #=32 416 km	14000 #=14	0 0 min	21360 ≥21360	7760 #=32 416 km	14000 #=14	0 0 min
E	6660 ≥6660	6660 #=52 506 km	0 #=0	0 0 min	6660 ≥6660	6660 #=52 506 km	0 #=0	0 0 min
F	756 ≥334	514 #=2 31 km	0 #=0	242 242 min	744 ≥214	514 #=2 31 km	0 #=0	230 163 min
G	181408	118756	61000	1652	2900	2900	0	0

	≥753	#=204 11796 km	#=61	1652 min	≥553	#=20 250 km	#=0	
H	18466 9 ≥1171	122244 #=204 12144 km	62000 #=62	425 425 min	6120 ≥827	4696 #=24 270 km	0 #=0	1424 1744 min
I	18339 4 ≥911	119252 #=204 11845 km	62000 #=62	2142 2142 min	4169 ≥668	3596 #=28 320 km	0 #=0	573 857 min

In contrast, instances ‘F’ to ‘I’ feature partially blocked segments, and they behave very differently. The commercial MILP solver fails to prove optimality for any of these instances. For test instance ‘F’, both DM and HSM find attractive solutions. With merely 12 affected services, instance ‘F’ is the easiest one of partially blocked instances.

The benefits of HSM are highlighted by instances ‘G’, ‘H’ and ’I’. HSM finds good solutions within the allotted 15 minutes of computation time. The solutions by DM do not have any practical value at all; more than two thirds of the services are cancelled.

We claim that the HSM solutions of ‘F’ to ‘I’ are of good quality from a practical point of view, even if the MILP solver is not able to find tight optimality gaps. Indeed, an informal assessment revealed that the amount of cancellation is in line with the traffic that the available infrastructure is likely to be capable of facilitating. The amount of time shifts is also acceptable. Instance ‘H’ has by far the highest amount of shift with its 1744 minutes. Out of the 9564 departure and arrival events in total, 8303 events are not shifted, and the remaining 1261 events have an average shift of 1.83 minutes, ranging between 0 and 5 minutes.

We note that, for the harder test instances ‘F’ to ‘I’, the solution quality of DM does not benefit from increasing the time limit from 15 minutes to 4 hours. The DM solutions are still worse (in some cases much worse) than the HSM solutions, except for test instance ‘F’ where the solution found by HSM is proven to be optimal.

7.3.8.2 Resolving track capacity violations

In the second group of our tests, we investigate the performance of the iterative solution approach. In particular, we are interested to see how many iterations are necessary to eliminate all track capacity violations. Moreover, we wonder at which costs (in terms of cancellation or time shift) the track capacity violations of the initial NAHT solution can be repaired. In these tests, the interactive feedback step was performed by the developers of the methods, after extensively discussing the problem setting with field experts.

We summarise the results in Table 7.3.4. It turns out that a handful of iterations are sufficient to resolve the capacity violations. Instance ‘G’ does not require any feedback action, since the first iteration’s NAHT solution has no capacity violation. The capacity violations in the other instances disappear after no more than 5 iterations. Table 7.3.4 also compares the objective value and some crucial characteristics of the solutions. The resolution of the capacity violation is achieved at a moderately higher objective value than that in the first iteration, and very little additional cancellation is used. As for time shifts, even the sharp increase in instance ‘E’ is acceptable in practice: 278 events (out of 9564) are shifted with an average of 3.9 minutes, and with shifts as high as 13 minutes.

Table 7.3.7. Comparison between the first iteration's solution (containing track capacity violation) and last iteration's solution (containing no track capacity violation). The first row of each instance indicates the total objective value as well as the contribution of passenger service cancellations, freight service cancellations and time shifts. Also, the number of feedback actions is shown. The subsequent rows contain the number of (fully or partially) cancelled services, the total geographic length of the cancelled passenger services, and the total amount of shift in minutes.

Instance	First iteration				Last iteration				#Feed-back
A	3860	3860 #=16 266 km	0 #=0	0 0 min	4303	4208 #=16 301 km	0 #=0	95 94 min	3
B	7940	2940 #=32 174 km	5000 #=5	0 0 min	9040	4040 #=28 324 km	5000 #=5	0 0 min	1
C	12648	7648 #=36 485 km	5000 #=5	0 0 min	13235	8112 #=36 651 km	5000 #=5	123 114 min	3
D	21360	7360 #=32 416 km	14000 #=14	0 0 min	21418	7360 #=32 416 km	14000 #=14	58 58 min	1
E	6660	6660 #=52 506 km	0 #=0	0 0 min	8852	7676 #=52 608 km	0 #=0	1176 1098 min	5
F	744	514 #=2 31 km	0 #=0	230 163 min	1097	1028 #=4 63 km	0 #=0	69 68 min	2
G	2900	2900 #=20 250 km	0 #=0	0 0 min	–	– – –	– –	– –	0
H	5805	4725 #=24 272 km	0 #=0	1080 1290 min	6607	5836 #=24 383 km	0 #=0	771 630 min	3
I	4459	4056 #=28 325 km	0 #=0	403 943 min	4436	3652 #=24 285 km	0 #=0	784 1094	1

7.3.9 Conclusions and future work

In this research we consider the problem of adjusting a nominal timetable due to preventive maintenance possessions. Our goal is to solve large and complex networks such as the Dutch railway system. Next to the macroscopic network view, we want to account for the limited track capacity of the stations.

We propose an optimisation model for solving the network-level timetabling problem as well as a station-level macroscopic track usage optimisation model that in essence evaluates the outcome

of the network-level model by counting the number of tracks. The network-level and station-level models are coupled by a feedback step in which human decision makers provide their expert judgement.

We carry out computational experiments in a test environment. Our tests use the real-life Dutch timetable and infrastructure data. We consider all passenger services of NSR as well as all freight services.

The proposed iterative approach turns out to be able to solve the considered network-level timetable optimisation problem combined with macroscopic station capacity limitations. Our approach consistently provides solutions with a good practical value. Both the solution time of the individual optimisation steps (limited at 15 minutes) and the small number of iterations are in the range that is desirable in the timetable design workflow.

The proposed approach is a good candidate to become the core of a decision support system. Our further work aims at helping this transition by inviting practitioners for an evaluation of our models on real-life problem instances, in order to refine models and to assess the true value of our methods in practice. These investigations are planned to be a part of Work Package 7.

The network-level timetable optimisation model is challenging to solve when the maintenance possessions block some segments partially. We are working on alternative mathematical formulations and solution methods to reduce the solution times and to find tighter optimality gaps.

The current feedback step from the station-level model to the network-level model is based on expert judgement. By including the experts in the solution process, we expect to enhance the practical value of the solutions. For example, the experts can choose services whose cancellations would cause the least harm to the passenger and freight flows. While the experts' insights are likely to be valuable in future extensions of our algorithm, we are looking for ways to automate the feedback, and to integrate some or all ideas of the station-level model into the network-level optimisation model.

We acknowledge that the practical feasibility of a timetable depends on mesoscopic or microscopic aspects such as the routes of the services inside the stations. We are going to investigate the link from macroscopic timetable optimisation to microscopic simulation, so that the simulation outcome can help the optimisation models obtain practically feasible solutions.

7.4 Optimized Short-Term Insertion of Additional Trains Into Existing Timetables

7.4.1 Background

Railway freight transportation, which consists in using railways and trains to transport cargoes, has increasingly been deemed as a great alternative to road transportation. Indeed, road congestion, high transportation costs and environmental concerns motivate the use of rail rather than road to transport goods. However, unlike railway passenger transportation which is usually predictable and for which timetables are established months to years in advance, railway freight transportation is more unpredictable. Indeed, requests for train paths could be formulated by freight Train Operation Companies (TOCs) to the Infrastructure Manager (IM) within a week from executing an existing timetable. To ensure the best possible utilization of the infrastructure, it is in the IM's best interest to quickly fulfill these requests by finding automated ways to address the challenge of inserting additional trains into existing timetables.

The current approach of the planners to deal with this problem consists in solving it manually in a sequential manner, separating local levels at the stations from broader levels within the routes. Although this approach provides feasible solutions in some cases, there are many cases where solutions cannot be found if the problem is not solved in a global and integrated manner, considering simultaneously routing and platforming problems. The purpose of this work is to study this problem and explore such solution approaches, with the aim of better supporting the planners' decisions.

The document is organized as follows. First, a literature review on residual capacity management is presented in Section 7.4.2. Then, an existing graph-based solution approach for residual capacity management is presented in Section 7.4.3. A formal definition of the studied problem is outlined in Section 7.4.4. Then, a resulting subproblem and an algorithm intended to solve the problem is introduced in Section 7.4.5. Computational tests and preliminary results of the proposed algorithm are presented in Section 7.4.6, followed by a conclusion and future work in Section 7.4.7.

7.4.2 Residual capacity management

As previously stated, the frequently unpredictable nature of railway freight transportation leads, for the IM, to the problem of short-term insertion of additional trains into existing timetables. The problem is described as follows. First, a TOC makes a freight train insertion request to the IM. This request is represented by a departure station and an arrival station (and optionally intermediate stations), in addition to time windows for the train at these two stations and the maximum travel time to complete its journey. Based on these inputs and information related to the infrastructure, the rolling stock and the existing timetables, the IM proposes to the TOC one or several train paths from which one must be chosen. These train paths must satisfy the usual safety and conflict avoidance constraints related to train timetabling.

Despite its relevance in improving the use of the railway infrastructure, this problem has received relatively limited attention in the literature. Various approaches have been proposed to solve variants of the problem, including Mixed Integer Programming (MIP) formulations solved using standard commercial solvers as in Jiang et al. (2014), Borndörfer et al. (2016), Gao et al. (2018) and Tan et al. (2020), metaheuristics as in Tan et al. (2020) or a Lagrangian relaxation heuristic as in Cacchiani et al. (2010). Other methods include adapted shortest-path algorithms performed on time-expanded graphs as in Flier et al. (2009), Haehn et al. (2020) and Ljunggren et al. (2021), in addition to approaches inspired from the resolution of the job-shop scheduling problem as presented in Burdett and Kozan (2009) and Tan et al. (2015). In almost all of these works, the problem is treated at the macroscopic level of infrastructure modeling, with applications on specific parts of a given railway network, usually a given railway line. This raises doubts regarding the level of applicability of these methods to solve the problem of inserting one or several trains in real-life railway timetabling.

Based on this conclusion, the French IM, *SNCF Réseau*, proposes to study the problem at a more detailed level of infrastructure modeling by exploring methods that are applicable not only to a single railway line, but also in complex networks. These methods are intended for the *Short-term Digital Capacity Management* (STDCM) module of *Open-source Railway Designer* (OSRD), a microscopic railway simulation tool currently being developed by the IM. By studying the problem directly at the microscopic level of infrastructure modelling for all types of networks, such methods are able to determine railway planning solutions by simulating timetables at a very detailed level. These solutions are more likely to be directly applicable in real-life. The resulting approaches, if successfully implemented, are more reliable than most of the existing macroscopic approaches that determine solutions that might not be valid, or at least reach the expected performances,

when implemented in the railway system. Details on the characteristics and functionalities of OSRD for timetabling, operational studies, infrastructure management and short-term capacity management can be found in the websites provided in Appendix F. In the rest of this work, a first solution approach proposed by Charpentier (2023) and implemented in OSRD is presented, along with an algorithm to improve its performance based on departure track reallocation.

7.4.3 A first approach to insert a single train

7.4.3.1 Time-expanded graph representation

Let us consider the simplified static graph representation of a reduced railway infrastructure shown in Figure 7.4.1, where the nodes model the geographical locations of all the infrastructure's components, and the edges model their connecting elements. As the infrastructure is represented at the microscopic level, the model combines information related to blocks, routes, track elements, signaling and speed, among other relevant factors. For example, node *A* could represent the geographical location of a switch in this part of the infrastructure, while nodes *F* and *D* could represent the positions of two buffer stops.

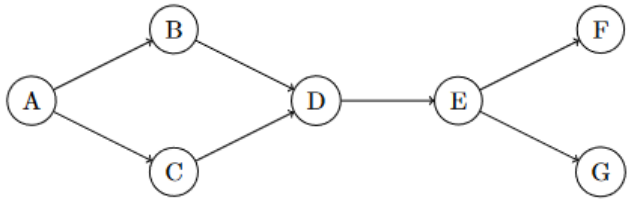


Figure 7.4.1: Simplified graph representation of a part of a railway infrastructure

When solving the problem using the approach described in Section 7.4.3.2, the graph is reconstructed at each second of the time horizon, taking into consideration the simulation results that indicate track availability in the infrastructure. This results in a time-expanded graph where edges represent the possibility to travel from one node to another between two consecutive time steps. A time-expanded graph constructed from the simplified infrastructure presented in Figure 7.4.1 is illustrated in Figure 7.4.2. In this example, we can see that from time step $T = 0$ to time step $T = 2$, the track sections between node *A* and nodes *B* and *C* are both available, allowing a train to use these sections during this time period. This possibility of movement is represented using the dashed edges that link these nodes. However, between time steps $T = 1$ and $T = 2$, we can observe that there is no edge connecting node *E* to node *G*, which indicates that the track section connecting the two nodes is now occupied after being free between time steps $T = 0$ and $T = 1$. In the next section, we present how this time-expanded graph is used to find an additional train path using a simulation-based path finding algorithm.

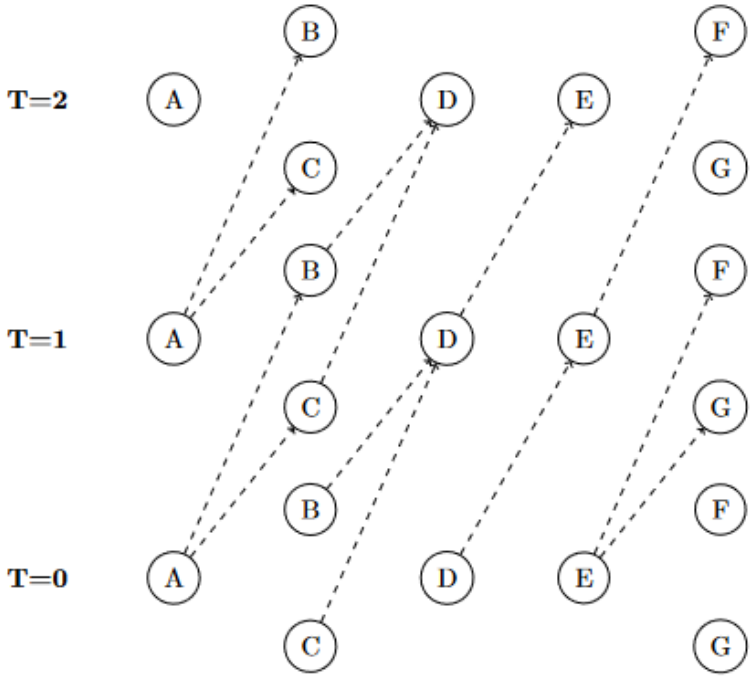


Figure 7.4.2: Time-expanded graph representation of the network

7.4.3.2 Simulation based A* search algorithm

Algorithm 7.4.1 outlines a first solution approach to solve the problem of inserting a single train into an existing timetable. It takes as inputs the infrastructure, the rolling stock, the existing timetable, user-defined constraints and various parameters. Based on these inputs, the algorithm searches for an additional train path in the infrastructure for the train we intend to insert, following six main steps detailed below.

- **Step 1.** First, a microscopic simulation is performed to determine the state of the infrastructure during the desired insertion period of the additional train. It is important to note that the margins (minimum time between two train paths) required before and after each train path are considered in the simulation. In this step, the inserted train is also included in the simulation. The aim is to explore all the potential paths, with different arrivals and departures to/from relevant operational points, that the train could use considering its maximum speed. This results in an initial time-expanded graph that is improved in Step 2.

- **Step 2.** This is a post-processing step intended to handle speed discontinuities. A speed discontinuity occurs when the speeds computed on the edge that leaves a node and enters another node are not feasible in practice. This results from the fact that, when the inserted train explores the infrastructure in Step 1, it does so at maximum speed one edge at a time, without visibility on the potential next edges. The purpose of this step is to adapt the speeds preceding a discontinuity accordingly. The result of these first two steps is a graph similar to the one illustrated in Figure 7.4.2.

- **Step 3.** Here, the A* search algorithm is performed on the resulting graph to find a path between the origin and destination nodes. Essentially, A* can be seen as an extension of Dijkstra's shortest path algorithm that uses heuristics to guide its search based on minimizing a cost function of the path that depends on both the total travel time and the departure time.

Note that the total travel time is estimated based on the geographical distance from the destination and the maximum speed of the train.

- **Step 4.** In this step, a conflict avoidance post-processing procedure is performed if needed. Indeed, in all the previous steps, it is possible to keep a path that is not initially feasible due to conflicts with the existing trains but that could become valid using departure time shifts for the inserted train at every operational point of the infrastructure.
- **Step 5.** Once a path has been found, the corresponding speed profile of the inserted train is computed, which characterizes the speeds at which the train should travel in all the track sections between its origin and its destination, while respecting all necessary safety constraints and speed limitations it encounters.
- **Step 6.** The algorithm either returns a feasible path or indicates that no path has been found for the insertion request. We note that, in the latter case, infeasibility is not proven. Indeed, this is a heuristic method; not finding a path does not necessarily mean that a path does not actually exist.

Algorithm 7.4.1: *Short-term Train Path finding Algorithm*

Require: Infrastructure, Rolling Stock, Constraints, Existing timetable
Ensure: Feasible Path or NULL
1: Simulation for infrastructure availability
2: Post-processing to fix speed discontinuities
3: Run A* search algorithm on resulting time-expanded graph
4: Post-processing step to unlock paths using departure time shifts
5: Compute speed profile for the resulting path
6: **return** Feasible Path or NULL

Implementing Algorithm 7.4.1's different steps involves choosing values for parameters that directly influence its performance in solving the problem. Although still subject to verification, it is expected that, for example, setting a higher maximum departure time shift or total travel time could yield more insertion solutions for the additional train. The same goes for the chosen simulation time step that could be aggregated to higher units of measure on the entire infrastructure, or carefully chosen parts of it. Other potential levers of optimization for the algorithm include minimal acceptable modifications to the existing timetable. One of these levers, which relies on track re-allocations in the existing timetable, is introduced in Section 7.4.5. In the next section, the general problem is formally defined.

7.4.4 General problem definition

Let us consider an existing conflict-free timetable *ET* for a set T of *n* trains. Each train $t_i \in$ T is defined by a departure station $S^d_{t_i}$, an arrival station $S^a_{t_i}$, a departure time $T^d_{t_i}$ and an arrival time $T^a_{t_i}$. For each train $t_i \in$ T, there is a set of intermediate stations S_{t_i}, and, for each station $s_j^{t_i} \in S_{t_i}$, a departure time $T^d_{s_j^{t_i}}$ and an arrival time $T^a_{s_j^{t_i}}$ are defined.

A request of inserting a train $\bar{t}$ into *ET* is defined by a departure station $S^d_{\bar{t}}$, an arrival station $S^a_{\bar{t}}$ and an optional set of intermediate stations $S_{\bar{t}}$ such that, for each station $s_i^{t_a} \in S_{\bar{t}}$, a departure time $T^d_{s_i^{\bar{t}}}$ and an arrival time $T^a_{s_i^{\bar{t}}}$ variables are defined. A maximum travel time $d^{max}_{\bar{t}}$ and a

departure time window $[T_{\bar{t}}^{d_{min}}, T_{\bar{t}}^{d_{max}}]$ are also defined such that $T_{\bar{t}}^{d_{min}}$ and $T_{\bar{t}}^{d_{max}}$ are respectively the minimum and maximum departure times from station $S_{\bar{t}}^{d}$.

Given these inputs, solving the problem consists in determining a new conflict-free timetable *NT* that includes train $\bar{t}$, with minimal to no changes in the existing timetable *ET*. This entails finding a departure time $T_{\bar{t}}^{d}$ of train $\bar{t}$ from $S_{\bar{t}}^{d}$ within the defined departure time window, a resulting arrival time $T_{\bar{t}}^{a}$ of $\bar{t}$ at $S_{\bar{t}}^{a}$ and, if included in the insertion request, the arrival times $T_{s_i^{\bar{t}}}^{a}$ and departure times $T_{s_i^{\bar{t}}}^{d}$ to and from the intermediate stations $\bar{t}$. Note that, although an imposed arrival time window for $\bar{t}$ is not considered, it is an operationally relevant parameter that could be included in future work.

Regarding the optimization criteria, we aim to find the additional train path for train $\bar{t}$ with: (1) The minimal total travel duration $d_{\bar{t}}$, and (2) The earliest departure time $T_{\bar{t}}^{d}$. The objective function to minimize could be formulated as $\alpha T_{\bar{t}}^{d} + \beta d_{\bar{t}}$, where α and β are weights reflecting the importance of each criterion. The definitions of decision variables and parameters of the problem are summarized in Table 7.4.1 and Table 7.4.2, respectively.

Table 7.4.1: Problem decision variables

Variables	Definition
$T_{\bar{t}}^{d}$	Departure time of inserted train $\bar{t}$
$T_{\bar{t}}^{a}$	Arrival time of inserted train $\bar{t}$
$T_{s_i^{\bar{t}}}^{a}$	Arrival time of inserted train $\bar{t}$ to intermediate station $s_i^{\bar{t}}$
$T_{s_i^{\bar{t}}}^{d}$	Departure time of inserted train $\bar{t}$ from intermediate station $s_i^{\bar{t}}$
$Tr_{t_i}^{d}$	Departure track of train t_i
$d_{\bar{t}}$	Total travel duration of inserted train $\bar{t}$
n_{tc}	Total number of track re-allocations

Table 7.4.2: Problem parameters

Parameters	Definition
ET	Existing timetable
NT	New timetable
T	Set of existing trains
$S_{t_i}^{d}$	Departure station of train t_i
$S_{t_i}^{a}$	Arrival station of train t_i
$T_{t_i}^{d}$	Departure time of train t_i
$T_{t_i}^{a}$	Arrival time of train t_i
S_{t_i}	Set of intermediate stations of train t_i
$T_{s_j^{t_i}}^{a}$	Arrival time of train t_i to intermediate station s_j
$T_{s_j^{t_i}}^{d}$	Departure time of train t_i from intermediate station s_j
$\bar{t}$	Additional train to be inserted
$S_{\bar{t}}^{d}$	Departure station of inserted train $\bar{t}$
$S_{\bar{t}}^{a}$	Arrival station of inserted train $\bar{t}$
$S_{\bar{t}}$	Set of intermediate stations of inserted train $\bar{t}$
$[T_{\bar{t}}^{d_{min}}, T_{\bar{t}}^{d_{max}}]$	Departure time window of inserted train $\bar{t}$

$d_{\bar{t}}^{max}$	Maximum travel time requested for inserted train $\bar{t}$
$AT_{t_i}^{d}$	Set of alternative departure tracks of train t_i
α, β	Weights in the objective function

Solving this general problem using the approach described in Section 7.4.3 may lead to a new track reallocation subproblem. The latter is described in the next section, as well as a first solution approach.

7.4.5 Unlocking solutions using track reallocation

7.4.5.1 General principle

As mentioned in Section 7.4.2, the outlined solution approach has been implemented and tested in OSRD. While the approach is able to provide feasible solutions in relatively small computing times,
only the current state of the parameters given as inputs is taken into account. However, more solutions can be found by applying minor acceptable changes to these inputs. This is particularly useful for unlocking situations where the algorithm cannot find a feasible solution.
Multiple levers could be explored, one of them being the reallocation of station tracks. Indeed, by observing the simulation results of some train insertion attempts, it became apparent that the path finding algorithm could not be completed because of potential conflicts with other existing trains inside or near stations. Hence, reorganizing track occupancy within these bottleneck stations could help create space for an additional train path, returning solutions that otherwise would have been overlooked. In the following sections, we first extend our initial problem definition by describing our new track reallocation subproblem in Section 7.4.5.2. As we only focus, for now, on the departure tracks of the existing trains, we describe a first track reallocation algorithm in Section 7.4.5.3 that aims at retrieving solutions overlooked by the algorithm presented in Section 7.4.3.2.

7.4.5.2 Subproblem description

Exploring the possibility of track reallocation requires us to define a new subproblem as follows. Let $Tr_{t_i}^{d}$ be the decision variable that defines, for each train $t_i \in$ T, its departure track from its departure station $S_{t_i}^{d}$. Let f = min n_{tc} be the objective function of the subproblem, where n_{tc} is the overall number of re-allocations resulting from a successful track reallocation procedure. We also define, for each train $t_i \in$ T, a set of alternative departure tracks $AT_{t_i}^{d}$ from which a new track can be allocated to train t_i, if necessary.
Solving the departure track reallocation subproblem requires finding the new track occupancy in the departure stations that solve the general problem of finding the additional train path for train $\bar{t}$, with the minimal total travel duration $d_{\bar{t}}$ and earliest departure time $T_{\bar{t}}^{d}$, as defined in Section 7.4.4, while minimizing n_{tc}. In the next section, we present a first solution approach to solve the track reallocation subproblem, with a focus on feasibility over solution quality.

7.4.5.3 A track reallocation algorithm

To explore the possibility of track reallocation, we propose a first algorithm, detailed in Algorithm 7.4.2, that reallocates tracks in the departure stations of the existing timetable. The algorithm is divided into three main blocks:

- **Block 1** (lines 1--9)**.** In this initial block, the initial short-term path finding algorithm is run for the additional train while considering the existing timetable. If a path is found, it is returned

without the need to perform any track re-allocations. Otherwise, a list of all existing trains that could potentially be in conflict with the additional train is defined. These are the trains that are using the same part of the infrastructure during the same period of time as the added train, considering its departure time window and maximum travel duration.

- **Block 2** (lines 10--15)**.** This block handles the track reallocation process. While no short-term path with no conflicts in the new timetable is found, track re-allocations are performed for the trains in the list constructed in Block 1 by choosing a new departure track from their respective lists of alternative departure tracks. Each time a new departure track is selected, the initial short-term path finding algorithm is run. If successful, the third block of the algorithm is entered.

- **Block 3** (lines 16--28). This is the verification block. Its purpose is to check if the new timetable, resulting from adding the newly found additional train path, contains no conflicts. Conflict detection is performed on the entire timetable to check for conflicts not only between the added train and the existing ones, but also between the existing trains, in case of a track reallocation. If the new timetable is valid, it is returned and the while loop initiated in block 2 is exited. Otherwise, the next alternative departure track for the selected train is considered, or the next train if all track re-allocations have been tried. If all possible track re-allocations for all the selected trains are not successful, the algorithm returns that no feasible path has been found for the additional train. Infeasibility is still not proven here, as not finding a path does not necessarily mean that a solution does not exist.

This first algorithm is proposed with the intention of testing if track reallocation is an efficient lever to improve the performance of the STDCM module in OSRD. It is important to note that, when changing the departure track of a train, OSRD recomputes a new path for the train that may differ from the initial path. As these tools are mainly provided for decision support, it is ultimately the planners who decide whether the modifications in the existing timetable are acceptable or not in order to approve the insertion of the additional train.
Additionally, this track reallocation lever should be generalized to all stations, and not only the departure stations. The final goal is to improve the algorithm so that it detects the bottleneck stations and performs track reallocation to unlock additional paths for the inserted train. Also, note that the current brute-force approach is likely to face scaling challenges and more optimized ways of performing track reallocation must be considered in the algorithm. Also, other additional relevant criteria such as the overall number of changes suggested in Section 7.4.5.2 could be optimized.

Algorithm 7.4.2: Track Reallocation for Short-term Train Insertion

Algorithm 2: Track Reallocation for Short-term Train Insertion V0

```
Data: Infrastructure, Conflict-free Timetable, Insertion request
Result: Conflict-free Timetable with inserted train
Run STDCM algorithm
if Feasible Path Found then
    return Feasible Path
else
    Short-term path ← False
    Conflict Free TT ← False
    foreach Train ∈ Existing Timetable do
        if Potential_Conflict(Train, Added Train) is True then
            Append Train to Reallocate_Trains_List
    while (Short-term path is False) & (Checked trains < len(Reallocate_Trains_List) ) do
        Train ← Train from Reallocate_Trains_List
        Append possible Departure Tracks of Train to Track_List
        while (Conflict Free TT is False) & (Checked Tracks < len(Track_List)) do
            Departure_Track_of_Train ← Track from Track_List
            Run STDCM algorithm
            if Feasible Path Found then
                New Timetable ← Existing Timetable + Added Train
                Conflict Free TT ← ConflictDetection (New Timetable)
                if Conflict Free TT is True then
                    Short-term path ← True
                else
                    Increment Checked Tracks
            else
                Increment Checked Tracks
        Increment Checked Trains
    if Short-term path is True then
        return New Timetable
    else
        return No feasible path found
```

7.4.6 Computational experiments

7.4.6.1 Experimental setup

The algorithm proposed in Section 7.4.5.3 has been implemented and tested to validate the efficiency of track reallocations in improving the performances of the initial algorithm. The tests were performed considering the toy infrastructure *small_infra* presented in Figure 7.4.1. It is one of the fictional test infrastructures available on OSRD and is composed of 8 stations, each one having between 1 and 4 station tracks. We note that the infrastructure is represented at the microscopic level, which considers the most precise level of modelling detail. To test our algorithm, we run experiments considering the same unsolved short-term insertion request for a train departing from the South-East station to the North-West station for three different scenarios described below.

- **Scenario 1.** In this scenario, we consider an existing conflict-free timetable of 23 trains with various departure and arrival stations, running on the infrastructure between 08:08:29 and 11:00:17. The short-term insertion request is defined by a departure at 08:00:00 with a 2-hour time window. The purpose of this first scenario is to perform a general test of the algorithm and see if it can unlock solutions for the initially unsolved insertion request.

- **Scenario 2.** In this scenario, we consider the same existing timetable considered in Scenario 1, but with a departure time for the inserted train at 9:30:00 and a time window of 1 hour. The purpose of this scenario and this “tighter” insertion request is to see if solutions could be found in the second part of the existing timetable.

- **Scenario 3.** This scenario considers a different existing timetable of 26 trains running on *small_infra* between 08:57:50 and 10:39:37 and a departure time for the inserted train at 8:55:00 with a time window of 1 hour. The purpose of this scenario is to show the importance of adding a bottleneck station detection block to the proposed track reallocation algorithm.

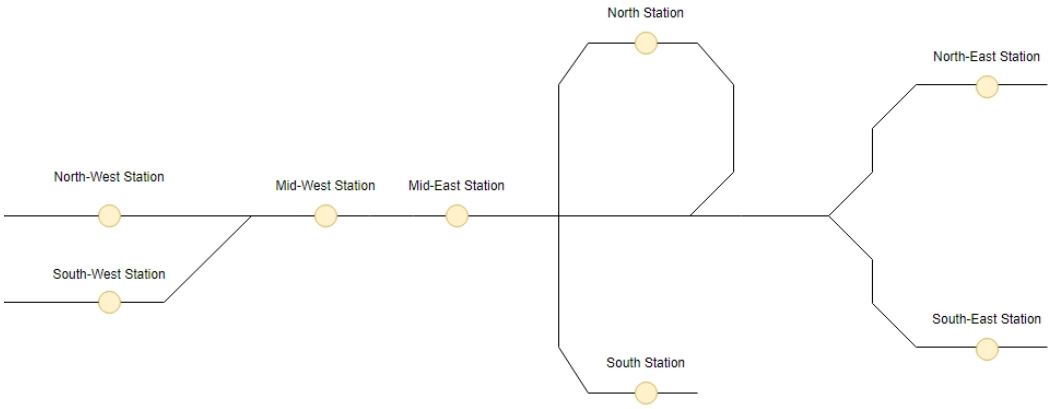


Figure 7.4.1: Test infrastructure small_infra

7.4.6.2 Preliminary results

Running the implemented track reallocation algorithm results in proposing short-term train paths for the insertion requests considered for all the three scenarios described in Section 7.4.6.1. For Scenario 1, a train path with a departure at 9:01:00 and an arrival at 9:15:00 is found by changing the departure track of a train departing from the North-West station at 09:09:24 and arriving to the North-East station at 09:22:08. As mentioned in Section 7.4.5.3, changing the departure track of a train may induce changes in its itinerary. The solution provided in Scenario 1 results in a total of 4 changes in the tracks used by the existing train.

In Scenario 2, a different train path is provided after running the track reallocation algorithm. In this solution, the inserted train departs at 9:57:00 and arrives at 10:11:00. This solution results from changing the departure track of a train, different than the one affected in Scenario 1, departing from the North-West station at 10:12:21 and arriving at the South-East station at 10:24:27. Also, the solution requires only one track change in the affected train's itinerary; the departure station track. This result highlights the importance of including the number of track re-allocations as an optimization criterion to minimize in the proposed track reallocation algorithm, as the currently implemented version blindly returns the first feasible solution it finds.

In Scenario 3, a train path for the additional train with a departure at 9:22:00 and arrival at 09:35:00 is found by changing the departure track of a train departing this time from the Mid-West station at 9:30:00 and arriving at the North-East station at 09:40:16. This solution results in only one change in the affected train's itinerary. Unlike the first two scenarios, the track reallocation that results in unlocking a train path for the additional train is performed in the Mid-West station. This highlights the importance of including a bottleneck station detection block in the track reallocation algorithm, to improve its performance by directly targeting these stations.

The main takeaways from these preliminary results can be summarized as follows. First, track reallocation appears to be an efficient lever that is worth exploring and developing for unlocking more solutions of the studied problem. Second, the current version of the algorithm should be improved by considering optimization criteria, starting with the proposed criterion of minimizing number of acceptable track re-allocations in the original timetable. Finally, the algorithm should include a bottleneck station detection block that would help in better targeting the stations where

track reallocation should be performed.

7.4.7 Conclusion and future work

In this work, we studied the problem of short-term scheduling of additional trains into existing timetables. The problem was defined and positioned in the literature, and a graph-based solution approach developed by SNCF Réseau for its microscopic railway simulation tool OSRD was presented. As the method may overlook solutions that could result from acceptable changes in the inputs, we proposed to explore the reallocation of tracks in the existing timetable as a potential lever to unlock more train paths. A first algorithm relying on departure track reallocations was proposed to test the efficiency of this lever. Preliminary experimental results showed that the track reallocation lever is worth exploring, and provided insights on how the current version of the algorithm could be developed and improved.
Future work, which will be conducted in Work Package 7, includes adding a bottleneck station detection component to the algorithm and considering optimization criteria such as the proposed minimum number of track re-allocation. Also, the problem of inserting multiple additional trains will be studied, in addition to the insertion of a single train in case of a timetable disruption.

7.5 A new spatial decomposition for (Short-term) Train Timetabling

7.5.1 Background

The long-term train timetabling (LTT) problem is to design a (periodic) timetable for a set of trains that operate in a given railway network for a time period that goes from one to several years. The short-term train timetabling (STT) concerns with the problem of “adjusting" a given timetable - for a time period that goes from next day up to the next year - usually in case of some variation in the set of trains or in the railway network. In this case, the typical applications refer to the insertion of a new train, the cancellation of a scheduled train, or the (temporary) unavailability of some resource in the railway network. Beside the objective function and some issues related to the fact that in the STT case some decisions pertaining the trains may be considered as fixed, most of the more important features that characterize both LTT and STT problems are shared by the two problems. Therefore, in the following we will denote as TT a problem that can be either an LTT problem or an STT problem.

In fact, when solving a TT problem, the decisions to be taken pertain to two main aspects:

- routing issues, i.e. finding a route that takes each train from its origin to its destination.
- conflict issues, i.e. resolving the precedence relations on each couple of trains that want to use the same resource.

Once such a decision stage has been completed, then a scheduling stage follows, where train operations are scheduled over time. Clearly, the two phases are not independent one from the other. Indeed, the train scheduling is defined according to the decisions taken in the decision stage. On the other hand, the quality of the solution of the decision stage is measured in terms of the quality of the schedule that can be derived from it.

It is well known that solving the TT problem as a whole is very difficult and it is often the case that even relatively small instances cannot be solved in a computational time that is compatible with any real time application.

The contribution provided by this sub-task is the definition of new methodological tools for the solution of TT problems that, in particular, we think could be effectively used for solving large scale STT problems. In particular, we will introduce a decomposition scheme based on the concepts of lines and stations, later referred to as LS-ALG. This algorithm represents a rather natural approach for solving TT problems, where trains move alternatively traversing line tracks (that mainly define the routes of the trains on a macro scale, while only few conflicts show up) and stations (where most of the train meetings and conflicts need to be synchronized, while routing decisions only affect local detours). We describe in Section 7.5.2 the LS-ALG procedure, giving detail on the overall algorithm as well as on the mathematical models behind the idea.

The computational effectiveness of a line-station decomposition approach for the rail scheduling problem strongly relies on the balance between the computational effort one has to spend in solving the line (master) problem and the one needed for solving the station (slave) problems. In fact, on the one hand, having large (and/or complex and/or congested) stations implies spending more computational time to find good slave solutions but also having simpler line problems to be solved. Then, in Section 7.5.3, we discuss the problem, addressed as the *Railway Minimum Clustering Problem (RMCP)*, of decomposing the original network into lines and stations in a suitable way that could possibly benefit the LS algorithm. In particular, we define an Integer Linear Programming formulation to solve RMCP to optimality, as well as a heuristic procedure.

7.5.2 The LS algorithm

Following other successful examples in the literature (see Lamorgese and Mannino (2015)), the basic idea of the LS algorithm is to decompose the resources of the railway network into station resources (tracks, platform tracks, switches and all resources attaining the so call station movements of the trains - see Borndoerfer and Schlecte (2007) and Caprara et al. (2011)) and line resources (roughly defined as the tracks between two consecutive stations). Then, the procedure consists of a master-slave iterative scheme, where the master problem concerns routing (and scheduling) the trains on the line resources. At each iteration *i*, a (possibly optimal) solution to the master problem is calculated. By solving a timetabling problem on each station of the network, we can check if such solution can be locally extended at the station level. Each station problem that results infeasible according to the current master solution produces a “no good" constraint that *locally* cuts off such solution. Otherwise, if a feasible (optimal) solution $\mathbf{x}_n^i$ to the station problem n is found, we refine the master problem by adding a set of linear constraints on the line resources that are implied by applying $\mathbf{x}_n^i$ at the trains that pass through station n.

7.5.2.1 The master problem

Let the *master graph* $G_L = (N_L, E_L)$ (see Figure 7.5.1 (a)) model the line-station decomposition defined above. In particular, the stations of the railway network are associated with the nodes N_L, while the edges E_L define the line structure of the network (i.e. the tracks that connects the stations of the network). Observe that we could have several parallel edges between two adjacent stations. In this case, each of such edges represents a single track between an exit point of the first station and the entry point of the second one. We denote by $\delta(n)$ the set of tracks that are incident to any station $n \in N_L$.

Let T be the train set. Then, for each train $t \in T$, the solution of the master problem defines a path P_t on G_L from its origin to its destination. We denote by $\mathbf{z}$ the binary variables that define such paths (i.e., $z_t(e) = 1$ if $e \in P_t$, for each $e \in E_L$ and $t \in T$). Moreover, for each edge e of E_L, the master solution provides a time window $[\alpha_e^t, \beta_e^t]$, where α_e^t is the time at which t enters line e and β_e^t is the time at which t frees line e. Since they are feasible for the master problem, such values satisfy, for each $t \in T$:

- the line crossing constraints: for each $e \in P_t$, $\beta_e^t \geq \alpha_e^t + p_e^t$, where p_e^t, is the minimum time needed by t to cross e;
- the line conflict constraints: for each $t' \in T \backslash \{t\}$ and each $e \in P_t \cap P_{t'}$, we have that $(\alpha_e^t, \beta_e^t) \cap (\alpha_e^{t'}, \beta_e^{t'}) = \emptyset$;
- the (relaxed) station crossing constraints: for each e, f consecutive in P_t, $\alpha_f^t \geq \beta_e^t + q_n^t$, where q_n^t, is the minimum time needed by t to cross the station n between e and f.

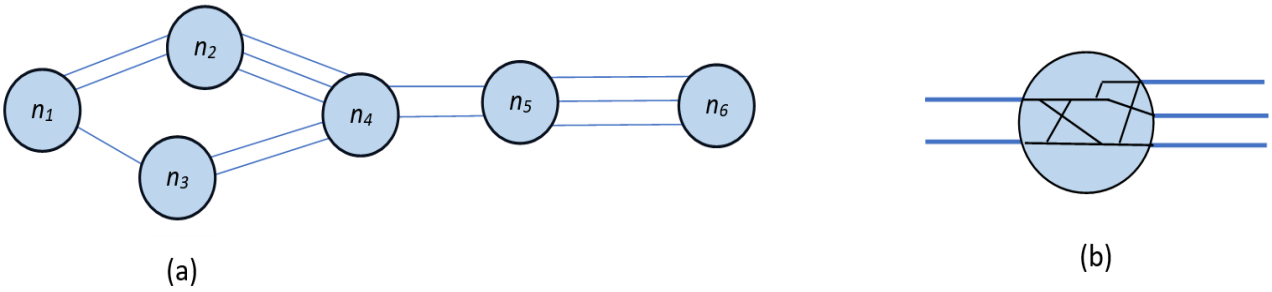


Figure 7.5.1. In (a), an example of a master graph $G_L = (N_L, E_L)$. In (b), a station graph $G_n = (N_n, E_n)$ with its internal tracks.

As for constraints 2., for the sake of simplicity in the exposition, we consider that only one train can occupy a single line at any time. Though, we can very easily adapt the model to allow multiple follower trains (suitably separated by a given headway amount of time) to use different track segments of the same line at the same time.

We can assume that the objective function of the master problem can be obtained as the projection of the objective function of original TT problem into the resources of the master graph.

In the following, we denote by $\mathbf{y}$ the precedence binary variables that are implied by the $[\alpha_e^t, \beta_e^t]$ intervals of the master solution. In particular, for each ordered couple $t, t' \in T$ and each $e \in P_t \cap P_{t'}$, we let $y_{t,t'}(e) = 1$ if train t uses resource e before train t' (i.e. $\alpha_e^t < \alpha_e^{t'}$). Now, for each iteration i of the LS algorithm, let $(\mathbf{z}^i, \mathbf{y}^i)$ be the route and precedence variables of the solution to the master problem. For each station $n \in N_L$, such solution induces:

- A set of trains $T_n \subseteq T$ that pass through station n;
- An assignment $A(\mathbf{z}_n^i)$ that indicate, for each train $t \in T_n$, an entering track $e_n^t \in \delta(n)$ and an exit track $b_n^t \in \delta(n)$;
- A partial order $O(\mathbf{y}_n^i)$ that defines the precedence of the trains of T_n on the tracks of $\delta(n)$.

7.5.2.2 The station problem

As already mentioned, at each iteration i of the LS algorithm, we try to extend the solution of the master problem to each station $n \in N_L$. The station problem for station n is therefore to find an optimal schedule for the trains of T_n on the resources of n (modeled by a station graph $G_n = (N_n, E_n)$, as Fig. 7.5.1 (b)) that satisfies both the assignment $A(\mathbf{z}^i)$ and the partial order $O(\mathbf{y}^i)$. Here, the objective is to obtain a feasible solution that, for each train $t \in T_n$, induces a time window for the occupancy of each assigned resource $e \in \delta(n)$ that minimizes the deviation from the values $[\alpha_e^t, \beta_e^t]$ defined by the solution of the master problem. Solving the station problem has two possible alternative outcomes:

- The problem is infeasible. In this case, we add the master problem a “no good" constraint that cuts off the current solution that induces $A(\mathbf{z}^i)$ and $O(\mathbf{y}^i)$.
- The problem admits a feasible (optimal) solution $\mathbf{x}_n^i$. Applying this solution to the master problem implies to satisfy a finite set of constraints $D_n^i(\alpha,\beta) \geq d_n^i$ on the occupancy time windows of the trains of T_n on the tracks of $\delta(n)$. Hence, we proceed as follows:
  - We add the master problem a binary variable w_n^i that gets activated (is forced by linear constraints to assume value 1 if both the assignment $A(\mathbf{z}^i)$ and the partial order $P(\mathbf{y}^i)$ are satisfied by the current master problem solution.
  - We add the master problem the set of constraints $D_n^i(\alpha,\beta) \geq d_n^i - M(1-w_n^i)$. Observe that, being M a “big enough" constant value, the constraints $D_n^i(\alpha,\beta) \geq d_n^i$ result to be activated by the activation of variable w_n^i.

7.5.3 Defining the extended stations

As already mentioned in the Introduction section, the computational effectiveness of a line-station decomposition approach strongly depends on the balance between the computational effort one has to spend in solving the line (master) problem and the one needed for solving the station (slave) problems. In fact, on the one hand, having large (and/or complex and/or congested) stations implies spending more computational time to find good slave solutions but also having a simpler line problem to be solved. On the other hand, if stations are smaller (simpler, with less resources, etc.), one can expect to have a more difficult master problem to solve while easily producing good station solutions. As a consequence, it could be very useful to have at hand an algorithmic tool that, according to the different methods used to solve the two problems and the current status of the network (number and position of the trains, congestion rate, presence of unavailable resources, etc.), can dynamically find a partition of the resources into lines and stations of suitable size. Indeed, stations can be simply seen as connected portions of the railway network. In the following, we present an Integer Linear Problem for solving the problem to exact optimality as well as a heuristic procedure. It is worth remarking again that solving such a problem represents an ancillary aspect of an effective implementation of the LS algorithm. Therefore, the quality of the solutions provided by the exact and the heuristic approaches we present here should be eventually measured in terms of the computation benefit they can provide to the LS method.

7.5.3.1 Definition of the problem

In the following, we assume we are given a railway network modeled by a graph $G = (S, L)$. The nodes S represent the stations of the network, while the edges L are associated with the track lines. As we can have multiple tracks among the same couple of station, it turns out that L is actually a multiset (see Figure 7.5.2 for an example). We say that $e \subseteq S$ is an *extended station* if it induces a connected component of G.

Then, let $\mathcal{S}$ be a partition of S into extended stations. We define the *extended graph* $G_E = (\mathcal{S}, \mathcal{L})$. Here, we have an edge $\{e_u, e_v\} \in \mathcal{L}$ for any $\{u, v\} \in L$, with $u \in e_u$ and $v \in e_v$ (therefore, also $\mathcal{L}$ is a multiset). See Figure 7.5.3. We associate each station $s \in S$ with a measure $q(s)$ of its complexity and assume that the complexity $Q(e)$ of an extended station e is the sum of the complexities of its stations (i.e. $Q(e) = \sum_{s \in e} q(s)$). Moreover, let $\overline{Q}$ be a given maximum complexity for an extended station. Similarly, let Δ be the given maximum number of tracks that can be incident to any extended station (Δ represents a measure of the maximum complexity of linking a station to the rest of the network). Then we say that the extended graph $G_E = (\mathcal{S}, \mathcal{L})$ is *feasible* if, for each extended station $e \in \mathcal{S}$:

- The complexity of e does not exceed $\overline{Q}$;
- the degree of e does not exceed Δ.

Then, given $G = (S, L)$, Q, and Δ, the Railway Minimum Cluster Problem (RMCP) is to find a minimum size partition $\mathcal{S}$ such that $G_{\mathcal{S}} = (\mathcal{S}, \mathcal{L})$ is feasible. We defined and implemented an integer linear programming formulation for MSCP that is capable to solve in reasonable computing times instances of realistic size instances (Section 7.5.3.2). A heuristic procedure is also presented in Section 7.5.3.3. Computational experiments are still in progress.

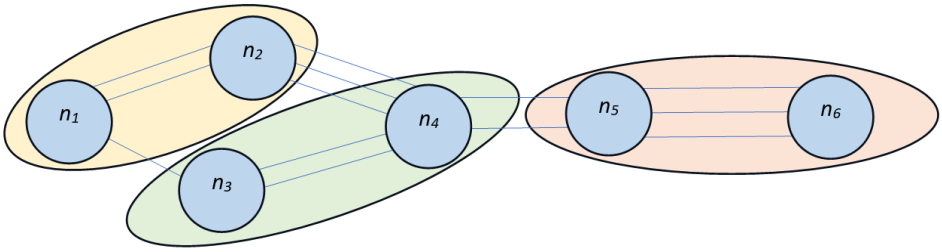


Figure 7.5.2. A graph $G = (S, L)$ that represents a railway network with station $S = \{s_1,\ldots,s_6\}$.

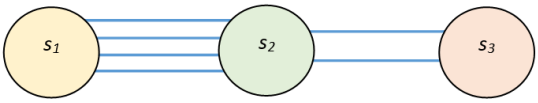


Figure 7.5.3. The extended graph $G_E = (\mathcal{S}, \mathcal{L})$ defined from G by the partition $\mathcal{S}$ of the stations set, being $\mathcal{S} = \{e_1 = \{s_1, s_2\}, e_2 = \{s_3, s_4\}, e_3 = \{s_5, s_6\}\}$.

7.5.3.2 An Integer Linear Formulation for RMCP

In the following, we assume we are given a railway network represented by the graph $G = (S, L)$, a complexity $q(s)$ for each $s \in S$, a maximum value $\overline{Q}$ for the complexity of an extended station, and a maximum value Δ for the number of edges that be incident to any extended station. Again, RMSCP is to define a partition $\mathcal{S}$ of minimum size such that the extended graph $G_E = (\mathcal{S}, \mathcal{L})$ is feasible. Therefore, we use the following set of binary variables.

- for each station $s \in S$ and each cluster $e \in \mathcal{S}$,
  - $x_{se} = 1$ if e includes s
- for each edge $\{u, v\} = \ell \in L$ and each cluster $e \in \mathcal{S}$,
  - $w_{\ell e} = 1$ if cluster e contains both u and v;
  - $y_{\ell e} = 1$ if cluster e contains u but not v (i.e. $x_{ue} + x_{ve} = 1$);
- for each cluster $e \in \mathcal{S}$
  - $z_e = 1$ if cluster e is not empty.

Then we define the following constraints.

Since every station is assigned to a cluster, we have that

$$\sum_{e \in \mathcal{S}} x_{se} = 1, \qquad s \in S$$

The complexity of an extended station cannot exceed Q.

$$\sum_{s \in S} q(s)\, x_{se} \leq \overline{Q}, \qquad e \in \mathcal{S}$$

The connectivity of an extended station cannot exceed Δ

$$\sum_{\ell \in L} y_{\ell e} \leq \Delta, \qquad e \in \mathcal{S}$$

Variables y must be *activated*

$$y_{\ell e} \geq x_{ue} - x_{ve}, \qquad \ell = \{u, v\} \in L, e \in \mathcal{S}$$

Variables w_{le} can assume value 1 only if both the extremes of l belong to the extended station e

$$w_{\ell e} \leq x_{ue}, \quad w_{\ell e} \leq x_{ve} \qquad \ell = \{u, v\} \in L, e \in \mathcal{S}$$

Variables z must be *activated*

$$z_e \geq x_{se}, \qquad s \in S, e \in \mathcal{S}$$

Each extended station must be a connected component of the graph G. This implies that, if both $u, v \in S$ belong to the cluster e, then, for each cut $\delta(S)$ separating u from v in G, at least one edge l must belong to e.. Therefore, we have the following inequalities

$$\sum_{\ell \in \delta(U)} w_{\ell e} \geq x_{ue} + x_{ve} - 1, \qquad e \in \mathcal{S}, u, v \in S, U \subset S \text{ with } u \in U, v \notin U$$

All variables are binary

$$\begin{aligned} x_{se} &\in \{0,1\}, && s \in S, e \in \mathcal{S} \\ y_{\ell e} &\in \{0,1\}, && \ell \in L, e \in \mathcal{S} \\ w_{\ell e} &\in \{0,1\}, && \ell \in L, e \in \mathcal{S} \\ z_e &\in \{0,1\}, && e \in \mathcal{S} \end{aligned}$$

Finally, the objective function is to minimize the number of activated (i.e. non empty) extended stations

$$\min \sum_{e \in \mathcal{S}} z_e$$

7.5.3.3 A heuristic algorithm for RMCP

We also defined and implemented an iterative heuristic procedure for the Railway Minimum Cluster Problem that works as follows

- **set** $\overline{S} = S$
- **set** $k = 1$
- **while** $\overline{S} \neq \emptyset$
  - **set** $C = \{u\}$ with $u \in \overline{S}$

- **set** $\overline{S} = \overline{S} \backslash u$
- **set** $Q(\overline{S}) = q(u)$
- **while** there exists $v \in \overline{S}$ such that: v is adjacent to any node in C and $Q(\overline{S}) + q(v) \leq \overline{Q}$
  - **set** $C = C \cup \{v\}$
  - **set** $\overline{S} = \overline{S} \backslash v$
  - **set** $Q(\overline{S}) = Q(\overline{S}) + q(v)$
- **while** $\delta(C) > \Delta$
  - in the opposite order with respect to the ones that have been inserted in C, remove nodes from C and re-push them into $\overline{S}$
- **assign** all nodes in C to cluster k
- **set** $k = k + 1$

All along the algorithm, $\overline{S}$ is the set of stations that still have to be assigned to some cluster. Then, at each iteration of the algorithm, we: i) pick a station u from $\overline{S}$ and assign it to a new cluster C (steps 4., 5., 6.); ii) starting from u, iteratively expand C as much as possible, while maintaining the connectivity of the corresponding sub-graph and satisfying the complexity constraint (steps 7., ..., 19); iii) iteratively remove boundary nodes from C until the connectivity constraint is satisfied (steps 11, 12.). Observe that, since we assume $\delta(s) \leq \Delta$ for each $s \in S$, then, at the end of the while loop of step 11., the current set C contains at least one element.

7.5.3.4 Computational experiments

In this subsection we present the results of a very preliminary set of computational experiments we performed in order to evaluate the performance of both the exact algorithm based on the binary linear formulation defined in Section 7.5.3.2 as well as of the heuristic procedure illustrated in Section 7.5.3.3. In particular, we solved the binary formulation using XPress v.v9.3.1. In particular, we dynamically generated the inequalities the enforce the connectivity of the clusters within a separation callback triggered at every incumbent integer solution of the branch-&-bound tree. Both procedures have been implemented in python and the code ran on a 12th Gen Intel(R) Core(TM) i5-1245U 1.60 GHz machine with 32 Gb of ram. The instances have been generated slightly perturbing (we maintained the number of stations, lines, and resources) the rail network of a Class 1 US company. For all instances, we set $\Delta = max_{u \in S} \delta(u)$ and $\overline{Q} = max_{u \in S} q(u)$. In Table 1, we report the results of such computational experiments. In particular, for each instance, column $Stations$ indicate the number of stations of the instance, column z^* the value of the optimal solution, column $\overline{z}$ the value returned by the heuristic solution defined in Section 7.5.3.3 (and passed as warm-up solution to the ILP solver), while in column CPU Time, we report the computational time (in seconds) needed to solve the instance to proven optimality (the computational times required by the heuristic algorithm turned out to be negligible for all instances).

Table 1. The results of the computational experiments for the exact and the heuristic procedure for the Railway Minimum Clustering Problem

Instance	Stations	z^*	$\overline{z}$	CPU Time
Instance 1	43	2	3	0.18
Instance 2	73	9	10	1.48
Instance 3	119	8	9	20.03
Instance 4	59	5	5	0.05
Instance 5	73	9	10	1.48
Instance 6	170	11	12	168.36

7.6 A genetic algorithm for decision support in timetable optimisation

7.6.1 Background

Efficient railway operations are fundamental to modern transportation systems, particularly in densely populated regions where railways play a crucial role in reducing congestion and providing sustainable alternatives to road transport. Timetable optimization is a central challenge for railway management, especially when dealing with large, interconnected networks. A timetable is not only a static schedule but must also adapt to various real-time events and disruptions that arise during operations.

Conflicts in railway timetables occur when two or more trains require access to the same segment of track or other resources at overlapping times, leading to potential delays or disruptions. Additionally, conflicts can arise from infrastructure limitations, such as planned track maintenance or station closures. These conflicts must be resolved in a timely and efficient manner to maintain smooth operations, minimize delays, and ensure passenger satisfaction.

Traditionally, timetable adjustments and conflict resolution have required significant manual intervention from planners. However, advancements in optimization algorithms, particularly those leveraging computational techniques such as mathematical programming and genetic algorithms, have paved the way for automatic timetable optimization. These approaches have been explored extensively in recent research. Alfieri et al. (2006) demonstrated the use of mathematical programming to address train scheduling problems, focusing on minimizing delays through efficient allocation of resources. Similarly, Cacchiani et al. (2010, 2013) applied heuristic approaches to solve real-world train unit assignment issues, achieving promising results in both short- and long-term planning scenarios.

The method proposed in this paper extends the literature by addressing timetable conflicts through a fully automated process that resolves conflicts without requiring user intervention. Our approach leverages a combination of predefined resolution methods, adapted to the specific types of conflicts encountered in railway networks, and a genetic algorithm that optimizes the resolution process. The goal is to minimize disruptions and deliver an optimized timetable based on key performance indicators (KPIs) such as total train travel time, number of unresolved conflicts, and number of modified trains.

In this work, we propose an optimization framework that automates the resolution of several types of timetable conflicts, including spatial and temporal overlaps, infrastructure disruptions, and track occupation issues. This framework processes an initial timetable containing conflicts, applies resolution methods tailored to each conflict type, and outputs a new, conflict-free timetable optimized according to a set of predefined KPIs.

The solution methodology is developed within the context of ongoing research in the area of railway optimization and builds on key concepts from mathematical programming and

evolutionary computation. By adopting a genetic algorithm approach, this process efficiently explores the space of possible solutions, combining different resolution methods to identify the optimal conflict resolution strategy. As seen in previous research (Cacchiani et al., 2010), genetic algorithms provide a robust mechanism for handling large search spaces and complex, multi-dimensional problems like railway timetable optimization.
In summary, this document outlines a novel approach for railway timetable conflict resolution, providing a scalable and automated method for optimizing railway schedules. This contribution is positioned within the broader context of optimization in transportation systems and offers practical insights into real-time operational improvements for railway networks.

7.6.2 Problem definition

Timetable optimization in rail networks is a complex task due to the dynamic nature of train operations and the variety of conflicts that can arise. These conflicts occur when two or more trains attempt to use the same track or other shared resources at overlapping times. Additionally, unforeseen infrastructure limitations, such as maintenance or station closures, can exacerbate these conflicts by restricting available routes. As railway networks grow and become more interconnected, the frequency and complexity of these conflicts increase, necessitating a robust and automated solution.
Conflicts in railway timetables typically fall into three broad categories:

- Catch-up Conflicts: These occur when two trains share a route segment, and one of the trains is delayed, leading to an overlap in their scheduled use of the same track or station. Without resolution, this type of conflict can result in further delays for both trains.
- Crossing Conflicts: These happen when trains on intersecting routes arrive at a crossing point simultaneously. The trains cannot occupy the same segment of track at the same time, causing a need for one train to yield to the other, potentially resulting in cascading delays.
- Track Occupation Conflicts: This conflict arises when a train is scheduled to occupy a track or station platform that is already in use by another train, either due to delays or poor scheduling. Resolving this conflict requires adjusting one or both train schedules to ensure sequential use of the shared resource.

The goal of timetable optimization is to automatically resolve these types of conflicts without human intervention. This requires a system that not only identifies the conflicts but also applies pre-defined resolution methods to adjust train schedules in a way that maintains overall operational efficiency. The resolution of these conflicts must be done carefully to minimize disruption and avoid creating new conflicts elsewhere in the timetable.
The input to the timetable optimization process is a timetable containing multiple types of conflicts, along with data regarding:

- Spatial and Temporal Constraints: These include details of when and where the conflicts occur, such as overlapping train schedules, track usage restrictions, and other relevant operational details.
- Set of Conflicts to Resolve: Conflicts could be specific to a particular train or affect multiple trains across the network. The system must be flexible enough to handle a range of conflict types and scales, from localized single-train issues to network-wide conflicts.

A timetable conflict is defined as any situation where a train's planned route is disrupted by the presence of other trains, infrastructure issues, or unavailable resources. For example, a conflict

may occur when two trains are scheduled to occupy the same track section simultaneously. Conflicts may also arise when a train is prevented from continuing its journey due to infrastructure problems, such as planned maintenance or unexpected station closures, which create route restrictions.
In order to resolve these conflicts, the system uses a series of resolution methods, each tailored to a specific conflict type. These methods adjust train schedules by modifying departure or arrival times, rerouting trains, or adjusting the number of stops. The resolution methods are applied automatically based on predefined rules and the characteristics of the conflict. Each method ensures that conflicts are resolved without causing new conflicts or violating operational constraints.

7.6.3 Challenges in Conflict Resolution

Each resolution method must be carefully selected based on the nature of the conflict and the specific constraints of the railway network. For instance, some resolution methods may only be applicable if certain conditions are met (e.g., if the conflict occurs at a station with available tracks or if the delay can be absorbed without impacting subsequent services).
The challenge is to ensure that these resolution methods are applied in such a way that the overall timetable remains robust and efficient. Any adjustment to a train's schedule may have ripple effects across the network, potentially causing new conflicts or affecting train connections, passenger transfers, and resource allocation (e.g., rolling stock and crew). The resolution process must also consider the Key Performance Indicators (KPIs) that govern railway operations.
The optimization process not only resolves current conflicts but also seeks to avoid the introduction of new conflicts during the resolution process. By using a genetic algorithm, the system intelligently explores the solution space, balancing the need for conflict resolution with the overall performance of the railway network.

7.6.4 Process

The primary objective of the timetable optimization process is to automatically resolve conflicts in train schedules to achieve the best possible planning scenario according to predefined KPIs. This process operates without user intervention, requiring only an initial invocation by the user to begin the optimization process. The aim is to minimize disruptions in train services while maintaining overall operational efficiency, including train travel times, the number of modified schedules, and the number of unresolved conflicts.
However, it is important to note that, due to the complexity of the timetable optimization problem, the proposed solution method does not guarantee an optimal solution or even a feasible solution in all cases. This limitation stems from the nature of the conflicts involved and the size of the solution space, which grows exponentially with the number of conflicts and trains. Even with advanced heuristic approaches, such as the use of a genetic algorithm, the solution may not always converge to an optimal or feasible outcome due to computational constraints and the complexity of the problem.

Conflict Sorting and Prioritization
The first step in the process is sorting the identified conflicts based on a configurable criterion. This sorting could prioritize conflicts based on their temporal characteristics, the severity of the impact on the timetable, or a combination of both factors. By establishing a clear order of resolution, the process ensures that conflicts are tackled systematically. The sorting algorithm is

highly flexible, allowing for adjustments to meet the specific needs of the railway operator.
Once sorted, the system processes each conflict individually, applying the most appropriate resolution method for that specific scenario. As conflicts are interdependent, the order in which they are resolved plays a critical role in the quality of the final solution. A change in one conflict's resolution can create ripple effects that alter the availability of resources or the timing of subsequent train services, thus affecting the remaining conflicts in the queue.

Application of Resolution Methods
Each conflict in the sorted list is assigned a predefined resolution method tailored to the specific conflict type (catch-up, crossing, or track occupation conflict). These methods are automatically applied based on the spatial and temporal characteristics of the conflict. For instance, a conflict caused by two trains attempting to use the same track simultaneously can be resolved by adjusting the schedule of one train, rerouting it, or making minor adjustments to its timetable at a preceding node. The application of these resolution methods is controlled by a set of business rules and operational constraints that ensure the feasibility of the resulting schedule adjustments.
It is important to note that while each resolution method is designed to resolve the conflict it targets, the broader impact of its application may introduce new conflicts elsewhere in the timetable. Therefore, the process must continuously evaluate the overall timetable to ensure that the resolution of one conflict does not negatively affect the broader network. This is particularly challenging when dealing with large railway networks, as minor schedule adjustments in one part of the network can cause significant disruptions in other parts.

7.6.5 Genetic Algorithm for Conflict Resolution in Timetabling

In this work, we utilize a genetic algorithm (GA) to address the challenge of resolving timetable conflicts efficiently and effectively. The GA is employed to explore the space of possible timetable adjustments, seeking to find a combination of conflict resolution methods that optimizes the predefined KPIs, such as the number of unresolved conflicts, the total train travel time, and the number of modified train schedules. By using a GA, we aim to balance feasibility with efficiency, given the large solution space and the complexity of railway timetables.

Chromosome Representation
In the context of timetable optimization, each chromosome represents a possible solution to the timetable conflict problem. Specifically, a chromosome encodes a sequence of resolution methods, with each gene corresponding to a particular conflict. The value of each gene represents the resolution method applied to that conflict.
For example, consider a timetable with three identified conflicts, C1, C2, and C3. A chromosome in this context might look like this:
Chromosome 1:($C1M2$,$C2M1$,$C3M3$)
Chromosome 2:($C1M1$,$C2M2$,$C3M3$)
In this case, conflict C1 is resolved using resolution method M2, conflict C2 is resolved using method M1, and conflict C3 is resolved using method M3. Each chromosome is therefore a unique combination of methods applied to the set of conflicts, representing a potential conflict-free timetable.
The order of genes (conflicts) is crucial because the resolution of one conflict may affect subsequent conflicts in the timetable. For example, resolving a track occupation conflict may change the availability of the track for other trains, thereby altering the conditions of later conflicts.
Crossover Operation

Crossover is used to combine the characteristics of two parent chromosomes to create offspring chromosomes. In our application, crossover occurs by selecting a point along the sequence of conflicts and swapping the corresponding resolution methods between the two parents.
For instance, consider two parent chromosomes:
Parent 1:($C1M2$,$C2M1$,$C3M3$)
Parent 2:($C1M1$,$C2M2$,$C3M1$)
A single-point crossover at conflict C2 would result in two offspring chromosomes:
Offspring 1: ($C1M2$,$C2M2$,$C3M1$)
Offspring 2: ($C1M1$,$C2M1$,$C3M3$)
This crossover operation ensures that the offspring inherit some characteristics from each parent while introducing new combinations of resolution methods. The choice of the crossover point is typically random, but the goal is to explore new areas of the solution space while maintaining the integrity of the original schedules as much as possible.

Mutation Operation
The mutation operation introduces diversity into the population by randomly altering the resolution method applied to a conflict in the chromosome. Mutation is particularly important in avoiding local optima by ensuring that the algorithm continues to explore new solutions.
In timetabling, mutation may occur by randomly selecting a conflict in the chromosome and changing its associated resolution method. For example, if we mutate Offspring 1 from the earlier example, the result could be:
Before Mutation: ($C1M2$,$C2M2$,$C3M1$)
After Mutation: ($C1M2$,$C2M3$,$C3M1$)
Here, the resolution method for conflict C2 is changed from M2 to M3, potentially offering a better solution for that particular conflict while maintaining the same resolution methods for the other conflicts. The mutation rate is kept low to ensure that the solution space is explored in a controlled manner, preventing too many random changes that could destabilize the timetable.

Application to Timetabling
In applying the GA to timetabling, the fitness function plays a critical role in evaluating the quality of each chromosome (i.e., each combination of conflict resolutions). The fitness function assesses the number of unresolved conflicts, the total modification to train schedules, and other KPIs. Chromosomes with fewer unresolved conflicts and minimal schedule disruptions receive higher fitness scores, guiding the GA toward better solutions in subsequent generations.
As the algorithm iterates, the crossover and mutation operations allow it to explore different combinations of conflict resolutions, gradually improving the timetable while minimizing the negative impacts on train schedules. However, as with most heuristic approaches, there is no guarantee of finding the globally optimal solution due to the large and complex solution space.
By applying the GA in this way, we can efficiently navigate the challenging problem of timetable optimization, reducing the number of conflicts and improving operational efficiency in a way that would be infeasible with manual adjustments alone. In general terms, the operation follows these steps:

1. Initial Population Selection

In the genetic algorithm (GA) used for timetable optimization, each individual in the population represents a potential solution to the conflict resolution problem, encoded as a *chromosome*. Each chromosome consists of a series of *genes*, where each gene is defined as a *Conflict-Method pair*.

This means that every gene within a chromosome represents a specific conflict in the timetable and the method selected to resolve that conflict.

Importantly, although the user does not directly see the chromosome itself, it serves as an internal mechanism for encoding how conflicts are resolved within the system. The distinction between different methods may not always be immediately visible in the actual train schedules, particularly in cases where methods apply the same adjustment (e.g., adding a delay) at different points along the route. For example, if two methods both delay the train by one minute but apply the change at different stops (A and B), the impact on the overall timetable at the conflict location (node D) would appear identical. However, the *underlying method* used to resolve the conflict is different, as it determines *where* in the route the schedule adjustment is applied.

In summary, the key elements of each gene are:

- *Conflict*: The location, time, and trains involved in the conflict (e.g., an occupied station or track).
- *Method*: The specific resolution approach applied, which dictates *where* along the route the change is made and *which train* is affected. This could involve advancing or delaying a train's schedule at various points (such as stop A or stop B) to resolve the conflict.

Representation of Chromosomes

In our case, the chromosome consists of a sequence of genes representing different conflict-method pairs. Here's an example for clarification:

- *Chromosome*: (Conflict 1 - Method A, Conflict 2 - Method B, Conflict 3 - Method C)
  - *Conflict 1*: A scheduling conflict at node D between Train 1 and Train 2.
  - *Method A*: Delay Train 1 by one minute at stop A.
  - *Conflict 2*: A track occupation conflict at node E.
  - *Method B*: Delay Train 2 by 30 seconds at stop B.
  - *Conflict 3*: A schedule overlap at node F.
  - *Method C*: Advance Train 3 by two minutes at node F to clear the conflict.

In larger, more complex networks, many chromosomes might result in similar timetable outputs, especially when different methods apply the same time shift at various locations. However, the difference between these methods matters for the *system* as it directly influences the resolution strategy and the knock-on effects for other parts of the network. The system needs to know not only *how much* time to shift a train but also *where* to apply that shift to minimize the overall impact on the timetable.

Initial Population Creation

The initial population is created by randomly generating **N** chromosomes, where each chromosome is a combination of conflict-method pairs. The size of the population, **N**, is typically determined as a configurable parameter, with a common approach being to select 1/3 of the total population. This allows for a diverse range of potential solutions to be explored during the optimization process, ensuring that the algorithm can efficiently search through different conflict-resolution strategies. For example:

- *Chromosome 1*: (Conflict 1 - Method A, Conflict 2 - Method B, Conflict 3 - Method C)
- *Chromosome 2*: (Conflict 1 - Method B, Conflict 2 - Method A, Conflict 3 - Method C)
- *Chromosome 3*: (Conflict 1 - Method C, Conflict 2 - Method B, Conflict 3 - Method A)

This initial population forms the starting point for the genetic algorithm's optimization process. As the algorithm progresses through iterations, it selects, combines, and mutates these chromosomes to evolve towards an optimal or feasible timetable solution.

By representing each gene as a conflict-method pair, the algorithm ensures that it has enough flexibility to explore different ways of resolving conflicts, even if those methods appear identical in the final timetable. This approach is essential for handling larger and more complex networks, where small changes in the choice of method can lead to significant differences in overall performance.

2. Initial Population Evaluation

Once the initial population is selected, each chromosome in the population is evaluated through a fitness function that measures how well the proposed solution resolves the conflicts in the timetable. The fitness function is designed to assess both the local impact of each conflict resolution as well as the global effects on the overall timetable. Specifically, it considers not only the immediate resolution of conflicts but also the potential ripple effects (or knock-on effects) that any schedule adjustments might cause throughout the entire timetable.

In the context of timetabling, the fitness function evaluates how well each solution (encoded in a chromosome) maintains the overall efficiency of the railway system while minimizing disruptions. The fitness evaluation follows these key steps:

- **Conflict Resolution Impact:** The chromosome represents a sequence of conflict resolutions, with each gene encoding the timing or order adjustments for specific conflicts. As the algorithm applies the resolution methods encoded in the chromosome, it continuously evaluates the changes these adjustments make to the train schedules.

  For example, if Train 1's arrival time is adjusted to resolve a conflict with Train 2, the fitness function evaluates not only whether this adjustment resolves the conflict but also how it impacts subsequent events in the timetable (e.g., whether it causes new conflicts or delays down the line).

- **Global Recalculation of Timetable:** After each conflict is resolved, the timetable is recalculated in its entirety to account for potential knock-on effects. This means that the algorithm does not stop after solving individual conflicts but instead examines the broader implications of the conflict resolution. By recalculating the timetable for each solution, the algorithm can identify and penalize solutions that introduce new conflicts or result in significant disruptions elsewhere in the network.

- **Key Performance Indicators (KPIs):** The fitness function uses a set of KPIs to quantify the quality of the solution. These KPIs are weighted according to their importance.

The fitness function for a solution is computed as a weighted sum of these KPIs, where each KPI is assigned a weight based on its relevance to the overall system performance:

$f(x1,x2,...,xn)=w1\cdot x1+w2\cdot x2+...+wn\cdot xn$

Here, $x1,x2,...,xn$ represent the values of the KPIs for a given solution (such as the number of unresolved conflicts, travel time, and delays), and $w1,w2,...,wn$ represent the corresponding weights. These weights are configurable and can be adjusted based on operational priorities. For instance, if minimizing delays is a critical concern, the weight for the delay KPI can be increased.

- **Local vs. Global Fitness:**
  - Local Evaluation: Initially, the fitness function evaluates the local impact of resolving each conflict individually. This means assessing how well each conflict is resolved without introducing immediate issues for the affected trains (e.g., does shifting a departure time solve a conflict without causing immediate delays for that train?).
  - Global Evaluation: After resolving each conflict, the timetable is recalculated as a whole to ensure that no new conflicts arise in other parts of the schedule. This global evaluation ensures that the solution considers knock-on effects, such as cascading delays or resource conflicts that might be introduced by changes made earlier in the timetable.

The fitness function therefore serves as a comprehensive tool to evaluate not only whether conflicts are resolved but also whether the entire timetable remains efficient and operationally feasible after adjustments are made.

To continue with our example, let's assume we have the following KPIs:

- $x1$: The number of unresolved conflicts after recalculating the timetable.
- $x2$: The total travel time for all trains in the network.
- $x3$: The number of modified train stops.
- x4 : The number of delays introduced.

For each solution, the fitness function might look like:

$f(x1,x2,x3,x4)=w1\cdot x1+w2\cdot x2+w3\cdot x3+w4\cdot x4$

Where $w1,w2,w3,w4$ are the weights applied to each KPI based on their importance. A solution with fewer unresolved conflicts and minimal disruptions to train schedules would score higher and be selected for the next iteration.

In this way, the fitness function not only measures the direct impact of conflict resolutions but also ensures that the resulting timetable remains feasible and efficient from a global perspective.

3. Selection of Parent Chromosomes for Crossover

To create new solutions in each generation of the genetic algorithm, two *parent chromosomes* are selected from the current population to undergo crossover. This selection process is crucial, as choosing high-quality parents increases the likelihood of producing offspring with improved fitness.

We employ a method known as *Tournament Selection* to select these parents. Tournament Selection is a widely used approach in genetic algorithms, as it allows the algorithm to favor individuals with higher fitness while maintaining some diversity by including less-fit individuals.

The Tournament Selection process follows these steps:

- *Random Selection of Individuals*: A small, random subset of individuals (typically two) is chosen from the population to participate in the "tournament." For instance, in our example with chromosomes 2, 4, 7, and 10, two individuals are randomly selected for each tournament round.
- *Evaluation and Selection of the Fittest Individual:* The fitness of each individual in the subset is evaluated, and the individual with the highest fitness is chosen to proceed to the next generation. This ensures that fitter individuals have a higher chance of being selected, though there is still a possibility for less-fit individuals to participate, preserving genetic diversity.
- *Repeating the Process for Multiple Pairings*: The tournament selection process is repeated multiple times until two distinct parent chromosomes are chosen. For instance, if the algorithm first randomly selects chromosomes 4 and 7, it evaluates their fitness and picks the one with the higher fitness score (e.g., chromosome 4). This process is then repeated for another subset, say, chromosomes 2 and 10, with the fittest among them (e.g., chromosome 2) selected as the second parent.

In this way, Tournament Selection ensures that individuals with better fitness have a higher probability of selection, but the random element introduces genetic diversity, which is essential for avoiding premature convergence to local optima.

4. Crossover to Generate Offspring Chromosomes

Once the two parent chromosomes are selected, they undergo **crossover**, a process that combines their genetic material to create two new **offspring chromosomes**. The goal of crossover is to produce offspring that inherit advantageous traits from both parents, potentially leading to solutions with higher fitness in the subsequent generation.

The crossover process includes the following steps:

a. *Determining Crossover Points:* The algorithm decides whether to use a single-point or two-point crossover, depending on the number of genes and the complexity of the solution. For chromosomes with a smaller number of genes, such as our example (where each chromosome represents a schedule adjustment with only a few conflicts), a *single-point crossover* may be used. For larger chromosomes, a *two-point crossover* is often preferred to allow for a more thorough combination of parental traits.

   - *Single-point Crossover*: A single cut point is chosen randomly along the chromosome, splitting it into two parts. Each offspring inherits one part from each parent.
   - *Two-point Crossover*: Two cut points are chosen, dividing the chromosome into three parts. Each offspring inherits the middle section from one parent and the outer sections from the other parent.

b. *Swapping Genetic Material*: In single-point crossover, each parent's chromosome is split at the chosen crossover point, and the segments from the two parents are combined to form two new offspring chromosomes. For example, if chromosomes 4 and 2 were selected as parents with a single crossover point between the first and second genes, the offspring would be created as follows:

    - *Parent 1 (Chromosome 4)*: (C1: Train 1 arrival at 08:10, C2: Train 2 departure at 08:20, C3: Train 3 arrival at 08:30)
    - *Parent 2 (Chromosome 2)*: (C1: Train 1 arrival at 08:15, C2: Train 2 departure at 08:25, C3: Train 3 arrival at 08:35)
    - *Offspring 1* (Combining first segment of Parent 1 with second segment of Parent 2):
        - (C1: Train 1 arrival at 08:10, C2: Train 2 departure at 08:25, C3: Train 3 arrival at 08:35)
    - *Offspring 2* (Combining first segment of Parent 2 with second segment of Parent 1):
        - (C1: Train 1 arrival at 08:15, C2: Train 2 departure at 08:20, C3: Train 3 arrival at 08:30)

c. *Resulting Offspring Chromosomes*: The two offspring chromosomes now contain a combination of traits from each parent. This process allows the offspring to potentially exhibit new solutions to conflicts in the timetable. The generated offspring will then be evaluated by the fitness function in the next generation to determine their suitability.

By combining portions of each parent's chromosome, the crossover process introduces variation into the population, allowing the genetic algorithm to explore new areas of the solution space. This exploration is crucial for gradually improving the quality of the timetable, as it enables the discovery of configurations that balance efficiency with minimal disruptions. The offspring with higher fitness scores are retained in the population for future generations, further refining the timetable through successive rounds of selection and crossover.

5. Mutation of the Offspring Chromosome

Mutation introduces variability into the offspring chromosomes, allowing the algorithm to explore new solutions that may not be accessible through crossover alone. In the context of timetable optimization, mutation modifies the event timing or train order within a gene to test alternative conflict resolutions.

Each gene in a chromosome represents a specific conflict and the method chosen to resolve it. Mutation randomly selects a gene in the offspring chromosome and changes the current resolution method for that conflict to an alternative method. This mutation process is controlled by a configurable *mutation probability*. Typically, this probability is kept low (e.g., 0.04%) to prevent excessive random alterations, which could destabilize the solution.

The mutation process includes these steps:

- **Determining Mutation Occurrence:** For each offspring chromosome, the system generates a random number (e.g., between 1 and 1000). If the number falls within a predefined range (e.g., 1 to 4 for a 0.4% mutation rate), a mutation will occur in that chromosome for the current iteration.

- **Selecting the Gene for Mutation:** Once a chromosome is chosen for mutation, a specific gene within the chromosome is randomly selected. This gene corresponds to a particular conflict in the timetable, and the mutation will involve changing its resolution method.

  For example, consider a gene associated with conflict *C2*, which has three possible resolution methods (M1, M2, and M3). If the original gene value is *C2M1* (indicating method M1 is used to resolve conflict C2), the mutation would randomly change it to *C2M2* or *C2M3*.

- **Implementing the Mutation:** The mutation replaces the existing resolution method in the selected gene with the new one. This alteration may lead to different timetable adjustments, which will be evaluated for fitness in subsequent iterations. Mutations help the algorithm explore a broader range of solutions, potentially discovering new approaches to conflict resolution that improve timetable efficiency.

6. Replacement in the Population

After generating offspring chromosomes through crossover and mutation, the next step is to *select the best individuals* to form the population for the next generation. This selection process replaces some of the lower-quality individuals from the initial population with the newly created offspring if they offer improved fitness scores.

The replacement process follows these key steps:

a. **Evaluating Fitness of New Offspring:** Each offspring is evaluated using the fitness function, which assesses its overall performance based on KPIs such as the number of unresolved conflicts, train travel times, and schedule disruptions. This ensures that only the most effective solutions (i.e., those with higher fitness scores) are retained.
b. **Selecting Individuals for Replacement:** The algorithm identifies the individuals in the current population with the lowest fitness scores as candidates for replacement. These lower-quality solutions are the least effective in resolving conflicts and maintaining timetable efficiency, making them suitable for replacement by better-performing offspring.
c. **Replacement Decision:** For each offspring, if its fitness score is higher than that of a low-performing individual in the current population, it will replace that individual. However, if the new offspring has a lower fitness score, it will not replace the existing population member, thereby preserving the overall quality of the population.

   This selective replacement process ensures that the population gradually improves over generations. It prevents deterioration by retaining only high-quality individuals, allowing the algorithm to converge toward an optimized timetable solution.

d. **Maintaining Population Diversity:** To avoid premature convergence to suboptimal solutions, the algorithm may occasionally retain some low-fitness individuals to ensure diversity. This balance between selecting optimal solutions and maintaining a varied population helps the algorithm avoid local optima, increasing the chances of finding a globally optimized timetable.

In summary, the mutation and replacement processes work together to introduce variability and ensure that only the most effective solutions are preserved. Mutation explores new solutions by altering specific conflict resolutions within chromosomes, while replacement refines the population by selecting higher-fitness individuals, ensuring that each generation builds on the strengths of the previous one. Through these iterative processes, the genetic algorithm gradually optimizes the timetable, improving efficiency and reducing conflicts over successive generations.

7. Validation to Finish the Algorithm

The algorithm includes a validation step to determine whether the stopping condition has been met, indicating that the optimization process can conclude. Common stopping criteria include:

- **Maximum Number of Iterations:** The algorithm stops after completing a predefined number of iterations. This maximum limit is configurable, allowing the user to control the algorithm's runtime based on computational resources or desired solution quality.
- **Convergence of Population:** The process stops when the population has converged, meaning there is little to no change in the fitness values of individuals across iterations. This stability indicates that further iterations are unlikely to produce significantly better solutions, as the algorithm has reached a plateau.

If neither condition is met, the algorithm continues with additional iterations, applying crossover and mutation operations to generate new offspring, followed by evaluation and selection for the next generation. Each iteration represents a complete cycle of these steps, and with each cycle, the algorithm seeks to refine the timetable solution further.

8. Completion and Output of the Algorithm

Once the stopping condition is satisfied, the algorithm finalizes its output, resulting in an optimized timetable that resolves conflicts based on the best solution found. This solution may still contain some unresolved conflicts or new minor conflicts introduced by timetable adjustments, but it represents the most efficient configuration achieved within the parameters set for the algorithm.

The generated timetable is a distinct scenario from the original one, incorporating optimized adjustments to improve operational efficiency and minimize conflicts. If the user wishes to refine the solution further or apply the optimization to a different scope or scale, they can relaunch the process. Each new run of the algorithm starts from scratch, using the latest data and disregarding past iterations, allowing for flexible adaptation to changing operational needs or updated timetable requirements.

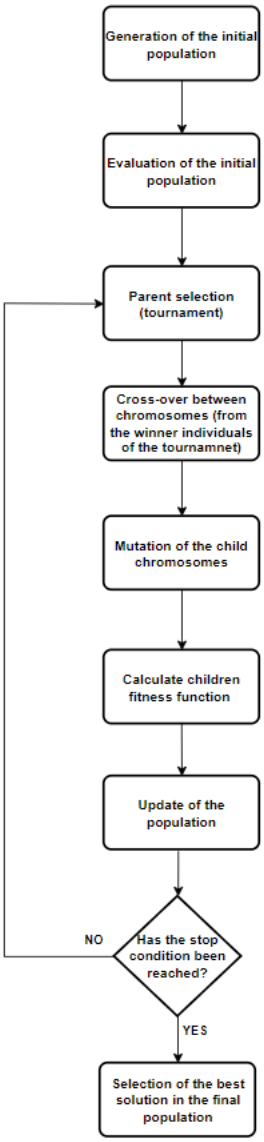


Figure 7.6.1: Flowchart of a genetic algorithm

9. End of the Process

To ensure that the final optimized timetable provides a true improvement over the original, a final comparison is made between the timetable generated by the algorithm and the original one. This comparison is carried out using a *comparison function* based on key performance indicators (KPIs), similar to the fitness function but possibly customized for final evaluation purposes. This function is configurable, allowing operators to adjust the evaluation criteria based on specific needs, such as minimizing delays, reducing unresolved conflicts, or enhancing operational efficiency.
The final comparison process works as follows:

a. **Evaluate and Compare:** Both the optimized timetable and the original timetable are evaluated using the comparison function, which quantifies the effectiveness of each plan based on the chosen KPIs.
b. **Select the Better Solution: I**f the optimized timetable demonstrates improvements in line with the KPIs, it is selected as the final output. However, if the optimized solution fails to outperform the original timetable, the algorithm will revert to the original plan.

This last step ensures that the user receives the more effective of the two timetables. The chosen timetable, either the optimized or the original, is then presented as the final output, along with information about the changes made, remaining conflicts, and the overall impact on performance.

Iterative Reapplication for Enhanced Solutions

Given the complexity of railway networks and the frequent interdependencies between train schedules, the algorithm is designed to address the most critical conflicts in each iteration. However, resolving a set of conflicts in one iteration does not necessarily yield a fully feasible or optimal solution, especially in larger, more complex timetables. To address this, it may be beneficial to reapply the algorithm iteratively.
In practice, running the algorithm multiple times allows for further refinement and incremental improvements to the timetable, addressing any remaining conflicts or knock-on effects that arise after each iteration. In this approach:

- After each iteration, the revised timetable is re-evaluated, and any new conflicts introduced by previous adjustments are identified.
- The algorithm is reinitialized with the latest data, ensuring that each iteration builds on the refinements of the previous one.

7.6.6 Preliminary Results

An example of the initiation and first iteration of the algorithm has been made. The example has three trains that are involved in three conflicts. Figure 7.6. shows the graphical timetable of the initial scenario.

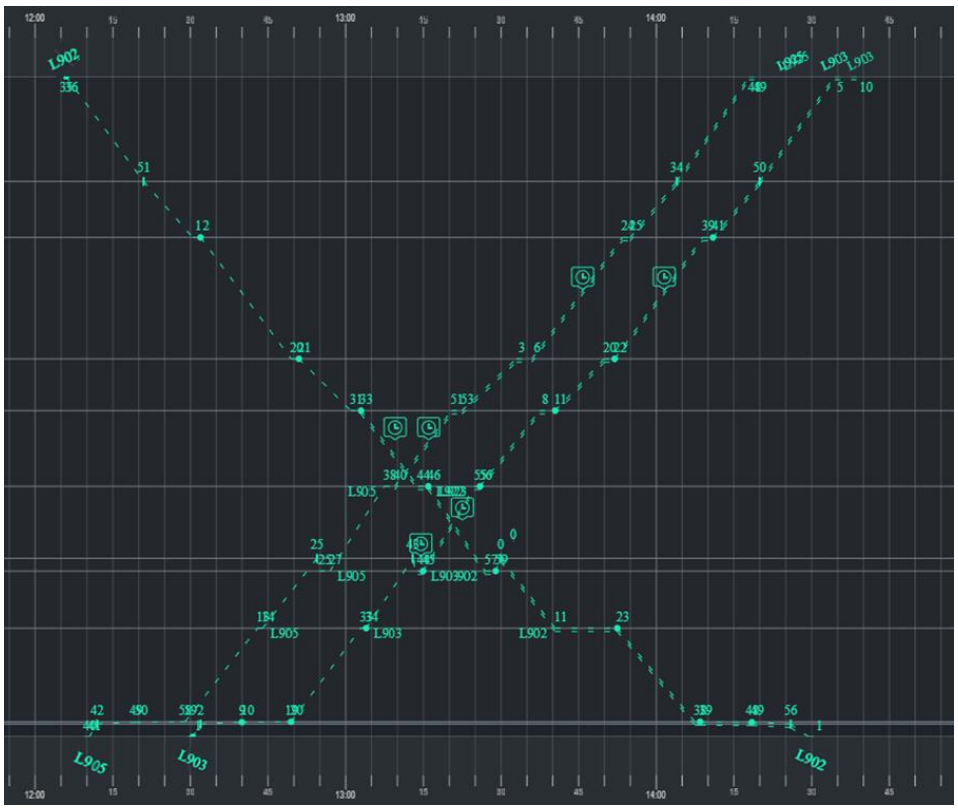


Figure 7.6.2: Initial situation with three different trains with three conflicts.

The graphical timetable shows the timetable of the trains, in which horizontal axe represents the time and vertical axe represents the space. The discontinuous lines represent the planned runs of

these trains. Each line is the planned run of a train. The numbers visualized are the minutes of arrivals and departures at the time control zones in stations. The symbols like a “clock” indicate time conflicts.
Each conflict indicator belongs to a train run and indicates that the train is involved in a time conflict. In this case, each conflict involves two trains. Because of that, two conflict indicators are represented for each conflict, one per train involved in the conflict. Hence, although the picture shows six conflict indicators, there are only three conflicts in the scenario.
From now on, the names of trains will be called L902, L903 and L905. These trains will have the planned schedule.

Table 7.6.1: Schedule and train routes of the trains in the initial situation with the conflicts.

Arrival->Departure Departure<-Arrival	L902	L903	L905	
Station 1	21:01:00	19:00:00	18:40:00	
	21:00:00	19:01:00	18:41:00	
Station 2	20:56:30	19:02:30	18:42:30	
	20:56:30	19:02:30	18:42:30	
Station 3	20:49:00	19:09:00	18:49:00	
	20:48:00	19:10:00	18:50:00	
Station 4	20:39:00	19:18:30	18:58:30	
	20:38:00	19:19:30	18:59:30	
Station 5	20:23:00	19:33:30	19:13:30	
	20:11:00	19:35:30	19:14:30	
Station 6	20:00:30	19:44:30	19:25:00	
	20:00:30	19:44:30	19:25:00	
Station 7	19:59:30	19:45:00	19:25:30	
	19:57:30	19:47:00	19:27:30	L902-L903
Station 8	19:46:30	19:57:30	19:38:00	Crossing
	19:44:30	19:59:30	19:40:00	L902-L905
Station 9	19:33:30	20:10:30	19:51:00	Crossing
	19:31:30	20:12:30	19:53:00	
Station 10	19:21:30	20:22:00	20:03:30	
	19:20:00	20:24:00	20:07:30	L903-L905
Station 11	19:02:30	20:41:00	20:25:00	Catch-up
	19:01:00	20:43:00	20:26:30	
Station 12	18:51:30	20:52:30	20:35:30	
	18:51:30	20:52:30	20:35:30	
Station 13	18:36:30	21:06:30	20:49:30	
	18:35:30	21:08:00	20:50:30	

The three conflict for the scenario are the following:

- Crossing conflict that involves L902 and L903 located in head on section between stations Station 7 and Station 8
- Crossing conflict that involves L902 and L905 located in head on section between stations Station 8 and Station 9
- Catch-up conflict that involves L903 and L905 located between stations Station 10 and Station 11

Each conflict can be solved by different resolution methods. Each resolution method is designed to solve one conflict. The algorithm will use three methods to solve the conflicts of the scenario. The combination between the conflicts and methods provide all the generated chromosomes used by the algorithm. In these tests, for simplification, only three methods are considered. It must be noted that the impact of increasing conflict is more significant, as the number of solutions grows exponentially. However, increasing methods also has a considerable effect in the number of solutions.

Table 7.6.2: Methods chosen for the algorithm that could solve conflicts C1, C2 and C3.

Methods	C1: Crossing between L902-L905	C2: Crossing between L902-903	C3: Catch-up between L903-L905
M1	Delay the stop time in the train L905 5min 31s on Station 8	Delay the stop time in the train L902 12min 1s on Station 8	Delay the stop time in the train L905 32min 51s on Station 10 (Over-taking)
M2	Delay the stop time in the train L902 18min 31s on Station 9	Delay the stop time in the train L903 17min 1s on Station 6	Delay the stop time in the train L903 21s on Station 10
M3	-	Create a stop in the train L903 16min 1s on Station 6	-

Method M3 is empty for conflicts C1 and C3 because the method (create stop) is not applicable to solve these conflicts.
The conflicts are organised chronologically and these KPIs have been selected:

- Total amount of time that the schedule of all trains is modified. (x_1)
- The number of trains modified. (x_2)
- The number of unresolved conflicts left. (x_3)

The KPIs is a set of key performance indicators to assess the quality of the solution process used. These KPIs are weighted according to their importance to generate the fitness function. The fitness function takes into account these KPIs and assigns them a weight based on its relevance to the overall system performance. The fitness function is a configured function within the algorithm to assess the proposed solution.
The fitness function is $f(x_1, x_2, x_3) = 60\%x_1 + 20\%x_2 + 20\%x_3$.
The total population is the set of all possible combinations conflict-method as proposal to solve the scenario. From this total population, the algorithm must select an initial population composed by chromosomes, in this test, 1/3 of the total population is included in the initial population. The percentage of the total population included is a configuration parameter. The selection of the initial population is random.

Table 7.6.3: Total Population.

Individuals/Chromosomes	Total population
1	C1M1-C2M1-C3M1
2	C1M1-C2M1-C3M2
3	C1M1-C2M2-C3M1
4	C1M1-C2M2-C3M2
5	C1M1-C2M3-C3M1
6	C1M1-C2M3-C3M2
7	C1M2-C2M1-C3M1
8	C1M2-C2M1-C3M2
9	C1M2-C2M2-C3M1
10	C1M2-C2M2-C3M2
11	C1M2-C2M3-C3M1
12	C1M2-C2M3-C3M2

Table 7.6.4: Initial Population

Initial population	
1	C1M1-C2M1-C3M1
6	C1M1-C2M3-C3M2
8	C1M2-C2M1-C3M2
9	C1M2-C2M2-C3M1

Through the fitness function, we obtain a value for each chromosome that evaluates how well a particular solution performs. To properly apply the fitness function it is necessary to normalise the data that are added like the time and the number of conflicts because they do not have the same units, and so the value can be established between 0-1 to sum.

Table 7.6.5: Travel times

Arrival->Departure Departure<-Arrival	Travel time
L902	2:25:30
L903	2:08:00
L905	2:10:30
Total (sum of times)	6:44:00
Total in seconds	24240

Table 7.6.6: Schedule changed and affected by the methods of chromosome 1

Chromosome 1	C1 Delay train L905 in Station 8 0:05:31	C2 Delay train L902 in Station 8 0:12:01	C3 Delay train L905 in Station 10 0:27:20
Station 1	18:40:00	21:13:01	18:40:00
	18:41:00	21:12:01	18:41:00
Station 2	18:42:30	21:08:31	18:42:30
	18:42:30	21:08:31	18:42:30
Station 3	18:49:00	21:01:01	18:49:00
	18:50:00	21:00:01	18:50:00
Station 4	18:58:30	20:51:01	18:58:30
	18:59:30	20:50:01	18:59:30
Station 5	19:13:30	20:35:01	19:13:30
	19:14:30	20:23:01	19:14:30
Station 6	19:25:00	20:12:31	19:25:00
	19:25:00	20:12:31	19:25:00
Station 7	19:25:30	20:11:31	19:25:30
	19:27:30	20:09:31	19:27:30
Station 8	19:38:00	19:58:31	19:38:00
	19:45:31	19:44:30	19:45:31
Station 9	19:56:31	19:33:30	19:56:31
	19:58:31	19:31:30	19:58:31
Station 10	20:09:01	19:21:30	20:09:01
	20:13:01	19:20:00	20:40:21
Station 11	20:30:31	19:02:30	20:57:51
	20:32:01	19:01:00	20:59:21
Station 12	20:41:01	18:51:30	21:08:21
	20:41:01	18:51:30	21:08:21
Station 13	20:55:01	18:36:30	21:22:21
	20:56:01	18:35:30	21:23:21

The schedule of the train L902 and L905 is modified from the colour change (from dark green to light green or from yellow to orange). Colour green indicates the direction of the train goes from Station 1 to Station 13. Colour yellow-orange indicates the direction of the train does from Station 13 to Station 1.

*Table 7.6.7**8**: Travel time changed after the application of the methods*

Chromosome 1	Travel time with the modifications	Differences with the original
C1 - Delay train L905 in Station 8		
0:05:31	2:16:01	0:05:31
C2 - Delay train L902 in Station 8		
0:12:01	2:37:31	0:12:01
C3 - Delay train L905 in Station 10		
0:27:20	2:43:21	0:32:51

Table 7.6.89: Data needed for the algorithm from the travel time changed of the chromosome 1

Modified trains	Total travel time differences	Total differences in seconds
2	0:44:52	2692

Then, the calculation of the value of the fitness function of the chromosome 1 is done. Table 7.6.10 shows the KPIs calculated for the chromosome 1 and the value of the fitness function.

Table 7.6.9: KPIs calculated to the chromosome 1

	Chromosome
KPIs	1
Total amount of time that the schedule of all trains is modified (s)	0,055528053
Total amount of trains on which a modification has been made (u)	0,666666667
Total number of unresolved conflicts (u)	0,555555556
Fitness function (minimise)	0,277761276
Fitness function (maximise)	0,722238724

Where all have been calculated in a proportion with the normalise total. The process shall be made for the rest of chromosomes at the same way. Table 7.6.11 shows the KPIs and the value of the fitness function for all the chromosomes of the initial population.

Table 7.6.10: All KPIs calculated

	Chromosomes			
KPIs	1	6	8	9
Total amount of time that the schedule of all trains is modified (s)	0,055528053	0,022937294	0,023349835	0,063572607
Total amount of trains on which a modification has been made (u)	0,666666667	0,666666667	0,666666667	0,666666667
Total number of unresolved conflicts (u)	0,555555556	0,277777778	0,277777778	0,555555556
Fitness function (minimise)	0,277761276	0,202651265	0,20289879	0,282588009
Fitness function (maximise)	0,722238724	0,797348735	0,79710121	0,717411991

Continuing with the steps on the algorithm, the tournament between the chromosomes is done obtaining two winners. After that, the winner chromosomes are crossed over obtaining two children. Table 7.6.12 shows the tournament between the chromosomes and the cross-over.

Table 7.6.11: Start to the first iteration

TOURNAMENT Competition 1:		Fitness Value
6	C1M1-C2M3-C3M2	0,79734873
8	C1M2-C2M1-C3M2	0,79710121
Winner/Father1	C1M1-C2M3-C3M2	0,79734873

Competition 2:		Fitness Value
1	C1M1-C2M1-C3M1	0,72223872
9	C1M2-C2M2-C3M1	0,71741199
Winner/Father 2	C1M1-C2M1-C3M1	0,72223872

Cross-over:	Cut		CHILDS	
C1M1-C2M3-C3M2	C1M1-C2M3	C3M2	C1M1-C2M3-C3M1	Child 1
C1M1-C2M1-C3M1	C1M1-C2M1	C3M1	C1M1-C2M1-C3M2	Child 2

The fitness value is obtained for each child (it has the same calculation as before). Table 7.6.13 shows the calculated KPIs from the obtained children.

Table 7.6.12: Calculated KPIs from the obtained children

	Chromosomes	
KPIs	Child 1	Child 2
Total amount of time that the schedule of all trains is modified (s)	0,022937294	0,021534653
Total amount of trains on which a modification has been made (u)	0,666666667	1
Total number of unresolved conflicts (u)	0,277777778	0,555555556
Fitness function (minimise)	0,202651265	0,324031903
Fitness function (maximise)	0,797348735	0,675968097

The value of fitness function for child 1 is better than for child 2. A comparison between the initial chromosomes and the children is made. Table 7.6.14 shows the comparison between fitness values.

Table 7.6.13: Result from the comparison between fitness values

Fitness values comparison (> =Green; < Red)		Child 1	Child 2
		0,797348735	0,675968097
Chromosome 1	0,72223872		
Chromosome 6	0,79734873		
Chromosome 8	0,79710121		
Chromosome 9	0,71741199		

Replace the worst Discarded child

The worst chromosome (9) is replaced by the child 1. Table 7.6.15 shows this replacement in the population.

Table 7.6.14: Change in the population

Population Before	
1	C1M1-C2M1-C3M1
6	C1M1-C2M3-C3M2
8	C1M2-C2M1-C3M2
9	C1M2-C2M2-C3M1
Population After↓	
1	C1M1-C2M1-C3M1

6	C1M1-C2M3-C3M2
8	C1M2-C2M1-C3M2
Child 1*	C1M1-C2M3-C3M1

*Notice that this child is the individual 5 from the total population. The crossover operation has led to a solution that was not present in the initial population.

And here ends the first iteration. If the termination has not been met the algorithm must go on with the new population (1, 6, 8 and child 1) until if it those. If the algorithm stopped in this iteration described, it would return the child 1. Figure 7.6.3 shows the final situation from the child 1.

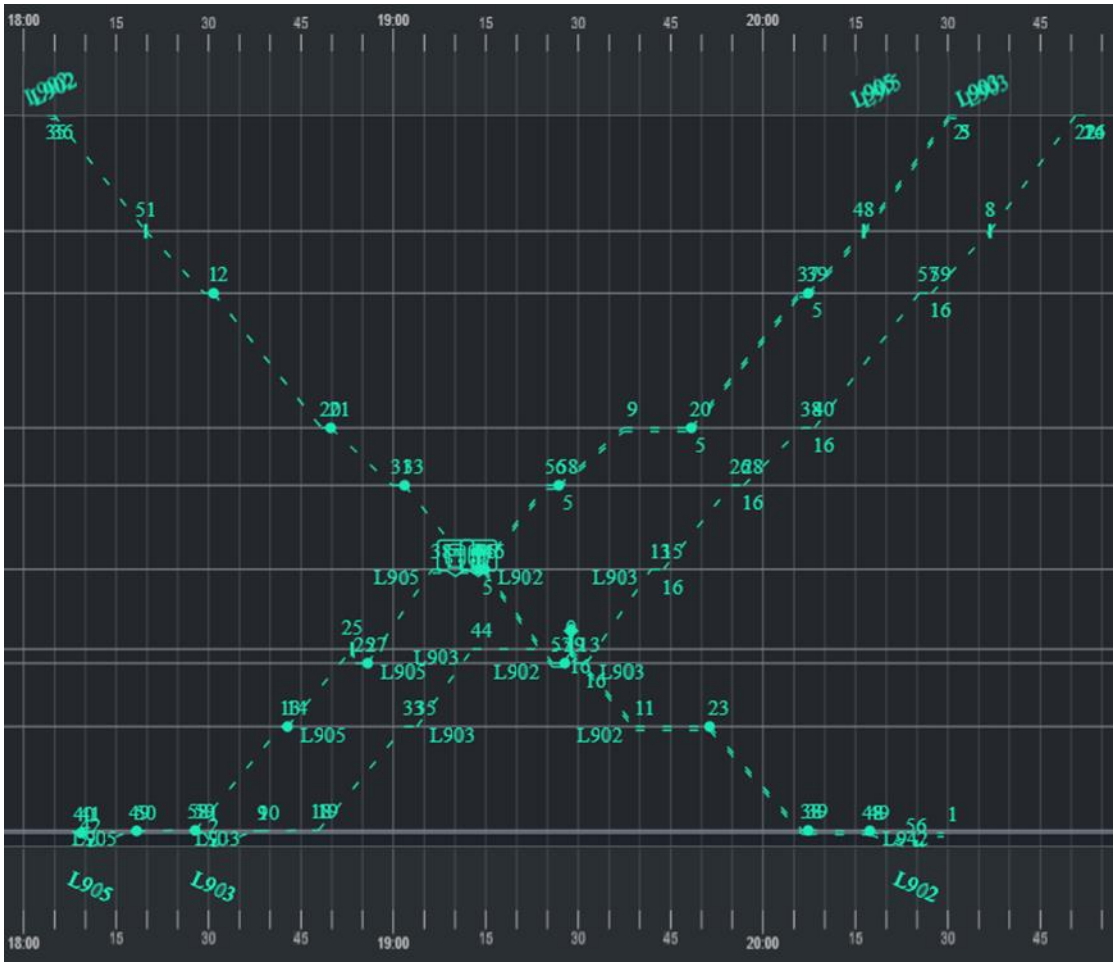


Figure 7.6.3: Final situation form the child 1

Table 7.6.15: Changes made in the schedule trains from the Child 1

Child 1	C1 Delay train L905 in Station 8 0:05:31	C2 Create stop in train L903 on Station 6 0:16:01
Station 1	18:40:00	19:00:00
	18:41:00	19:01:00
Station 2	18:42:30	19:02:30
	18:42:30	19:02:30
Station 3	18:49:00	19:09:00
	18:50:00	19:10:00
Station 4	18:58:30	19:18:30
	18:59:30	19:19:30
Station 5	19:13:30	19:33:30
	19:14:30	19:35:30
Station 6	19:25:00	19:44:30
	19:25:00	20:00:31
Station 7	19:25:30	20:01:01
	19:27:30	20:03:01
Station 8	19:38:00	20:13:31
	19:45:31	20:15:31
Station 9	19:56:31	20:26:31
	19:58:31	20:28:31
Station 10	20:09:01	20:38:01
	20:10:01	20:40:01
Station 11	20:27:31	20:57:01
	20:29:01	20:59:01
Station 12	20:38:01	21:08:31
	20:38:01	21:08:31
Station 13	20:52:01	21:22:31
	20:53:01	21:24:01

The result planning is returned along the information of the KPIs. We get the planning schedules with the highest value in the fitness function from the obtained population through the iterations.

7.6.7 Conclusions

The genetic algorithm approach presented here offers practical advantages in reducing computation time and memory usage when addressing timetable conflicts in railway networks.
Compared to traditional optimization methods, such as Mixed Integer Programming (MIP) and heuristic approaches like Tabu Search, genetic algorithms (GA) have been shown to be more efficient, particularly in dense and complex railway networks.
Studies, such as those by Acuna-Agost & Cordeau (2011) and D'Ariano et al. (2010), show that GAs achieve faster processing times and reduced memory usage, especially in complex and densely interconnected railway networks. This efficiency makes GAs suitable for real-time applications

where rapid responses are critical, outperforming exhaustive methods that struggle with the high computational demands of congested scenarios.
However, it is essential to recognize that the algorithm does not guarantee an optimal solution, nor does it ensure a fully feasible or high-quality outcome after a single iteration. Realistic scenarios often involve complex, densely interconnected timetables with hundreds of conflicts, where iterative applications were necessary to converge towards a practical solution.

To assess the performance and applicability of the algorithm, it is recommended to test it on larger, realistic instances that reflect the actual complexity of operational railway networks. This includes:

- **High-density networks:** These networks, with many trains and complex schedules, provide a meaningful challenge and help determine if the algorithm can produce feasible timetables that meet operational standards.
- **Multiple rounds of application:** Running the algorithm iteratively in these realistic settings will reveal its effectiveness in achieving substantial conflict resolution and its potential to converge on a feasible solution.

Through iterative application and rigorous testing on realistic scenarios, the algorithm can offer a valuable tool for railway practitioners seeking to reduce conflicts and improve timetable efficiency, though with the understanding that additional refinement may be needed to meet the high standards required for operational deployment.

8. Algorithms for rolling-stock planning

Task 6.4 is about rolling stock planning. Physical trains are formed by compositions of individual train units. Such train units must flow in the railway network to be ready for their assignments in the right place (i.e. departure stations) at the right time (before departure times). The units may arrive directly from other train compositions, or from some yards. The problem of finding an optimal assignment of units to trains, and of planning their movements in the railway network, is called *rolling stock rotation.* Section 8.1 describes an integer flow model to rolling stock rotation, which is then solved by use of a MILP solver.
When not in use, train units must be parked in suitable yards. Moreover, they must be maintained (e.g. washed, repaired, checked, etc.). The problem of parking units and plan their movements in the yards (so that no conflicts arise in the use of tracks or resources) is called *stabling*. In Section 8.2 we describe an algorithm (based on Constraint Programming and local search) to solve the train stabling problem.

Table 8.1: overview Chapter 8

Section	Subtask	Use case	Demonstration
8.1	6.4.1	UC-FP1-WP3-28	11.1
8.2	6.4.2	UC-FP1-WP3-20	11.2

Each section starts with describing the background, then give a detailed problem description defining the use case in detail. In addition, we give a detailed description of the methodology to tackle the use case. For all use cases this is done by describing the algorithm. Finally, we validate the developed algorithms on representative problem instances (TRL 4 validation). The results are reported at the end of each section.

8.1 A MILP-based algorithm for rolling stock scheduling

8.1.1 Background

A (train) timetable establishes, for a line or portion of railway network, the routes and the schedule of the train services for passengers of freight in the portion during a specified period. To implement a timetable in practice, we need to assign the required physical rolling stock units (e.g. locomotives, wagons, etc.) to the train services. Each train service is characterized by an origin (station) with associated departure time, and a destination (station) with associated arrival time. The physical units required by a train service must be available in the origin before the departure time and are released when the train arrives at destination. After arrival, the units can be used by other train services. An unused unit can be parked in special yards (the *depots*) or in the arrival stations of the last services which used the unit. When the unit is required from a train service in a given station, then either it is already there, released by a previous train service, or it must arrive from a depot. The movement of rolling stock units must be thus carefully planned to satisfy the requirements of the timetable. In general, one wishes to minimize the size of the fleet or the cost of acquisition of new rolling stock, ensuring a certain level of robustness of the solution. Because of its centrality in railway management and planning, rolling stock rotation problem (RSR) has been widely studied in the literature (see, e.g., Alfieri et al. 2006, Cacchiani et al. 2010, Cacchiani et al. 2013, Grimm et al. 2023, Hoogervost et al. 2021). The RSR is an optimization problem, normally tackled by means of mathematical programming techniques.

This study, developed within the WP6 of Europe's rail joint undertaking project MOTIONAL, will follow the mainstream. The solution to the RSR in WP6 is required to complement the short- and long-term timetabling algorithms developed in the same WP. We will focus on a particular version of the RSR, arising in the context of the Norwegian railway system. One of us is actually responsible of the design of long-term timetables for the Norwegian Railway Directorate (Jernbanedirektoratet). To tackle the problem, similarly to what is done for instance in Cacchiani et al. 2010, we will use integer flow optimization. The movement of rolling stock units of a given type in the railway network can indeed be viewed as an integer flow. We will focus only on passengers' timetables and the corresponding train services, which only make use of *electrical multiple units* (EMUs). An EMU is a powered train unit that can drive by itself. Each unit is composed by one or more wagons and can be combined with other EMUs to form a longer train and carry passengers. There are several types of EMUs with different technical and physical characteristics - such as number of wagons, power, number of passengers for wagon, etc.

In Figure 8.1.1 one of the train units types available in the Norwegian fleet.

Figure 8.1.1 Train unit type 69 – Serie C.

8.1.2 The problem

8.1.2.1 Stations and depots

EMUs, possibly of different types, are assembled to form a train running through the railway network. For our purposes, it is sufficient to picture such network as a set of stations S and a set of depots L connected by tracks. Stations are places where passengers are embarked and disembarked. Depots are places where EMUs are parked for the night or during the day, and where various maintenance operations can be carried out. We assume a depot $l \in L$ has a maximum capacity k_l to accommodate units. In our simplified model, k_l represents the upper bound for the sum of the lengths of the individual units parked in the depot. In practice, the situation is more complicated, because this capacity is calculated by summing up the lengths of all depot tracks, but a unit cannot be split between two tracks. For instance, if the depot has two tracks of 40 meters each, then the capacity is calculated as 80 meters. However, the depot can accommodate two units of 35 meters each but cannot accommodate a single unit of 45 meters. In this study we will relax this integrality constraint.

Train units must be shunted from depots to the stations where they are required, and from the stations to the depots where they are parked. For each depot $l \in L$ and each station $s \in S$, we let β_s^l be the time to move a train unit from l to s (and vice versa). Note that some depots may be located at stations (and in this case the shunting time will be 0 or very small). Crucially, since we are dealing with long-term timetabling, in this work we neglect the planning of actual movements of empty units through the network.

8.1.2.2 Train services

We plan only for 24 hours of operations (*planning horizon* or *planning day* or, simply, *day*). The start of the planning day does not necessarily coincide with the start 00:00 of the standard day. Each physical train is assigned to a specific *train service*, which is defined by 1. the *route*, i.e. the

sequence of stations and tracks between stations traversed by the train, and 2. the *schedule*, that is the time (in the day) when the train service should enter and exit each station and track in its route. Moreover, the timetable also specifies the (minimum) number of passengers that the train service should be able to carry around. All train services are described in the *train timetable*. We denote by T the set of train services, and, for each $t \in T$, we let $d^t \in S$ and $a^t \in S$ be the first (departure, origin) and last (arrival, destination) station of the route of service t, respectively. Also, we let τ_d^t and τ_a^t be the scheduled times at departure and arrival of t, respectively.

8.1.2.3 EMUs types

The EMUs are classified by type, and we denote by U the set of EMU's types and by $T^u \subset T$ the set of train services that can use unit type $u \in U$. For each type $u \in U$, we let π_u be the number of units of type u. If the rotation plan uses more than π_u units of type u, then the train operator will have to purchase them, at the unitary acquisition cost c_u^A. Finally, we denote by λ_u the length of unit type $u \in U$.

During a planning day, an EMU can be used by (we say *assigned to*) several train services. Complying with our reference infrastructure manager business rules, in this work we assume that **all units assigned to a train service are of the same type**, so mingling types is forbidden. For each unit type $u \in U$ and train service $t \in T_u$, we denote by μ_u^t the number of units of type u required by the service t, which depends on the expected number of passengers transported by t. We also consider a maximum number $\hat{\mu}_u^t$ of units of type u that can be assembled on train service t. Note that it may be convenient to assign to t more than the minimum required number of units, so to make the additional units available at the arrival station to be assigned to departing train services.

A unit which is assigned to a train service t, must be at hand at its departure station d^t some time before the departure time τ_d^t of t. This time depends on the unit type and on the origin of the unit. For instance, if the unit comes directly from another train service arriving at the same station on the same track as train t, this buffer time can be rather small, otherwise, we will need to shunt the unit. So, for a pair of distinct train services $r, t \in T$, and unit type u, we let $\delta_u^{rt} \in Re$ be the minimum (time) required to shunt u from r to t and to be assembled. Conventionally, we let δ_u^{rt} be a very large constant when the unit cannot be shunted, for instance because the first train service arrives at a different station, or too late. Similarly, for $l \in L$ and $t \in T$, we let $\bar{\delta}_u^{lt}$ be the time required to shunt to train service t a unit arriving from depot l. Again, we let this be a very large constant if the depot l is "too far" from the departure station of t. Note that this time is not related to the time $\beta_{d^t}^l$ necessary to travel from depot l to the departure station d^t of t.

Finally, by business rules (in Norway), at the beginning and at the end of the planning day all EMUs are parked in some depot. Because we are looking for stationary solutions, a unit parked in depot $l \in L$ in the beginning of the day, will be parked in the same depot l at the end of the day.

Constants and parameters

S	Set of stations
L	Set of depots
T	Set of train services
U	Set of unit types
$T^u \subseteq T$	Set of train services that can use unit type $u \in U$
$H = \{1, \dots, q\}$	Discretized Planning Day
$D = L \times H$	Set of depot slots

$d^t \in S$	first (departure) station of train service $t \in T$
$a^t \in S$	last (arrival) station of train service $t \in T$
$\tau_d^t \in \Re$	schedule departure time of train service $t \in T$ from its first station
$\tau_a^t \in \Re$	schedule arrival time of train service $t \in T$ to its last station
μ_u^t	required number of units of type u for service $t \in T$
$\hat{\mu}_u^t \in Z_+$	maximum number of units of type u for service $t \in T$
π_u	number of available units of type $u \in U$
c_u^A	acquisition cost of one unit of type $u \in U$
λ_u	length of unit of type $u \in U$
$\tilde{\delta}_u^{rt}$	time to shunt unit type $u \in U$ from train service $r \in T$ to $t \in T$
$\bar{\delta}_u^{lt}$	time to shunt unit type $u \in U$ from depot $l \in L$ to train service $t \in T$
$\tilde{c}_u^{rt}$	cost to shunt unit type $u \in U$ from train s. $r \in T$ to train serv. $t \in T$
$\bar{c}_u^{lt}$	cost to shunt unit type $u \in U$ from depot $l \in L$ to train service $t \in T$
k_l	capacity of depot $l \in L$
β_s^l	Time to move a unit from (to) depot $l \in L$ to (from) station $s \in S$

Table 8.1.1 Model sets and constants.

8.1.2.4 Problem statement

The purpose of this work is to assign units to train services, enough to transport all expected passengers. Each train unit assigned to a train service t may come from a depot or from another train service t' ending at the departure station of t. In any case, the units assigned to t must be at hand at its departure station some time before the departure of t, enough to allow the necessary shunting movements. As a primary objective, we want to minimize the cost of acquisition of new units. We may also consider the cost of transferring units from depots to stations and vice-versa, and a cost for shunting units from train services to train services, but we will come back to this in the next section.

8.1.3 The Model and the Unit Assignment Graph

The flow of the EMUs of a given type from depots to (the departure station of) train services, from train services to train services, and from train services to depots, can be modeled as a circulation on a directed graph, called *Unit Assignment Graph*.

To represent the stock of EMUs in a depot at some specific times of the planning day we introduce the concept of *depot slot*. To this end, we discretize the planning day into time slots $H = \{1, \dots, q\}$. Each slot $h \in H$ corresponds to some half-open interval $[\tau_h, \tau_{h+1})$ in the 24 hours, starting at time τ_h. Note that slot 1 do not necessarily coincides with the beginning of the standard day. So, for instance, the planning day may start at time $\bar{\tau}_1 =$ 4:00 am. Also, the last slot $[\tau_q, \tau_{q+1})$ may start in the next standard day, and it will always end in the next standard day, namely at the beginning of the first slot of the next standard day (i.e. $\tau_{q+1} = \tau_1^{+1}$). In our example $\bar{\tau}_{q+1} =$ 4:00am^{+1}). We call the interval $[\tau_q, \tau_1^{+1}]$ as *planning night break*, or simply night break. We assume there is no movement of units in the (planning) night break.

A *depot slot* is simply a pair $(l, h) \in L \times H = D$, representing a depot l at time slot h. Note that in the night break, no trains can go out or in the depot. So, all and only the units of type u present at

depot l at the end of the planning day will be available at the beginning of the next planning day.

8.1.3.1 Unit Assignment Graph

We now introduce, for each unit type $u \in U$, the unit assignment graph $G_u = (V_u, A_u)$. In what follows, we let $T_u \subseteq T$ be the set of train services which can use units of type $u \in U$.
The set of nodes is $V_u = (T_u \cup D)$, that is, we have node for each train service that can use u and for each depot slot. A node associated to a depot slot (l, h) represents the stock of EMUs of type u stocked in depot l at the beginning τ_h of time slot h. A node associated with a train service represents the number of units of type u assembled on the corresponding physical train.

The arcs represent to possibility to "send" units between nodes. In particular, we let $A_u = A_u^{dt} \cup A_u^{td} \cup A_u^{tt} \cup A_u^{dd}$, distinguishing among four types of arcs:

- *Depot to Train arcs:* A_u^{dt}. For $d = (l, h) \in D, t \in T_u$, the arc $(d, t) \in A_u^{dt}$ if a unit from depot slot $d = (l, h)$ can reach t on time to be shunted before t departs from its departure station d^t, that is $\tau_h + \beta_{d^t}^l + \bar{\delta}_u^{lt} \leq \tau_d^t$.

- *Train to Depot arcs:* A_u^{td}. For $d = (l, h) \in D, t \in T_u$, the arc $(t, d) \in A_u^{td}$ if a unit from the arrival station a^t of t can reach depot l within the beginning τ_h of time slot h, that is $\tau_a^t + \beta_{a^t}^l \leq \tau_h$.

- *Train to Train arcs:* A_u^{tt}. For $r, t \in T_u$, the arc $(r, t) \in A_u^{tt}$ if the arrival station a^r of r coincides with the departure station d^t of t and there is enough time to shunt the train, i.e. $\tau_a^r + \tilde{\delta}_u^{rt} \leq \tau_d^t$.

- *Depot to Depot arcs:* A_u^{dd}. We assume that units are not shunted directly between two distinct depots[3]. So, we only have arcs from one depot slot $(l, h-1)$ to the next depot slot (l, h) to represent the stock of EMUs remaining in l from period $h-1$ to h. As usual, we let $h = 1$ when $h = q + 1$ (where q is the last time slot).

Note that, because there is no movement of train units during the "night break", arc $((l, h), (l, 1)) \in A_u$ is the only arc outgoing node (l, h) and the only arc incoming node $(l, 1)$ and all the units of type u present at depot l at the beginning of the planning day, flow on this arc.

[3] This is again according to our reference business rules

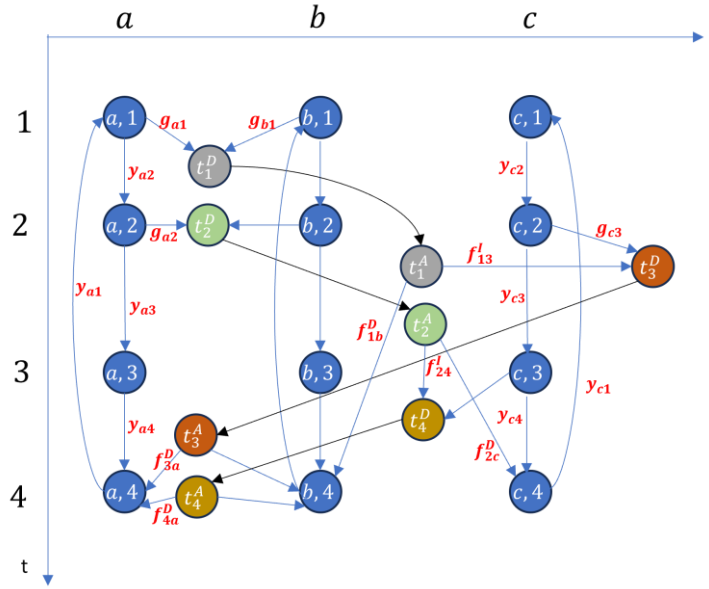


Figure 8.1.2. Example of flow network relative to one unit type, with 4 periods, 3 depots (a, b, c) and 3 trains departures t_1^D, t_2^D, t_3^D from origin and arrivals t_1^A, t_2^A, t_3^A at destination. This is a slightly expanded representation of graph G_u, to represent each node in a space-time diagram. We have split each train service node into a departure node and an arrival node. We added also an arc from the departure node to the arrival node to represent the flow of units associated with the train service. The actual flow network G_u can be obtained by contracting this extra arc, i.e. by identifying the departure and arrival node of each train service. On the y-axis we represent time, with the start of the planning day slots, whereas on the x-axis we represent stations and depots (assumed to be along a line).

8.1.3.2 Variables and constraints

Network, variables, costs

$V_u = T_u \cup D$	Set of nodes
$A_u = A_u^{dt} \cup A_u^{td} \cup A_u^{tt} \cup A_u^{dd}$	Set of arcs from depot slots (nodes) to train service
A_u^{dt}	Set of arcs from D to T_u
A_u^{td}	Set of arcs from T_u to D
A_u^{tt}	Set of arcs from T_u to T_u
A_u^{dd}	Set of arcs from D to D
$f_{uij}^T \in N$	Number of units sent from $i \in T_u$ to $j \in T_u$, for $(i,j) \in A_u^{tt}$
$f_{uid}^D \in N$	Number of units sent from $i \in T_u$ to $d \in D$, for $(i,d) \in A_u^{td}$
$g_{udi} \in N$	Number of units sent from $d \in D$ to $i \in T_u$, for $(d,i) \in A_u^{dt}$
$y_{udh} \in N$	Number of units sent to $d = (l,h) \in D$ from $(l, h-1)$, for $d \in D$
$x_{ui} \in \{0,1\}$	1 if the unit type u is assigned to $i \in T_u$
$w_u \in N$	number of extra units of type u to acquire
$c_u^A \in R_+$	cost of each extra unit of type u, to acquire
$c_{uij}^T \in R_+$	cost of sending one unit from $i \in T_u$ to $j \in T_u$, for $(i,j) \in A_u^{tt}$

$c^D_{udj} \in R_+$	cost of sending one unit from $d \in D$ to $j \in T_u$, for $(d,j) \in A^{dt}_u$

Table 8.1.2 Node and arc sets, MILP variables, MILP costs.

Flow variables, for each unit type $u \in U$

- For $(i,j) \in A^{tt}_u$: $f^T_{uij} \in N$ number of units of type $u \in U$ which flow from train service $i \in T_u$ to train service $j \in T_u$.
- For $(i,d) \in A^{td}_u$: $f^D_{uid} \in N$ number of units of type $u \in U$ which flow from train service $i \in T_u$ to depot slot $d \in D$.
- For $(d,i) \in A^{dt}_u$: $g_{udi} \in N$, number of units of type $u \in U$ sent from the depot slot $d \in D$ to train service $i \in T_u$.
- For $((l,h-1),(l,h)) \in A^{dd}_u$: y_{ulh} number of units of type $u \in U$ which flows into depot slot $(l,h) \in D$ from depot slot $(l,h-1)$, that is the number of units *remaining* in depot l at time h from the previous time slot. Recall also that the index h must be treated $(\mathrm{mod} q)$, and so $y_{ulq+1} = y_{ul1}$. Variable y_{ul1} represents the number of units of type u staying in the depot from the previous planning day and available at the beginning of the planning day. Also, since by assumption there are no other arcs incoming node $(l,1)$, y_{ul1} is the **total number of units** of type u present in the depot l at the beginning of the day.

Other variables

- $x_{ui} \in \{0,1\}$, which is 1 if unit type $u \in U$ is assigned to train service $i \in T$.
- $w_u \in N$, number of extra unit of type $u \in U$ to acquire.

Note that, since the units present in the last period q will stay in the depot during the night break to be available in the first period 1, and nothing arrives during the night break, the variable y_{ul1} is actually the **total** number of units of type u available at the depot l at the beginning of the day.

The number of units of type u flowing into a train service i equals the required number

$$\sum_{(j,i)\in A^{tt}_u} f^T_{uji} + \sum_{(d,i)\in A^{dt}_u} g_{udi} = \mu_{ui} x_{ui} \quad u \in U, i \in I.$$

The units of type u flowing out of i (towards other train services or depots) equals the incoming flow of units.

$$\sum_{(i,j)\in A^{tt}_u} f^T_{uij} + \sum_{(i,d)\in A^{td}_u} f^D_{uid} = \sum_{(j,i)\in A^{tt}_u} f^T_{uji} + \sum_{(d,i)\in A^{dt}_u} g_{udi} \quad u \in U, i \in I.$$

The number of units of type u flowing into depot service $d = (l,h) \in D$ equals the number of units flowing out from d.

$$y_{ulh} + \sum_{(i,d)\in A^{td}_u} f^D_{uid} = y_{ulh+1} + \sum_{(d,i)\in A^{dt}_u} g_{udi}\,, \quad u \in U, d = (l,h) \in D,$$

with $h+1 = 1$ when $h = q$.

In our simplified model, we consider that the units are "splittable" among the tracks of the depot, so what matters is the total length of the units present at the depot in a given time slot, which

should not exceed the total length of all tracks of the depot[4]. Because during a time slot units can go in and out the depot, in order to make sure that the capacity is never exceeded we need to bound the total incoming flow of units into the depot slot. So, the constraint becomes:

$$\sum_{u \in U} \lambda_u \, y_{ulh} + \sum_{u \in U} \lambda_u \sum_{(i,d) \in A_u^{td}} f_{uid}^D \leq k_l, \quad u \in U, d = (l,h) \in D.$$

However, if we assume that the time slots are such that, during each time slot, trains can only either arrive to, or depart from the depot, then the constraints above simply becomes

$$\sum_{u \in U} \lambda_u \, y_{ulh} \leq k_l, \quad u \in U, (l,h) \in D.$$

Each train service must be assigned a unit type

$$\sum_{u \in U : i \in T_u} x_{ui} = 1, \quad i \in T.$$

If we assign more units of type u than available, we need to purchase them. Now, recalling that all circulating units will be in some depot at the beginning of the planning day, the total number of units of type $u \in U$ is simply $\sum_{l \in L} y_{ul1}$. Then, the extra number w_u of units to be acquired must satisfy the constraint:

$$w_u \geq \sum_{l \in L} y_{ul1} - \pi_u \quad u \in U$$

8.1.3.3 Objective function

The objective function is the sum of different terms. The main component is the acquisition of extra rolling stock. To model this, we introduce variable w_u which is the number of extra units of type u that we need to acquire. So, we want to minimize the total acquisition cost, that is:

$$\min \sum_{u \in U} c_u^A \, w_u$$

Next, we may consider additional terms to the objective function. In particular, these are cost related to the origin of the rolling stock used for some train service j. These costs can factor in various needs and should be carefully calibrated.

- When the origin is another train service $i \in T$ (i.e. $(i,j) \in A_u^{tt}$), the cost c_{uij}^T of shunting a unit of type u may depend on how far the arrival track of i is from the departure track of j. For instance, if i arrives at the same track from where j departs, the shunting cost is 0.

- The cost of shunting units from a depot (slot) c_{udj}^D (such that $(d,j) \in A_u^{dt}$ should always be larger than the cost of shunting from another train service, also because the unit will travel empty from the depot to the departure station. The cost should also be proportional to the distance of the depot from the departure station. Finally, the depot slot may also play a role, since earlier time slots could be preferable for robustness consideration, but at the same time a unit should not arrive too early at the departure station as it may occupy tracks.

The final objective function writes as

[4] A more precise model should take the individual lengths of tracks and units into account, and the capacity constraint would transform into integer knapsack constraints

$$\min \sum_{u \in U} c_u^A \, w_u + \sum_{u \in U} \sum_{(i,j) \in A_u^{tt}} c_{uij}^T \, f_{uij}^D + \sum_{u \in U} \sum_{(d,j) \in A_u^{dt}} c_{udj}^D \, g_{udj}$$

As shown in Cacchiani et al. 2010, when $|U| = 1$ then the problem can be solved efficiently. In paper, besides the flow formulation, they also develop a path formulation whose relaxation can be solved more effectively (by column generation). They also introduce a new class of cuts when the number of cars assigned to a train service can vary (between 1 and 2).

A similar flow model is presented in Alfieri et al. 2006.

8.1.4 Computational experience

As mentioned in the introduction, the solution to the rolling stock rotation problem in WP6 is required to complete the timetabling solutions produced by the algorithms developed by WP6.2 (long-term timetabling). The experiments carried out in this section are intended to demonstrate that the developed method, that is our extended integer flow model, solved by the commercial solver GUROBI, will be able to satisfy the requirements and find possibly optimal solutions in reasonable time for the expected instances. Our experiments were carried out on a server with microprocessor Intel(R) Xeon(R) Gold 6240 CPU 2.60GHz.

In the current computational session, we have at hand one real-life timetable instance, relative to the Jærbane line, from Stavanger to Egersund. This line and associated timetable are representative of the expected instances in terms both of size of the network (28 stations), and number of train services, see Table 8.1.3. Because we did not have reliable data about the size and location of the depots, we created a synthetic instance by considering 3 "artificial" deposits, located respectively at the two end stations of the line (Stavanger and Moi) and in the center of the line. Also, the planning day is subdivided in 12 periods of 120 minutes each, starting at midnight. Finally, we consider two types of train unites and, for each train, the minimum number of train units is 2. As for the cost, we consider the number of train units used. We remark that although this instance is artificial, its dimension is representative of the type of instances that will pop up in the current rolling stock planning in Norway (and in other European countries), and thus of the instances which be handled in the demonstrations in WP7. In any case, further developments to the model and the algorithms will be carried out in the first year of WP7.

Table 8.1.3. Instance data

Line	# Stations/Stops	Train Services	Depots	Periods	Types
Jærbane	28	180	3	12	2

In Table 8.1.4, we report the size of the MILP model and the solution time to find the optimal solution. The problem could be solved in 0.28 seconds of computing time, implying that the model can be routinely solved for the size of interest for the Norwegian planners.

Table 8.1.4. Experiment results

ID	# Variables	# Constraints	Time (ms)	Cost
I1	17870	908	280	240

8.2 A local-search algorithm for rolling stock stabling

8.2.1 Background

Rolling stock stabling is a complex problem in the planning of railway operations. Yards are often near or in busy city centers, making it difficult and expensive to expand while passenger counts continue to grow. Creating the rolling stock stabling plan is a time-consuming process, requiring planners to start weeks in advance. This reduces the amount of actual realization information one can take into account and it is desired to create the stabling plan closer to the day of operation. Rolling stock stabling is also the enabler for executing planned timetables and therefore a critical part of the logistic process. Because of all of this, decision support is crucial.

8.2.1.1 Problem description

Stabling the rolling stock happens on a railway node between passenger services, mainly overnight. Such a railway node is usually defined as a station with its surrounding yards where the rolling stock should be stabled. The trains arrive at a platform track in the station, the passengers disembark, and the doors close. From this point onward the train is the responsibility of the node planner, until the train is brought to the planned platform track for departure. Only passenger trains are considered, consisting of one or more train units which are made up of multiple carriages. These train units are able to move on their own without a separate locomotive.
The rolling stock stabling plan is created after the timetable has been created, the network planning has been completed, and rolling stock has been assigned to the trains. All trains in the timetable have been assigned tracks where each arrival and departure at platform tracks in stations should take place. The specific routes these trains take are considered fixed, and crew shifts have been determined.

The trains stabled on a node might require service, such as, internal and external cleaning, toilet emptying, and mechanical inspections or repairs. To perform these activities specific facilities and/or crews might be required. It is also possible for trains to arrive and depart in different compositions, meaning that trains need to be split into separate units and combined again to form the correct compositions required for their departure. Some departing services might not have been assigned (fixed) rolling stock.

Combing these aspects, a correct rolling stock stabling plan moves each shunt train around the node such that no activities are delayed, the trains are stabled at tracks without exceeding track lengths or blocking movements of other trains. Moreover, the trains should be split and combined into the correct compositions. Driver schedules should also be created assigning specific drivers to each movement, taking into account driving and walking times. Furthermore, the trains need to receive all required services and therefore be stabled at the correct track and time to receive that service. Schedules for the service teams and facilities also need to be created to make sure that the trains can receive the service at the planned time. Lastly, the rolling stock must be correctly assigned to departing services such that the required compositions are met.

Local search solver

Previously, a local search solver was created to solve the rolling stock stabling problem (Van den Broek et al. (2021)). The solver uses several solution methods (among others MIP, CP, Local Search), and solves the rolling stock stabling plan in its entirety, taking into account matching of ingoing and outgoing rolling stock, parking and servicing of rolling stock, and personnel

assignment. The *matching* determines in which departing train composition and in which position of that departing train composition an arriving train unit will depart.
The core of the solver is a local search algorithm. Its input is an initial rolling stock stabling plan that is complete (containing all required activities such as services, train movements, matching, service schedules, and the order in which all this should be performed), but it is allowed to contain infeasibilities, such as delays, crossings and track length violations. Driver shifts do not need to be explicitly determined in the initial plan, as these are chosen heuristically by the local search solver. Since infeasibilities (or conflicts) are allowed such an initial plan can be easily constructed, and this is currently done with relatively simple heuristics (using MIP, CP) with many simplifications. The local search is then used to iteratively improve the plan by small, local mutations to the plan until a conflict-free plan has been found.

The heuristics and logic in the construction of the initial plan are relatively simple, which may result in an initial plan with many conflicts. This works fine for smaller nodes, but makes the problem hard to solve for large nodes like Utrecht. Firstly, the local search currently needs long computation times at large nodes to eliminate the conflicts (after 10 hours the node of Utrecht still has a couple of dozens conflicts). Secondly, the local search creates plans with no clear structure as the local search randomly makes many small changes to the initial plan. This is undesired since plans are easier to understand and execute if they are always of a more or less similar structure.

It is expected that the local search will perform much faster on larger nodes if it starts with a high-quality initial plan, and that, at the same time, the plans can be made more recognizable to planners. Therefore, it is our aim to improve the quality of the initial plan using more advanced algorithms in this work as part of Task 6.4.

The node of Utrecht, also the *demonstrator* for this research, will serve as the main test node for judging the quality of the improved method for the construction of an initial plan. Utrecht Central is the busiest train station (both in terms of number of passengers and number of trains) of the Netherlands. It consists of three separate shunt yards, where each shunt yard is only connected to a few platform tracks at the station (see Figure 8.2.1).

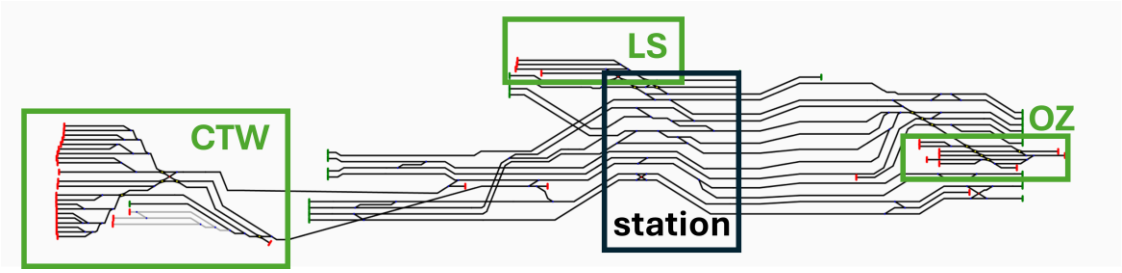


Figure 8.2.1: Schematic overview of the node of Utrecht consisting of a station and three yards (Landstraat (LS), Oost-Zijde (OZ) and Cartesiusweg (CTW).

8.2.1.2 High-level description of the algorithms

The proposed approach for the construction of a high-quality initial plan is to divide the creation of the initial plan into subproblems, focusing on the stabling and moving aspects of the problem. It solves the movements between station and yards first, as this is complicated because of the density of the traffic. This leads to separate problems for the yards which can then be solved independently. The found solutions are combined into a single plan for the entire node, by adding routing, service tasks by the same simple heuristics that are currently used in the creation of the initial plan. The combined solution is not necessarily conflict-free. We do expect that there are far

less conflicts than in the currently used approach, so the local search needs less time to find a conflict-free plan.
The following two algorithms have been created in this project, both using the CP-SAT solver included in Google OR Tools. Prior to these algorithms a matching has already been determined using an existing MIP model.

1. A *yard selection* algorithm which is a Constraint Programming (CP) model that, based on the fixed and predetermined matching, selects a yard for each shunt train. The model should guarantee that the movements to the yard do not conflict with routes of other shunt trains and the through-trains (whose schedule is input). Also, the selected yard should have facilities to perform all required services on the shunt train and the capacity of the yard should not be exceeded.

2. A *stabling algorithm* which is run separately for each yard. This is another CP model that finds the stabling tracks for all the rolling stock to be parked at this yard in such a way that the total length of the parked trains on a track does not exceed the track length at any point in time, and that shunt trains are never blocked (i.e., locked-in) by other shunt trains when they move (e.g., for departure or service). The matching that was previously determined right before the yard selection algorithm, is allowed to be changed. The initial matching was determined on the *node* level, and it was used by the yard selection algorithm to make sure that the chosen yard is a good choice for both the arriving and departing unit. However, on the *yard* level the chosen matching may not be the best since details of the yard were not taken into account yet. Therefore, the stabling algorithm has the freedom to change the matching of all the units that are sent to the yard under consideration, as a correctly chosen matching can prevent lock-ins.

These algorithms are replacements for an existing MIP model, which assigns a single parking track for each shunt train, thereby preferring tracks that allow for the most services required for that shunt train. Contrary to the above algorithms, this MIP model does not take into account other traffic or lock-ins of shunt units by other units.

The above algorithms are integrated into the local search solver. The construction of the full node plan out of the algorithms' output data is similar to the one described in Section 2.3.4 of Van den Broek (2022). Only this time the information from the original MIP model is replaced by that of the newly developed Algorithms 1 and 2. Service schedules are created using an existing heuristic of the local search solver.

8.2.1.3 Terminology

For the sake of clarity, a glossary can be found below for the sometimes rather specific terminology used throughout the text.

Glossary

Combine of two trains is the activity that connects the two trains into a single train. The opposite of a combine is a split.
Gateway is an access point of a yard. It separates the yard from the other part of the rail infrastructure in the sense that entering or leaving a yard can only happen by passing through a gateway. A yard can have multiple gateways.
Matching is an assignment of arriving train units to departing train unit services. There is some freedom in the assignment, since, typically, it is not specified which actual physical train unit should be put in a departure train composition, but only its unit type and the number of carriages. If multiple units of the same type and number of carriages arrive, any

of them can be put into the departure composition that requests such a unit.
Node is a station, its surrounding yards and the infrastructure in-between.
Relocation is a movement of shunt train within a single yard from one stabling track to another.
Reversal is a change of the moving direction of the train where the front cabin becomes the rear cabin and vice versa. During a reversal the train is parked at a parking track, the driver gets out of the front cabin and moves to the cabin on the other side of the train, which then becomes the front cabin.
Rolling stock stabling plan is an integral plan that specifies all the movements of shunt trains over a node starting from their arrivals at the node until their departures (usually along one of the platforms in the station). Its most important aspects are the movements to and from a yard, the movements within a yard, the services performed at a yard by which service team and on which service track, and the train driver assignment.
Shunt train is a plannable train for the node planner.
Split of a train is the activity that splits a train into two. Since train units are the smallest indivisible parts of a train, a train can only be split if it contains more than one train unit. The opposite of a split is a combine.
Train composition is an ordered list of train units that specify the units in a train. The order is important here. A train with the same train units but in a different order is considered to have a different composition. A train composition typically consists of 1 to 3 train units.
Train unit is a part of a train composition that cannot be split into smaller parts. It usually consists of multiple carriages. It is able to move on its own (no locomotive required). All plannable trains in this text are able to move on their own.
Transfer is a movement of shunt train between two yards.
Yard is a part of the rail infrastructure dedicated to parking and servicing shunt trains. It is separated from the rest of the infrastructure by a small number of access points called gateways.

8.2.2 Algorithm descriptions

8.2.2.1 Algorithm 1: Yard selection

The goal of the yard selection algorithm is to select, for a given matching, the best yard for each arriving and departing shunt train that needs to be shunted. In the step-by-step plan for the creation of the initial solution, it is carried out right after the matching has been solved. So, at this point, it is known in what departure composition each of the arriving units will be placed.

Concept of the model

Each yard has one or more access points called *gateways*. In a pre-processing step of the algorithm, the shortest route is calculated from the arrival track of an arriving shunt train to each gateway, and from each gateway to the departure track of a departing shunt train. In addition, for each of these routes it is calculated whether these can be performed within a reasonable timeslot after arrival / before departure without interfering with through-traffic. Since the through-traffic is fixed, these routes and corresponding timeslots can be calculated beforehand.
It is allowed that the chosen arrival and departure yard for a train unit differ so that the unit needs to make a transfer between its arrival and departure yard. This might be required, for example, in order to fulfill all required services of which only a subset can be performed on each yard. These transfers are not chosen explicitly and the taken routes are calculated as a post-processing step.

Conflicts between shunt trains themselves are ignored in this version of the algorithm, since it considerably makes the model more complex. It is our intention to prevent these conflicts as much

as possible in a later version of the model.

Model description

The yard selection algorithm picks a route and a time for each arriving and departing shunt train to or from a gateway such that

- all service requirements of each unit can be fulfilled at the chosen yards;
- the yard capacity is not exceeded at any point in time;
- each unit arrives at a yard before it departs from the yard,

while minimizing the following aspects using a weighted objective function. The used weights are described in Section 8.2.3.

- the number of routes overlapping with fixed movements;
- the number of reversals.
- the number of transfers of units.

The route and timing of a transfer are computed in the post-processing. Therefore, the yard capacity is in fact not fully known when the model is solved (in case there are any transfers). This is dealt with by only including a train in the computation of the capacity of its departure yard.

A full mathematical formulation of the problem is found in Appendix G.

8.2.2.2 Algorithm 2: Stabling algorithm

Once a route to or from a yard has been determined for each arriving or departing composition by the yard selection algorithm, it is known precisely when arriving train compositions and departing train compositions arrive at or depart from which yard and from which gateway. The stabling algorithm can then be run for each yard *separately*, to find the best stabling plan. It does so by

- selecting a matching of arriving to departing units;
- dividing the units over the stabling tracks

in such a way that as few units as possible get locked-in by other units without exceeding any of the track lengths at any time. A unit blocked by another unit when it needs to move, is said to be *locked in*. A conflict-free (i.e., no lock-ins occur) stabling plan provides a way to stable all incoming trains in such a way that all departing train compositions can depart according to the given departure plan without any of them being locked in by other units.

However, it is rare that a conflict-free solution exists if units can only be put on a single track without any movements to other stabling tracks. Therefore, each unit is allowed to switch tracks once in the stabling algorithm model. Such a movement will be referred to as a *relocation*.

Although a matching has already been determined before the yard selection algorithm, the stabling algorithm is allowed to change the matching again, but only for those units that are sent to the yard under consideration. The initial matching was determined on the *node* level, and it was used by the yard selection algorithm to make sure that the chosen yard is a good choice for both the arriving and departing unit. However, on the *yard* level the chosen matching may not be the best since details of the yard were not taken into account. For instance, on one-sided tracks, in general, train compositions departing latest will most likely be made up of units that arrived early. Since the freedom of choice of this assignment greatly influences the possibility of finding a conflict-free plan, the stabling plan and matching are determined simultaneously.

Concept of the model

The idea of the model is to model each arrival at and departure from a track of a unit as an event and assign an order to these events. If it is known in what order train units arrive at and depart from a track and from which side, one can derive whether any lock-ins occur. The goal of the model is to choose the tracks and sides in such a way that no lock-ins occur (or as few as possible if the absence of lock-ins is a soft constraint). The model is formulated at the train unit level because train compositions can be split and combined, and so, for instance, lock-ins need to be determined at a train unit level, and, consequently, the entire model is formulated at the train unit level.
The order is global, so not per track, and hence the values for the order numbers should be unique. Even if two units are in the same arriving composition, the order for the arrival events are different. The unit arriving first at the track gets the lowest order number and the order number gets incremented for each following unit in the train composition. Similarly, for departures, the first unit leaving the track gets the lowest order number.

A stabling plan is considered better if the number of relocations is lower. However, the number of relocations needs to be weighed against the number of splits and combines that result from the chosen matching, which are also costly operations in practice.

The model does not include all aspects of a full rolling stock plan. As always, when one starts building a new algorithm, one starts out with the core and adds additional features if the model turns out successful. In this case, the parking and matching of the trains is most crucial and have therefore been included. Train driver schedules are not considered, since this is usually not the bottleneck at a yard. Services scheduling is important, and would ideally be included in following versions of the model. Some ideas on this topic are explained in Section 8.2.2.3.

Model description

The model needs to select two tracks for each train unit. The first track is called the *arrival track*, the second is called the *departure track*. Both tracks can be the same for one unit, in which case there is no relocation for that unit. This case is actually preferred, since relocations are costly. It also needs to determine from what side each track is entered and left. Moreover, it needs to match each arriving unit to a departing unit.
This all done minimizing lock-ins, split and combines, and relocations in a weighted objective function. The used weights are described in Section 8.2.3.

The decision variables cannot be chosen freely and have to fulfill the following conditions in order to get a consistent stabling plan:

- arriving units are matched to exactly one departing unit;
- order numbers are unique;
- order numbers should be consistent if train units move together;
- relocations can only happen after arrival at the yard and before departure;
- only one activity can happen as the same time.
- when a relocation is underway, no arrival, departure or other relocation can take place (since only one movement at a time is allowed per yard);
- attached arriving / departing units should be parked at the same arrival / departure track;
- units can only be parked at tracks that are reachable from the gateway;

- units can only enter / leave a track from a given side if the gateway can be reached from that direction (if it is not a relocation);
- a unit must be able to go from the arrival track to its departure track (if it is a relocation)
- units can only leave the arrival track from a given side if the departure track can be reached from that direction (if it is a relocation);
- units can only enter the departure track from a given side if from that direction the arrival track can be reached (if it is a relocation).
- the length of the units parked on a track never exceeds the track length at any point in time.

A full mathematical formulation of the problem is found in Appendix G.

8.2.2.3 Considering service

The previously described stabling algorithm allows one movement from a stabling track to another stabling track per train unit. However, on many yards, units need to be moved to a separate service track for maintenance inspections or cleaning, after which they return to one of the stabling tracks. In these cases, at least two movements between stabling tracks per train unit are necessary (to and from the service track). One could extend the stabling algorithm to have two relocations per unit. This would mean the addition of another set of track variables and the corresponding arrival and departure side and order variables, not to mention another set of even more complex constraints. This gets increasingly complex with the number of allowed relocations, and therefore the previously explained modeling of the stabling algorithm is not suitable for such an extension. Luckily, some solvers (for instance, Google OR Tools CP-SAT) have the possibility to formulate *circuit constraints*, which makes the modelling of train movements in a yard easy. A circuit is a loop over nodes where each node is visited at most once. In our case, the nodes are the stabling tracks together with an artificial track that represents all the gateways. Each circuit is required to start and end at this track. The visits of multiple tracks can then easily be modelled by requiring that the sequence of movements of each train unit is a circuit. The stabling algorithm of Section 8.2.2.2 can be obtained if each circuit contains at most 3 nodes (artificial track, arrival track and departure track).

Usually, in practice, shunt trains are first parked at a stabling track, moved to one of the service tracks, and then parked again at a stabling track so it does not unnecessarily block the service track. In order to include service on designated service tracks, one could allow for larger circuits and require that each shunt train comes across one of the designated service tracks for a minimum duration. The downside of a circuit constraint is that the track after service must be different from the first parking track, which is an unnecessary condition in practice.

Although the circuit constraint makes modelling the rotation across the yard easy, our current implementation of the model turned out to be too slow for practical use at the moment, and so this model was not explored further in this work package. We hope that further tweaking the model might result in better, and hopefully, acceptable performance in the future. The mathematical description of the current model can be found in Appendix G.

8.2.3 Experiments

To compare the newly created algorithms to the previously existing algorithms, experiments were performed on real life scenarios. These scenarios were created for multiple railway nodes in the Netherlands of different sizes including Enkhuizen, Enschede, Amersfoort, Rotterdam, and

Utrecht. Although the primary focus has been the node of Utrecht (as this is the demonstrator for this project), other scenarios have been included to test the algorithms on different layouts. The properties of the used scenarios can be seen in Table 8.2.1.

Although these scenarios are based on real life past days, data for some details was not available and therefore estimated. Service requirements, service capacity, and available train drivers were estimated based on previous input from node planners, maintaining similar service team capacity usages. Driver schedules were not included for Enschede, Utrecht 2, and Utrecht 3 for now, since these are numerous and had to be filled in by hand. Furthermore, as per existing business rules, the matching for trains staying overnight was allowed to be changed, while the matching of all other trains was considered fixed.

For each experiment the scenarios were run with 18 random seeds, causing different matchings and random states for the solvers for each run. By doing this, the input for the algorithms is permuted for each run, thus resulting in a better picture of the performance of the algorithms on a scenario.

As stated before, the described algorithms aim to provide good initial solutions for an existing local search method (Van den Broek et al. (2021), Van den Broek (2022)). To test the effectiveness of the new initial solution algorithm the local search algorithm was run on the initial solutions for each run and the results were recorded. The time limits used in the experiments are shared between the initial solution and the local search (i.e. if the initial solution took 50 seconds and the time limit was 120s the local search was allowed to run for 70 seconds).

Table 8.2.1: Scenario characteristics

Scenario	Duration	Trains	Train units	Fixed activities	Tracks	Yards	Drivers
Enkhuizen	72h	23	28	375	7	1	yes
Enschede	24h	40	58	321	7	1	no
Rotterdam 1	24h	46	50	1937	30	4	yes
Rotterdam 2	24h	54	65	2355	30	4	yes
Amersfoort	24h	27	38	1899	24	1	yes
Utrecht 1	24h	146	174	4406	53	3	yes
Utrecht 2	24h	113	150	3208	54	3	no
Utrecht 3	24h	143	200	3296	54	3	no

For each run the time spent in each part of the algorithm, the objective value of the found initial solution, the objective value of the solution after the local search were recorded and analyzed. The objective value of the local search solver contains many aspects of the plan, such as crossings between trains, delayed trains, activities outside crew shifts, etc. From experience it is known that in most cases planners are able to resolve remaining conflicts in the plans once the costs get below about 100 (which corresponds rougly to 10 conflicts).

Experience has shown that sometimes the stabling algorithm does not find a feasible solution if the track length constraints are hard constraints. Although the yard selection algorithm does try not to exceed the capacity of the yard, it cannot always prevent this. Sometimes the number of trains is too large to fit on the yards. In practice, the surplus of trains can be parked at the station but this is not supported yet by the yard selection algorithm. In order not to end up with an infeasible problem, and in turn no initial plan, the track length constraints are taken soft in the stabling algorithm if its first run was infeasible.

Both yard selection and the stabling algorithm have a set of parameters, which are always set to

the same values (see Tables 8.2.2 and 8.2.3). None of these values have been tuned so far.

Table 8.2.2: Overview of the parameters values for the yard selection algorithm

parameter name	value
max runtime	60 seconds
arrival yard possible weight	10
departure yard possible weight	10
reversal weight	1
cutting loss factor	0.8
minimum time on yard	30 minutes
maximum time not to yard	30 minutes

Table 8.2.3: Overview of the parameters values for the stabling algorithm

parameter name	value
max run time	10 minutes
max #relocations	6
dislocation weight	1
split weight	1
combine weight	1
relocation weight	1
track length violation weight	1 per 10 meters

8.2.3.1 Experimental setup

All experiments were performed on a virtual machine in the cloud with 72 Intel(R) Xeon(R) Platinum 8168 vCPU's and 144 GiB of RAM. The CP models were solved by the Google OR-Tools CP-SAT v9.3.10497 solver using a single thread per run. A single thread was chosen to improve reproducibility. The settings of the algorithms were not yet tuned and should be considered further in future work.

8.2.3.2 Experiment 1

In the first experiment the local search was run until a shared time limit with the initial solution was reached. This time limit differs per scenario based on estimated difficulty of the scenario. In this experiment the local search continues after reaching feasibility, trying to improve the quality of the solution. This experiment was performed on all scenarios. In total 144 runs were performed for both the new and old algorithms. The chosen time limits can be seen in Table 8.2.4.

Table 8.2.4: Experiment 1 time limits

Scenario	Time limit
Enkhuizen	2 min
Enschede	5 min
Rotterdam 1	4 min
Rotterdam 2	6 min
Amersfoort	6 min
Utrecht 1	10 hours
Utrecht 2	10 hours
Utrecht 3	10 hours

8.2.3.3 Experiment 2

In the second experiment the scenarios were run until a feasible solution was found or a time limit was reached. This was performed to test the expectation that feasible solutions will be found faster with improved initial solutions. This experiment was not performed for the Utrecht scenarios, as currently no feasible solution is found for either algorithm within a reasonable time limit. The time limit per scenario can be found in Table 8.2.5.

Table 8.2.5: Experiment 2 time limits

Scenario	Time limit
Enkhuizen	20 min
Enschede	50 min
Rotterdam 1	40 min
Rotterdam 2	1 hour
Amersfoort	1 hour

8.2.4 Results

Ideally, one would judge the solutions of yard selection algorithm and the stabling algorithm directly. However, as far as we know, no general objective measures of the quality of a yard selection plan or a stabling plan exist. Therefore, the algorithm solutions are judged by their impact on the local search solver. The judgment by human planners is part of WP7.

8.2.4.1 Comparison of conflict costs

The costs of the initial plan can be found in Table 8.2.6.

Table 8.2.6: Overview of the average costs (over 18 runs) of the initial plan constructed with the old and new set of algorithms.

	avg. initial costs (old)	avg. initial costs (new)	improvement (%)
Amersfoort	4093.7	3658.9	10.6
Enkhuizen	4469.0	4444.1	0.6
Enschede	2074.3	1869.0	9.9
Rotterdam 1	1707.5	1159.4	32.1
Rotterdam 2	4264.9	2019.6	52.6
Utrecht 1	36427.5	8751.4	76.0
Utrecht 2	52346.7	17599.7	66.4
Utrecht 3	37296.4	16133.2	56.7

There is an average improvement across all scenarios, although for Enkhuizen this improvement is not statistically significant. This indicates that the local search solver is able to use the results of our algorithms for producing better initial plans. The differences for the Rotterdam and Utrecht scenarios are large (although the significance for Rotterdam 2 will not persist over time as we shall see later). This is most likely explained by the fact that these are the scenarios with multiple yards. This is a satisfying result, since the goal of the algorithmic development was to improve the initial plan for large stations, such as Utrecht. It means that the choices made in the algorithm for choice of yard and stabling track are well-made.

From experience, it is known that better initial solutions do not necessarily lead to better solutions after the local search has run for some time. Therefore, the final conflict costs after the chosen run times are compared in Table 8.2.7.

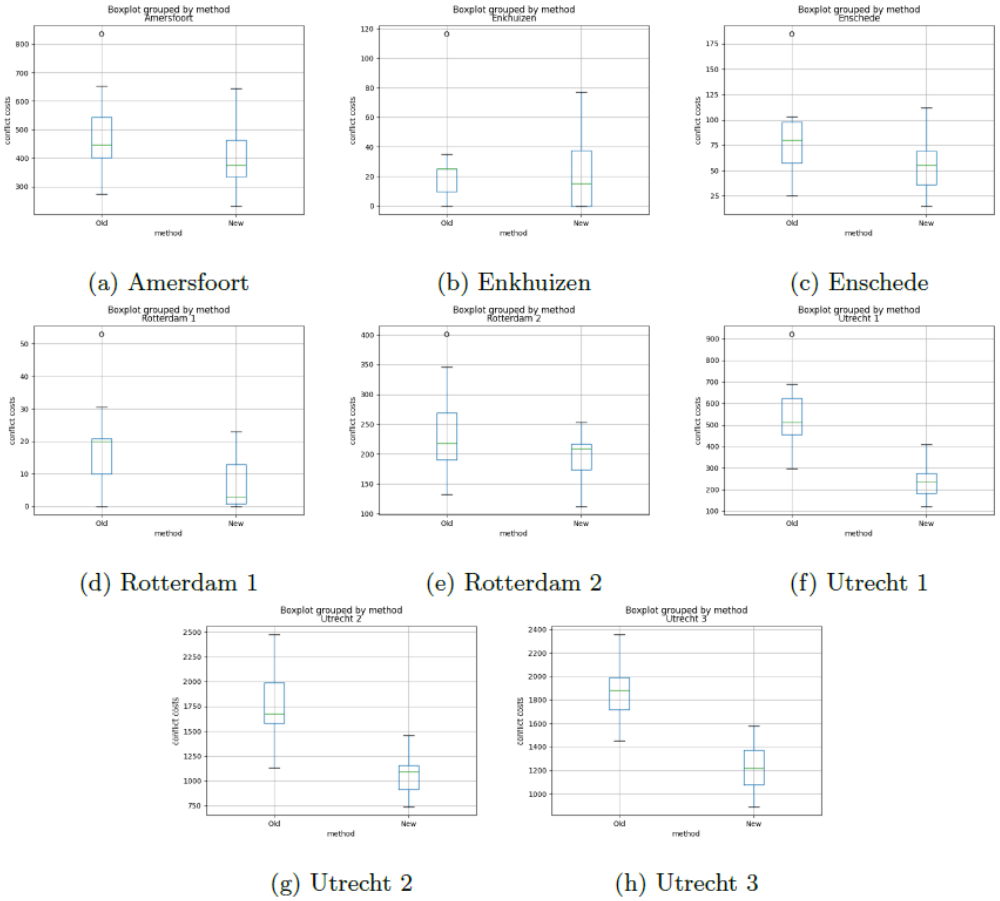
Table 8.2.7: Overview of costs of the final plan constructed with the old and new set of algorithms, including the number of solved instances (over 18 runs).

scenario	avg.costs (old)	avg.costs (new)	impr. (%)	#solved (old)	#solved (new)
Amersfoort	478.2	398.2	16.7	0	0
Enkhuizen	23.2	22.3	4.2	3	8
Enschede	81.5	53.1	34.8	0	0
Rotterdam 1	18.1	7.7	57.3	3	5
Rotterdam 2	229.9	196.2	14.7	0	0
Utrecht 1	538.8	247.1	54.1	0	0
Utrecht 2	1742.9	1055.7	39.4	0	0
Utrecht 3	1846.7	1205.2	34.7	0	0

Again, the new initial plan leads to an improvement across all scenarios, of which that of Amersfoort, Enschede and Rotterdam 1 and Utrecht are statistically significant. This means that the local search is well able to deal with the initial plan provided by our new set of algorithms. The improvements for the Rotterdam and Utrecht scenarios are again large, showing that local search performs well with the newly created initial plan.

The differences for Enkhuizen are small, because in both the old and new situation the local search solver is able to get close to a conflict-free solution and only small differences in conflict costs are expected. However, the new method is able to solve more instances within the allotted time (8 vs. 3).

All this is more or less confirmed by the boxplots in Figure 8.2.2. The boxplots also show that the spread of the conflict costs of the new approach is much larger for the smaller scenarios (Amersfoort, Enkhuizen, Enschede). No reason for this has been found so far, but it appears that sometimes the choices of the new algorithms turn out to have a poor effect on the local search. Its effect on the operational usability of the solver depends on the way it is going to be used. If the local search solver is run on the same scenario multiple times simultaneously producing several plans of which one is going to be selected, the outliers of large conflict costs have little effect.



(a) Amersfoort (b) Enkhuizen (c) Enschede

(d) Rotterdam 1 (e) Rotterdam 2 (f) Utrecht 1

(g) Utrecht 2 (h) Utrecht 3

Figure 8.2.2: Boxplots showing, per scenario, the distribution of final costs for the short runs.

Finally, the conflict costs of Utrecht 1 are considerably lower than those of Utrecht 2 and Utrecht 3. The solution of Utrecht 1 is even close to the level that a human planner will be able to solve the remaining conflicts, which is a situation that has never been achieved before with the old approach. Unfortunately, this is not consistent behavior as Utrecht 2 and Utrecht 3 show much higher costs. No reason has been found so far, but it is still under investigation why this is the case. The time needed to find the initial solution is found in Table 8.2.8.

Table 8.2.8: Overview of the run times for the creation of the initial plan constructed with the old and new set of algorithms, (over 18 runs).

	avg. run time initial (old) (min)	avg. run time initial (new) (min)
Amersfoort	0.1	1.0
Enkhuizen	0.0	0.2
Enschede	0.0	0.4
Rotterdam 1	0.1	0.1
Rotterdam 2	0.1	0.3

	avg. run time initial (old) (min)	avg. run time initial (new) (min)
Utrecht 1	0.7	2.2
Utrecht 2	2.2	10.1
Utrecht 3	1.2	2.2

It shows that the new approach for the creation of the initial plan takes considerably more time than the old approach, but the results in Table 8.2.7 show that it is well-spent time, since the local search performs much better. Also, the initial plan creation for Utrecht is in the order of minutes, which is negligible compared to the required computation time. The extra computation time for the initial plan due to the more complex algorithms, is therefore no issue.

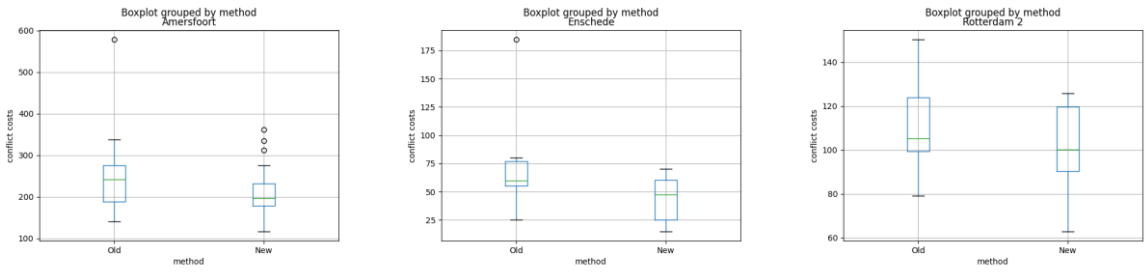
8.2.4.2 Comparison of conflict costs (extended runs)

The comparison of the costs of the final plan for the extended runs can be found in Table 8.2.9. These have only been carried out for the other scenarios than Utrecht, since for Utrecht, there is little to no progress made if the local search runs longer than 10 hours.

Table 8.2.9: Overview of average costs of the final plan (over 18 runs) constructed with the old and new set of algorithms after long runs of the local search.

scenario	avg.costs (old)	avg.costs (new)	impr. (%)	#solved (old)	#solved (new)
Amersfoort	256.7	215.2	16.2	0	0
Enkhuizen	3.4	1.2	66.3	15	16
Enschede	68.2	44.6	34.6	0	0
Rotterdam 1	1.1	1.7	-51.8	16	16
Rotterdam 2	110.1	99.7	9.5	0	0

The percentual improvement of the final costs are similar to those of the short run for the scenarios Amersfoort, Enschede and Rotterdam 2. Those of Enkhuizen and Rotterdam 1 are quite different, but these numbers in Table 8.2.9 are not representative because those are based on the values of the 2-3 unsolved instances. The improvement for Enschede is still statistically significant, whereas the improvement for Amersfoort and Rotterdam 2 are not. The boxplots in Figure 8.2.3 seem to support these observations. For instance, for Amersfoort and Rotterdam, one can indeed see an improvement in costs, but the boxplots have a large overlap indicating that the improvement is probably insignificant.



(a) Amersfoort *(b) Enschede* *(c) Rotterdam 2*

Figure 8.2.3: Boxplots showing, per scenario, the distribution of final costs for the long runs.

It is not at all unexpected that the improvement becomes insignificant after longer computation times. Because it takes more time to find the next improvement when the conflict costs are low, the difference between the old and new approach gets smaller over time. The advantage of the new set of algorithms is that it finds better solutions in less time. This is illustrated in Figure 8.2.4.

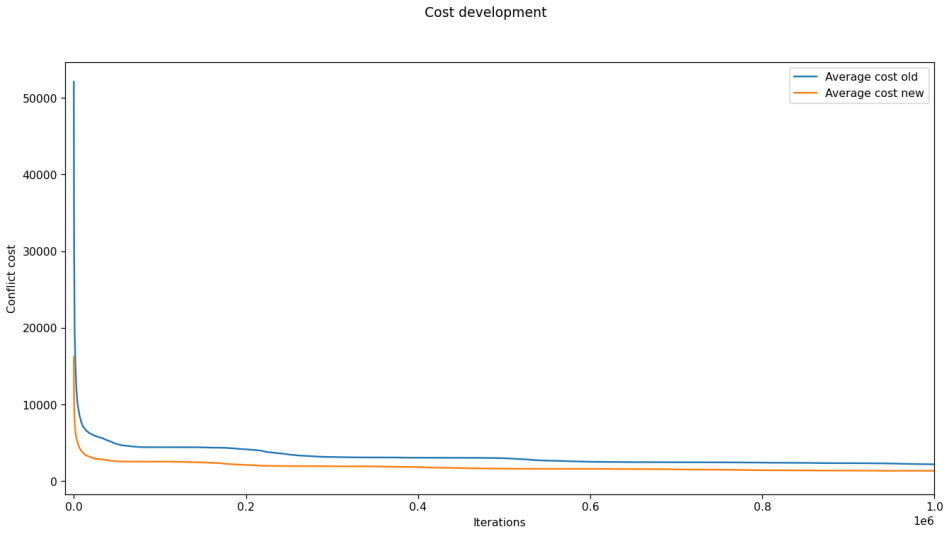


Figure 8.2.4: Comparison of the average of the cost development of Utrecht 2. The new set of algorithms lead to a quicker drop of the conflict costs and drops to a lower value. At some point, the cost development drops more slowly, and the conflict cost resulting from the old set of algorithms comes closer to but never gets below the costs resulting from the new approach.

Restricting our attention to the scenarios that actually get solved (Enkhuizen and Rotterdam 1), the number of solved instances are (almost) the same for both the old and new version.

Table 8.2.10: Overview of the average run time for finding a conflict-free plan with the old and new set of algorithms after extended runs of the local search (18 runs per scenario).

	avg. run time (old) (min)	avg. run time (new) (min)	impr. (%)
Enkhuizen	5.9	4.5	22.6
Rotterdam 1	15.4	9.1	41.1

However, Table 8.2.10 shows that the computation time for finding a conflict-free solution is on average reduced drastically for both Enkhuizen and Rotterdam 1, where the percentual improvement is larger for Rotterdam 1. This is probably explained by the fact that Rotterdam is larger than Enkhuizen and has multiple yards.

8.2.5 Conclusion

In conclusion, the presented algorithms for creating initial solutions for the local search solver show promising results. The presented algorithms outperform the previous method on most tested scenarios, reducing the conflict cost of the initial solution for almost all scenarios. The performance improvements are especially noticeable for the larger scenarios, such as Rotterdam and Utrecht. For these scenarios large reductions in conflict costs were found, reducing the initial costs by at least 30% on average, and more than 60% on average for some scenarios.

The new algorithms do, however, require larger computation times than the previous method. Nonetheless, the new algorithms also seem to perform well together with the local search solver, even though the local search was run for a shorter amount of time (due to the shared time limit with the initial solution). For most scenarios not yet solved in the allotted time limit, the remaining conflict costs were reduced considerably, while also reducing the average time required to solve the scenarios which were solved within the time limit by up to 40%. For the other scenarios, one can not conclusively state that the algorithms outperform the previous method, although an

improvement in conflict costs was still shown on average. Including the local search, the largest scenarios of Utrecht again show the biggest improvements.

The new algorithms do, however, seem to have a higher variation in costs between the runs compared to the old method for some scenarios.

8.2.6 Discussion

The shown methods show promising results, however, some aspects still need to be addressed. For example, for some scenarios the improvements were less noticeable then others, which warrants further investigation. Furthermore, on multiple scenarios a larger variation in the final costs was observed, which could be looked into further.

The effect of the settings for the different algorithms was not considered in this work. Experiments regarding these settings should be performed in future work, as it is believed that these settings could have a large impact on the final quality of the found solutions.

The alternative modeling of the stabling algorithm proposed in Section 2.3 proved to be too slow during preliminary testing and could be improved further. No experiments were carried out for that reason. However, during development of the models it was noticed that some settings of the CP-SAT solver had major impact on the computation times, which is something that might still benefit this modeling approach.

Lastly, the created plans were only compared based on the objective function in the local search solver. In future work this should also be looked at by human planners, to make sure that the created plans are sound.

9. Interaction with other Flagship Projects

No interactions with other FPs are considered in WP6.

10. Conclusions

In this deliverable we have described the outcomes of the research conducted in WP6. WP6 main tasks was to develop computer algorithms to find optimal or near-optimal

- Long term train timetables (Task 6.2)
- Short term train timetables (Task 6.3)
- Rolling stock plans, including rolling stock rotation, stabling and shunting of train units. (Task 6.4).

The quality (optimality) of the solutions is determined by suitable objectives or KPIs. Examples: for long-term timetabling we can measure the number of trains we are able to plan or the demand of traffic we can serve; For short term timetabling, typically one measures the deviation from the original wanted timetable; For rolling stock, the number of train units necessary to fulfil the plan; For stabling and shunting, the deviation from wanted departure times of train units.
Several innovative approaches were developed by the partners, also in order to solve specific versions associated with individual use-cases. In most cases, the problems were modelled as mixed integer optimization problems (MILPs) and solved by various techniques, including mathematical decomposition, mathematical programming, constraint programming, network optimization algorithms, genetic algorithms, local search, simulated annealing, and other ad-hoc approaches.
More specifically:

In 6.1, we developed an iterative algorithm to find high-quality solutions for the passenger-oriented timetabling problem where the perceived travel time of the passenger is to be minimised. Our approach combines an Adaptive Large Neighbourhood Search algorithm with a MIP solver. We evaluate our methods on problem instances of NSR.

In 6.2, we considered a new metric and exact solution for the tactical robust station routing problem: Using a MILP approach, we determine an optimal local routing that minimises the expected delay caused by isolated primary delays while ensuring feasibility and consistency with a network timetable. Furthermore, a potential application to evaluate and compare infrastructure variants in the strategic station planning process was discussed.

In 6.3, we presented a method to analyse capacity utilisation in strategic planning when building or rebuilding stations, which can also be applied to analyse capacity utilisation of different timetable alternatives. The method includes an analysis of the number of total possible passenger connections, train changes at the same platform, and crossing train paths.

In 6.4, we developed an innovative algorithm for automatic conflict detection and resolution with complex routing that takes inspiration from the conflict-based search framework used by the Multi-agent Pathfinding community. The algorithm is based on a time-indexed formulation, which can have several advantages when trains are dealing with numerous routing choices (usually freight trains).

In 6.5, we iteratively designed an HMI to visualize timetable quality metrics, aimed at supporting long-term timetable planners in working with optimization-based timetables. The HMI is grounded in user-centered design principles and tailored to reflect and support the planners' daily workflows. Introducing optimization-based support has the potential to shift planners' roles from manually entering detailed planning data towards controlling algorithms, enhancing efficiency and providing more effective tools for comparing and refining timetable solutions.

In 7.1, we developed an extension of state-of-the-art algorithms for disjunctive graph formulations. The application if the Genoa node, a complex piece of network in Italy with a combination of double- and single-track lines plus routing possibilities. The algorithm is intended for dealing with Temporary Capacity Restrictions (TCRs), and exploits the concepts of incremental timetabling, which tries to reconstruct a new feasible timetable that is as close as possible to a reference one. The MILP-based algorithm has been

significantly improved from previous implementations (which were already at the state-of-the-art), and now supports exact station platforming, alternative routes, and “lazy modelling” (i.e., trains are modelled only if needed).

In 7.2, we have shown the suitability of an incremental timetabling algorithm to adjust a timetable during a TCR. Our experiments are based on real data from the line from Gothenburg to the Swedish-Norwegian border. The timetable of one running day is considered in the test case, and it contains mixed traffic of passenger, freight and service trains with a total of 233 trains. The TCR considered is the closing of one track on an 8 km double track segment between two stations and a reduction of the speed limit on the remaining track. The results show that the algorithm seems very promising when replanning for this type of TCR.

In 7.3, we proposed an iterative approach to adjust a periodic (1-hour) timetable for planned infrastructure maintenance possessions. Our method combines a network-level macroscopic timetable optimization model with a station-level macroscopic track usage model. We demonstrate the power of the proposed approach by solving realistic large-scale problem instances, consisting of the entire network of NSR as well as of all freight services.

In 7.4, we presented the first simulation-based time-expanded graph approach, developed by the French infrastructure manager (SNCF Réseau), that solves the problem of residual capacity management. As this approach only takes into consideration the state of the existing timetable, we proposed a track reallocation algorithm to unlock more short-term train paths. Early experimental results on toy instances concluded on the algorithm’s potential efficiency in improving the performances of the first approach.

In 7.5 we developed a new line-station decomposition method for solving the (Short-term) Train Timetabling problem. We also introduced a new ILP formulation for suitably designing the (extended) stations to be used in such decomposition.

In 7.6, a Genetic Algorithm for decision support in timetable optimization is developed to address conflicts in railway schedules. The algorithm uses genetic techniques, such as crossover and mutation, to explore potential solutions for minimizing delays and optimizing operational efficiency. While the GA significantly reduces computational time and memory usage, it does not guarantee an optimal solution after a single iteration. The effectiveness of the approach improves through iterative applications, particularly in high-density, complex networks. By testing the algorithm in realistic scenarios, it shows promise as a practical tool for timetable optimization, although further refinement may be needed for full operational deployment.

In 8.1, we developed an integer multicommodity flow model to compute optimal rolling stock rotation for a given timetable, with (main) objective to minimize the number of train units necessary to fulfil the timetable.

In 8.2, we developed a Constraint Programming model for the routing and timing of shunt movements to shunt yards through the passing traffic in the station area, and another constraint programming model for the parking of shunt trains on the yards themselves with objective the number of blocked shunt trains. The results of both models are used to build an initial solution of much higher quality (compared to the current method) for an already existing local search algorithm.

Experiments carried out on artificial but realistic instances show that the methodologies are expected to perform well on the designed use-cases, which in turn are representative of real-life scenarios.
Difficulties may pop-up when scaling up to large networks or solving the individual problems as one problem: for instance, by combining long term timetabling and rolling stock rotation.
In any case, the work is not concluded, as the algorithms will still be developed during the first phase of next work-package WP7. More complex problems, or larger instances, may be the subject of the studies in a next wave.

11. References

Acuna-Agost, R., & Cordeau, J.-F. (2011). A Genetic Algorithm-Based Approach for Railway Traffic Management. Computers & Operations Research, 38(1), 92-101.

Alfieri, A., Groot, R., Kroon, L., Schrijver, A. Efficient circulation of railway rolling stock (2006). *Transportation Science* 40(3), pp. 378-391.

Bach, L., Mannino, C., Sartor, G. MILP approaches to practical real-time train scheduling: the Iron Ore Line case (2019). *INOC*, pp. 78-82.

Bendfeldt, J.P., Mohr, U., Müller, L. RailSys, a system to plan future railway needs (2000). *Computers in Railways* VII 50, pp. 249-255.

Borndörfer, R., Klug, T., Schlechte, T., Fügenschuh, A., Schang, T., and Schülldorf, H. The freight train routing problem for congested railway networks with mixed traffic (2016). *Transportation Science*, 50(2), pp. 408–423.

Borndörfer, R. ,Schlechte, T. Models for railway track allocation (2007). *ATMOS 2007—7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*

Broman, E., Eliasson, J., Aronsson, M. Efficient capacity allocation on deregulated railway markets (2022) . *Journal of Railway Transport Planning & Management* 21.

Brown, W., O'Hara, J., and Higgins, J. Advanced Alarm Systems: Guidance Development and Technical Basis (NUREG/CR-6684) (2000). Washington, DC: U.S. Nuclear Regulatory Commission.

Burdett, R. L. and Kozan, E. Techniques for inserting additional trains into existing timetables (2009). *Transportation Research Part B*, 43(8-9), pp. 821–836.

Burggraeve, S., Vansteenwegen, P. Robust routing and timetabling in complex railway stations (2017). *Transportation Research Part B* 101, pp. 228–244.

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

Cacchiani, V., Caprara, A., Toth, P. Solving a real-world train-unit assignment problem (2010). *Mathematical programming* 124, pp. 207-231.

Cacchiani, V., Caprara, A., Toth, P. A Lagrangian heuristic for a train-unit assignment problem (2013). *Discrete Applied Mathematics* 161(12), pp. 1707-1718.

Cacchiani, V., Toth P. Robust train timetabling (2018). In: *Handbook of Optimization in the Railway Industry*, pp. 93–115.

Caimi, G., Chudak, F., Fuchsberger, M., Laumanns, M. A New Resource-Constrained Multicommodity Flow Model for Conflict-Free Train Routing and Scheduling (2011). *Transportation Science* 45 (2), pp. 212-227.

Caimi, G., Kroon, L., Liebchen C. Models for railway timetable optimization: Applicability and applications in practice (2017). Journal of Rail Transport Planning & Management 6 (4), pp. 285–312.

Caprara, A., Galli, L., Stiller, S., Toth, P. Delay-Robust Event Scheduling (2014). *Operations Research* 62 (2), pp. 274–283.

Caprara, A., Galli, L., Toth, P. Solution of the train platforming problem (2011). *Transportation Science* 45(2) , pp. 246–257.

Caprara, A., Galli, L., Kroon, L., Maróti, G., Toth, P. Robust train routing and online re-scheduling (2010). *10th Workshop on algorithmic approaches for transportation modelling, optimization, and systems (ATMOS'10).*

Castillo, E. et al. Timetabling optimization of a single railway track line with sensitivity analysis (2009). *Top* 17, pp. 256–287.

Charpentier, E. Automated short-term train planning in OSRD (2023). Presented at the Free and Open Source Software Developers' European Meeting, Brussels, Belgium. Conference presentation.

Dahms, F.H.W., Frank A.-L., Kuehn S., & Poehle, D. Transforming Automatic Scheduling in a Working Application for a Railway Infrastructure Manager (2019). In 8th International Conference on Railway Operations Modelling and Analysis, Norrkoeping, Sweden.

D'Ariano, A., Corman, F., & Pacciarelli, D. (2010). Real-Time Railway Traffic Management Using Evolutionary Algorithms. IEEE Transactions on Intelligent Transportation Systems, 11(1), 1-11.

D'Ariano, A., Pranzo, M. An advanced real-time train dispatching system for minimizing the propagation of delays in a dispatching area under severe disturbances (2009). *Networks and Spatial Economics* 9, pp.63-84.

D'Ariano, A., Pacciarelli, D., Pranzo, M. A branch and bound algorithm for scheduling trains in a railway network (2007). *European Jounral of Operational. Research* 183(2), pp. 643–657.

DB InfraGO Konzernrichtlinie 808 "Kostenkennwertekatalog" (2016).

Delorme, X., Gandibleux, X., Rodriguez, J. Stability evaluation of a railway timetable at station level (2009). *European Journal of Operational Research* 195 (3), pp. 780–790.

Dewilde, T., Sels, P., Cattrysse, D., Vansteenwegen, P. Improving the robustness in railway station areas (2014). *European Journal of Operational Research* 235 (1), pp. 276–286.

Eckert, A. und Brinkmann, F. und Scheier, B. Kostenvergleich einer innovativen Zugvollständigkeitskontrolle / A cost comparison of innovative train integrity control (2020). *SIGNAL + DRAHT, 12/20* (112), pp. 52-58. DVV Media Group. ISSN 0037-4997.

European Commission. Guide to Cost-Benefit Analysis of Investment Projects: Economic appraisal tool for Cohesion Policy 2014-2020 (2014). ISBN 978-92-79-34796-2.

Europe's Rail (2022). Multi-Annual Work Program. www.rail-research.europa.eu/wp-content/uploads/2022/03/EURAIL_MAWP_final.pdf

Europe's Rail (2022). FP1-MOTIONAL, GRANT AGREEMENT, Project 101101973. https://projects.rail-research.europa.eu/eurail-fp1/

Europe's Rail (2022). FP1-MOTIONAL, Deliverable 3.1, Mapping against scope, specification of technical enablers, high-level use cases, high-level requirements, high level design for demonstrators in WPs 4-9. **https://www.ct5webapi.eu/api/documents/getfile?id=19bb4dd5-6a07-4abe-b714-1038f4cfa521**

Flier, H., Graffagnino, T., and Nunkesser, M. Scheduling additional trains on dense corridors 2009). In *International Symposium on Experimental Algorithms*, pp. 149–160. Springer.

Gao, Y., Kroon, L., Yang, L., and Gao, Z. Three-stage optimization method for the problem of scheduling additional trains on a high-speed rail corridor (2018). *Omega* 80, pp.175–191.

Gestrelius, S., Bohlin, M., & Aronsson, M. On the uniqueness of operation days and delivery commitment generation for train timetables (2015). In 6th International Conference on Railway

Operations Modelling and Analysis (RailTokyo2015), March 23-26, 2015, Tokyo, Japan.

Gestrelius, S., Aronsson, M., & Peterson, A. A MILP-based heuristic for a commercial train timetabling problem (2017). *Transportation Research Procedia*, *27*, 569-576.

Gestrelius, S., Joborn, M., & Ranjbar, Z. Flexible trains in timetabled traffic (2023). Presented at 10th International Conference on Railway Operations Modelling and Analysis (RailBelgrade2023).

Grimm, B., Hoogervorst, R., Borndörfer, R. A Comparison of Models for Rolling Stock Scheduling (2023). arXiv preprint arXiv:2312.09697.

Haehn, R., Ábrahám, E., and Nießen, N. Freight train scheduling in railway systems (2020). In *Measurement, Modelling and Evaluation of Computing Systems: 20th International GI/ITG Conference, MMB 2020, Saarbrücken, Germany, March 16–18, 2020, Proceedings 20*, pp. 225–241. Springer.

Harrod, S. Modeling network transition constraints with hypergraphs (2011). *Transportation Science* 45(1), pp. 81–97.

Hoogervorst, R., Dollevoet, T., Maróti, G., Huisman, D. A Variable Neighborhood Search heuristic for rolling stock rescheduling (2021).. *EURO Journal on transportation and logistics*, 10.

Jiang, Z., Tan, Y., Yalçınkaya, O., et al. . Scheduling additional train unit services on rail transit lines (2014). *Mathematical Problems in Engineering*.

Johansson, I. & Peterson, A. (2024). Rail Platform Allocation for Reliable Interchanges. *Transportation Research Procedia*, vol. 15, pp. 198-205.

Johansson, I., Weik, N. Strategic assessment of railway station capacity - Further development of a UIC 406-based approach considering timetable uncertainty (2021). Presented at the 9th International Conference on Railway Operations Modelling and Analysis, Beijing, China.

Kloster, O., Luteberget, B., Mannino, C., Sartor, G. An optimization-based decision support tool for incremental train timetabling (2023). *Operations Research Forum* 4 (3), p.65.

Lamorgese, L., Mannino, C. An exact decomposition approach for the real-time train dispatching problem (2015). *Operations Research* 63 (1), pp. 48-64.

Lamorgese, L., Mannino, C. A noncompact formulation for job-shop scheduling problems in traffic management (2019). *Operations Research* 67(6), pp.1586-1609.

Lamorgese, L., Mannino, C., Natvig E. An exact micro– macro approach to cyclic and non-cyclic train timetabling (2017). *Omega* 72, pp. 59–70.

Lantz, A. (2013). *Intervjumetodik*. Studentlitteratur.

Leutwiler, F., Corman, F. A logic-based Benders decomposition for microscopic railway timetable planning (2022). *European Journal of Operational Research* 303 (2), pp. 525–540.

Lidén, T. Railway infrastructure maintenance-a survey of planning problems and conducted research (2015). *Transportation Research Procedia*, *10*, 574-583.

Ljunggren, F., Persson, K., Peterson, A., and Schmidt, C. Railway timetabling: a maximum bottleneck path algorithm for finding an additional train path (2021). *Public Transport* 13, pp. 597–623.

Lusby, R.M. et al. (2011). “Railway track allocation: models and methods”. *OR spectrum* 33 (4), pp. 843–883.

Mascis, A., Pacciarelli, D. Job-shop scheduling with blocking and no-wait constraints (2002).

European Journal of Operational Research 143 (3), pp. 498–517.

Mannino, C., Nakkerud, A. Optimal Train Rescheduling in Oslo Central Station (2023). *Omega* 116, p.102796.

Nachtigall, K. Voget, S. A genetic algorithm approach to periodic railway synchronization (1996). Computers & Operations Research 23 (5), pp. 453–463.

Nellthorp, J. The principles behind transport appraisal (2017). *The Routledge Handbook of Transport Economics*, pp. 176–208.

Nielsen, L.K., Kroon, L., Maróti, G. A rolling horizon approach for disruption management of railway rolling stock (2012). *European Journal of Operational Research* 220(2), pp.496-509.

Pellegrini, P. et al. (2015). "RECIFE-MILP: An Effective MILP-Based Heuristic for the Real-Time Railway Traffic Management Problem". *IEEE Trans. Intell. Transp. Syst.* 16 (5), pp. 2609–2619.

Phillips, M., Likhachev, M. SIPP: Safe interval path planning for dynamic environments (2011). *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*. IEEE, pp. 5628–5635.

Polinder, G.J., Schmidt, M., Huisman, D. Timetabling for strategic passenger railway planning (2021). *Transportation Research Part B* 146, pp. 111–135.

Reynolds, E. et al. A multicommodity flow model for rerouting and retiming trains in real-time to reduce reactionary delay in complex station areas (2020). *Optimization Online*.

Samà, M., D'Ariano, A., Pacciarelli, D. Rolling horizon approach for aircraft scheduling in the terminal control area of busy airports (2013). *Procedia-Social and Behavioral Sciences* 80, pp.531-552.

Sartor, G., Mannino, C., Nygreen, T., Bach, L. A MILP model for quasi-periodic strategic train timetabling (2023). *Omega* 116, p.102798.

Schlechte, T. Borndörfer, R. Balancing efficiency and robustness–a bi-criteria optimization approach to railway track allocation (2010). In: Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems. Springer, pp. 105–116.

Sharon, G. et al. (2015). "Conflict-based search for optimal multi-agent pathfinding". In: *Artif. Intell.* 219, pp. 40–66.

Stanton, N. A. Hierarchical task analysis: Developments, applications, and extensions (2006). *Applied ergonomics*, 37(1), 55-79.

Stern, R. et al. (2019). "Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks". In: *Proceedings of the Twelfth International Symposium on Combinatorial Search, SOCS 2019, Napa, California, 16-17 July 2019*. Ed. by Pavel Surynek and William Yeoh. AAAI Press, pp. 151–158.

Swedish Transport Administration. Network statement 2024, Edition 2024-06-26 For deliveries from 2023-12-10 to 2024-12-14. https://bransch.trafikverket.se/en/startpage/operations/Operations-railway/Network-Statement/network-statement-2024/. 2023.

Tan, Y., Jiang, Z., et al. A branch and bound algorithm and iterative reordering strategies for inserting additional trains in real time: A case study in Germany (2015). *Mathematical Problems in Engineering*.

Tan, Y.-Y., Jiang, Z.-B., Li, Y.-X., Wang, R.-X. Integration of train-set circulation and adding train paths problem based on an existing cyclic timetable (2020). *IEEE Access* 8, pp. 87142–87163.

Trafikverket. (2024). Halmstad C, bangårdsombyggnad (in Swedish). URL: https://www.trafikverket.se/vara-projekt/projekt-i-hallands-lan/halmstad-vaxer/halmstad-c-bangardsombyggnad/. Last accessed: 2024-04-25.

UIC. (2013). Code 406 - capacity. International Union of Railways, 2nd edition, Paris, France.

Van Aken, S., Bešinović, N., Goverde, R. M. Designing alternative railway timetables under infrastructure maintenance possessions (2017a). *Transportation Research Part B* 98, pp. 224–238.

Van Aken, S., Bešinović, N., Goverde, R. M. Solving large-scale train timetable adjustment problems under infrastructure maintenance possessions (2017b). *Journal of Rail Transport Planning & Management* 7(3), pp.141–156.

Van den Broek, R., Hoogeveen H., Van den Akker, M., Huisman, B. A local search algorithm for train unit shunting with service scheduling (2021). *Transportation Science* 56(1), pp.141–161.

Van den Broek, R. Towards a Robust Planning of Train Shunting and Servicing (2022). PhD thesis, University of Utrecht.

Van Lieshout, R. N. Integrated periodic timetabling and vehicle circulation scheduling (2021). *Transportation Science* 55(3), pp. 768–790.

Weik, N., Warg, J., Johansson, I., Bohlin, M., Nießen, N. Extending UIC 406-based capacity analysis – New approaches for railway nodes and network effects (2020). *Journal of Rail Transport Planning & Management* 15, pp. 100199.

Widmann, P., Weik, N. Robust routing in railway stations - Generalizing and assessing robustness concepts using absorption graphs (2024). [Manuscript submitted for publication; available at SSRN: https://ssrn.com/abstract=5000863]

Zhan, S., Kroon, L.G., Zhao, J., Peng, Q. A rolling horizon approach to the high speed train rescheduling problem in case of a partial segment blockage (2016). *Transportation Research Part E: Logistics and Transportation Review* 95, pp.32-61.

Zwaneveld, P. J., Kroon, L.G., Van Hoesel S.P.M. Routing trains through a railway station based on a node packing model (2001). *European Journal of Operational Research* 128 (1), pp. 14–33.

12. Appendices

12.1. Appendix A

It provides additional information to Section 5.

50 years of OR in Railway Planning

Pedro José Correia Duarte[1], Stéphane Dauzère-Pérès[3,7], Dennis Huisman[1,2], Carlo Mannino[4], Giorgio Sartor[4], Norman Weik[5], Philipp Widmann[6]

[1]Econometric Institute and Erasmus Center for Optimization in Public Transport
Erasmus University Rotterdam, The Netherlands

[2]Process quality and Innovation, Netherlands Railways
Utrecht, The Netherlands

[3]Mines Saint-Etienne, Univ. Clermont Auvergne, CNRS, UMR 6158 LIMOS
CMP, Department of Manufacturing Sciences and Logistics, Gardanne, France

[4]SINTEF Digital, Department of Mathematics and Cybernetics
Forskningsveien 1, 0373, Oslo, Norway

[5]Technical University of Munich, TUM School of Engineering and Design
Professorship for Design and Operation of Public Rail Transport Systems (RTS)
Parkring 35, 85748, Garching, Germany

[6]German Aerospace Center (DLR), Institute of Transportation Systems
Lilienthalplatz 7, 38108, Braunschweig, Germany

[7]Department of Accounting and Operations Management
BI Norwegian Business School, Oslo, Norway

correiaduarte@ese.eur.nl, dauzere-peres@emse.fr, huisman@ese.eur.nl, carlo.mannino@sintef.no, giorgio.sartor@sintef.no, norman.weik@tum.de, philipp.widmann@dlr.de

Abstract

This survey paper discusses the literature on Operations Research (OR) models and algorithms for railway planning in the last decades. Since infrastructure and rolling stock are two resources that are both very capital-intensive and characterize the railway system, we focus on timetabling and rolling stock planning problems. For timetabling, we also classify the literature among two dimensions, namely the decision level (strategic, tactical, operational) and the type of network infrastructure (dependent routes, independent routes). We also discuss robustness aspects in both planning problems.

We focus the discussion of the literature on the applicability of the models in the European context, where different types of trains (high-speed passenger trains, high-frequent suburban trains and freight trains) often share the same tracks and the organization of the railways is usually split between an infrastructure manager and one or more railway undertakings operating the trains.

We conclude the paper with some challenges and future research directions.

1 Introduction

Although the title of this article might suggest that Operations Research (OR) models and algorithms for the railway domain were only developed in the last 50 years, the first applications appeared much earlier. Schrijver (2002) was able to discover that the first paper discussing a transportation problem for a railway application dates back to 1939 (Tolstoı, 1939). Interestingly, a few years later, another railway application appeared in the paper that first described the minimum cut problem. Harris and Ross (1955) study the railway network in the former Soviet Union and affiliated Eastern European countries in a secret report written for the US Air Force in 1955 (downgraded as unclassified in 1999). The authors calculate the maximum flow through

the network from origins in the Soviet Union to destinations in Eastern Europe, and then use that to find "the bottleneck" (nowadays known as the minimum cut). With this work, they laid the foundation for the famous paper of Ford and Fulkerson (1956).

In this survey paper, we focus on several (more recent) OR problems arising in railway planning. The railway domain is rich of challenging optimization problems that served as inspiration to many OR researchers. Good examples are the Edelman finalists Canadian Pacific Railway (CP) (Ireland et al., 2004), Netherlands Railways (NS) (Kroon et al., 2009) and Deutsche Bahn (DB) (Borndörfer et al., 2021), who all developed and applied sophisticated OR methods to solve real-world railway applications. With this paper, we want to review the basic models and the most important literature in the last decades of timetabling and rolling stock planning. We limit ourselves to these problems, because both topics use the two most capital-intensive resources to operate a railway system, namely infrastructure and rolling stock, and are both characteristic for railway operations. In addition, the building of infrastructure has a lead time of several decades, while the acquisition of rolling stock usually takes 5-10 years. Therefore, it is important to use these resources in an efficient way. Timetabling and rolling stock planning are both studied in the EU-Rail flagship project, MOTIONAL, funded by the European Union and the European railway sector, since there are also still many challenges for the future.

In this paper, we focus on the application of OR models to European railways. In Europe, long distance (often high-speed) passenger trains, regional/suburban passenger trains and freight trains often use the same railway infrastructure. Moreover, in most countries infrastructure management and railway operations are split in different organizations. As a result, there are different challenges in Europe than in North-America or Asia. Therefore, we focus on Operations Research models for European railway planning. For global surveys on railway planning, we refer to Harrod (2012). Other relevant topics in the railway domain are railway crew scheduling and disruption management. For these topics, we refer to recent overview papers by Heil et al. (2020) and Cacchiani et al. (2014), respectively.

1.1 Discussion on classifying the timetabling literature

Given a line plan and an infrastructure network, the *Train Timetabling Problem (TTP)* consists of finding a feasible schedule for the arrival and departure of trains at stations. The line plan is defined as the set of lines that all railway undertakings want to operate. For a passenger operator, a line is defined as a list of stations where the train is supposed to stop. For a freight operator, a line is defined simply by the origin and the destination terminal. Each line also has a certain frequency (e.g., twice per hour).

Railway timetabling can be classified in several ways. We distinguish between timetabling at the three classical decision levels (strategic, tactical and operational) and by the type of network infrastructure (independent routes versus dependent routes).

Regarding the decisions levels, **strategic** planning deals with decisions about the design of the railway infrastructure, both the network topology (e.g., a new track in a station) as well as technical systems (e.g., a new safety system). This determines the capacity of a railway network. Given the railway network and given the line plan, **tactical** timetable planning consists of constructing the yearly timetable. This is done once a year and constitutes the basis for the next level of planning. By **operational** planning we define the process of adjusting the current yearly timetable to create a new (possibly temporary) timetable that takes into account additional needs and constraints. This process can last from 1 day to 12 months ahead of the day of operations for the respective train. For instance, one may want to temporarily adjust the current timetable to account for maintenance activities which will block some tracks for a week. Or, one may want to add a special train to perform some extra service. We assume that the additional needs and constraints that are planned for *do not derive* from some unforeseen disruptive event (like a track failure) or from a primary train daily. These cases are handled by other railway processes, like *disruption management* or standard *dispatching*.

Regarding the way the railway network is modeled in the timetabling literature, either a single **independent** route is considered or multiple **dependent** routes are considered simultaneously. In the first case, the timetable of the route is assumed to be independent of the timetable of the other routes. This is for instance the case of dedicated high-speed railway lines. However, complex railway networks like the ones in Germany, the Netherlands or Switzerland have a strong inter-dependency between different routes, for instance, to ensure connections between each pair of trains at a major station. For example, at Zurich main station all

trains arrive a few minutes before minute .00 and .30 and all trains depart a few minutes after minute .00 and .30. In this way, passengers between many origin stations and departure stations have a service every half an hour. For these types of networks, the timetable has to be constructed for the full network at once. This leads to different optimization models and other computational challenges. As a consequence, this resulted in a different stream of literature and we decided to devote a separate section to each stream.

1.2 Outline of the paper

The remainder of this paper is structured as follows. In Section 2, we discuss two basic models and the literature on railway timetabling for independent routes. Section 3 presents a basic model for network timetabling and station planning in the context of dependent routes. Robustness considerations in timetabling are discussed in Section 4. The rolling stock planning literature is reviewed in Section 5. We conclude this paper in Section 6 with propositions of new research directions and challenges.

2 Timetabling for independent routes

This section deals with timetabling problems that arise from railway networks that can be decomposed into single independent routes. Trains running within such routes have little to no interaction with trains from different routes. For example, high-speed trains running within a dedicated high-speed route usually interact with regional trains only in few large stations, and this interaction is so limited that can typically be ignored. Another example is the railway infrastructure in Norway, in which all major routes stem from Oslo forming a star-shaped network.

When dealing with a single independent route, passenger connections with other routes become less relevant and they can usually be ignored. This simplification allows the model to consider a rich set of realistic constraints, and to solve both network and station timetabling simultaneously. As we will see in the next section, this is typically not possible when dealing with dependent routes.

The literature on timetabling for independent routes is overwhelmingly dominated by two formulations, one that makes use of the (infamous) big-M method and a time-indexed formulation. We will briefly introduce these two formulations before discussing the main recent contributions in strategic, tactical, and operational timetabling.

2.1 Basic big-M formulation

A classical and most exploited way to model train scheduling problems as a mixed integer linear program (MILP) is by means of the so called big-M formulation. The first such formulation was proposed by Carey and Lockwood (1995) and focuses on determining the order in which trains should be scheduled. In this type of model, the order of trains is defined using binary variables and the event times are defined using continuous variables. A more recent and comprehensive basic model corresponds to the one introduced in (Mascis & Pacciarelli, 2002) for generic job-shop scheduling problems with no-wait and blocking constraints. The railway network is discretized into a set S of block sections. Each block section is preceded by a signal and can accommodate at most one train at a time. Let I be the set of trains. Assuming the route of each train $i \in I$ through the network is given, this can be seen as a sequence $S^i \subseteq S$ of block sections. Then, a real variable t_s^i is associated with each train $i \in I$ and each block section $s \in S^i$ on the route of i, representing the time in which i enters s. The timetable determines, for each train, the departure times T_s^i from some special timing points $s \in D^i \subseteq S^i$, for instance at stations. We then have

$$t_s^i \geq T_s^i \qquad i \in I, s \in D^i \tag{1}$$

Next, let τ_s^i be the minimum time for train $i \in I$ to run through track section $s \in S^i$, and let denote by $d^i \in S^i$ the destination of i and by t_{s+1}^i the time i enters the block section following s in S^i, we have

$$t_{s+1}^i - t_s^i \geq \tau_s^i \qquad i \in I, s \in D^i \setminus \{d^i\} \tag{2}$$

Next, let i and $j \in I$ be distinct trains, and assume $r \in S^i$ and $s \in S^j$ are actually the same block section. Because the block section can accommodate at most one train at a time, then either train i exits r

(and enters the next block section in its route) before train j enters s, or j exits s before i enters r. This is immediately translated into the following *disjunctive constraint*:

$$t^i_{r+1} \leq t^j_s \ \ \textbf{OR} \ \ t^j_{s+1} \leq t^i_r.$$

The above disjunctive constraint can be linearized by applying the so-called *big-M trick*, namely by introducing a suitable large constant M and a binary variable x^{ij}_{rs} which is equal to 1 if train i exits r before train j enters s and is equal to 0 if j exits s before i enters r. Then the disjunctive constraint can be replaced by the following pair of linear constraints:

$$\begin{aligned} t^i_{r+1} &\leq t^j_s - M(1 - x^{ij}_{rs}) \\ t^j_{s+1} &\leq t^i_r - M x^{ij}_{rs} \end{aligned} \tag{3}$$

for all distinct pair of trains (i, j) on $I \times I$ and all incompatible pairs or sections (r, s), with $r \in S^i$ and $s \in S^j$.

As for the objective function, this consists usually in minimizing a cost associated with the deviation of the schedule $\mathbf{t}$ from a timetable $\mathbf{T}$. For instance, for each train $i \in I$, the delay in some timing points $A^i \subseteq S^i$ (as the incoming track in major stations or the final destination): in this case, the objective function can simply be:

$$\sum_{i \in I, s \in A^i} \max(0, t^i_s - T^i_s) \tag{4}$$

where T^i_s is the wanted arrival time in $s \in A^i$.

The objective function (4) can be easily linearized, and so (1)-(2), together with the binary stipulation on the x-variables (3) provide a big-M MILP formulation for the train timetabling problem.

Finally observe that it is not difficult to extend the model to cope with generic incompatible pairs of block sections, more complex section release mechanisms and to include the possibility of alternative routes for each train.

2.2 Basic time-indexed formulation

Big-M formulations like the one described above are compact but have weak relaxations, meaning that they usually return low quality bounds. A different approach for scheduling problems is introduced in Sousa and Wolsey (1992), where binary variables are used to represent events happening in a certain time window. It was first applied to train scheduling problems in Brännlund et al. (1998) and later improved upon by Caprara et al. (2002).

First, the planning time horizon is partitioned into typically, but not necessarily, equally sized time intervals $H = \{h_1, \ldots, h_q\}$. For the sake of simplicity, for $l = 1, \ldots, q$ we let h_l be the starting time of the $t - th$ interval.

Then, we introduce, for each train $i \in I$, each block section $s \in S^i$ and each $t \in H$, a binary variable x^t_{is} which 1 if i enters s at time $t \in H$, and 0, otherwise.

Clearly, since every train must enter every block section in its route S^i at some time, we have that

$$\sum_{t \in H} x^t_{is} = 1 \qquad i \in I, s \in S^i \tag{5}$$

Next, we can rewrite constraints (1)-(3) as linear constraints in the new variables. In particular, a train $i \in I$ cannot leave $s \in D^i$ before T^i_s:

$$x^t_{is} = 0 \qquad i \in I, s \in D^i, h_t < T^i_s \tag{6}$$

Then, a train i cannot travel faster then its minimum running time between two successive block sections:

$$\sum_{h < t + \tau^i_s} x^h_{is^+} \leq 1 - x^t_{is} \qquad i \in I, s \in S^i, t \in H \tag{7}$$

where, for $s \in S^i$, s^+ denotes the block section on the route of i next to s.

The fact that not two trains can be at the same time on the same block section can be expressed by the following incompatibility constraints

$$x^t_{ir} + x^h_{js} \leq 1 \tag{8}$$

whenever i, j are distinct trains, $r \in S^i$ and $s \in S^j$ are the same block section (or incompatible block sections), and $t + \tau^i_r > h$ AND $h + \tau^j_s > t$. Indeed, suppose that constraint (8) is violated for a pair $(\bar{i}, \bar{r}, \bar{t}), (\bar{j}, \bar{s}, \bar{h})$, that is $x^{\bar{t}}_{\bar{i}\bar{r}} = x^{\bar{h}}_{\bar{j}\bar{s}} = 1$. This implies that $\bar{r}$ and $\bar{s}$ are incompatible block sections and train $\bar{i}$ enters $\bar{r}$ at time $\bar{t}$ and a distinct train $\bar{j}$ enters $\bar{s}$ at time $\bar{h}$. Now, since $\bar{t}, \bar{h}$ satisfy $\bar{t} + \tau^i_r > \bar{h}$ AND $\bar{h} + \tau^j_s > \bar{t}$, then train $\bar{i}$ exits $\bar{r}$ after $\bar{j}$ enters $\bar{s}$ AND $\bar{j}$ exits $\bar{s}$ after $\bar{i}$ enters $\bar{r}$, and the two trains will be on the same (or incompatible) block section at the same time.

Finally, the objective function can be expressed as

$$\min \sum_{i \in I} \sum_{s \in S^i} \sum_{t \in H} c^t_{is} x^t_{is} \tag{9}$$

where c^t_{is} is the cost of train i entering block section s at time t. Note that any objective function, even non-linear, which only depends on the entry or exit times of the trains in some block sections, and it is separable in such times, can be expressed in the form (9). This is indeed the case of the most common objective function, both in the literature and in the practice.

Note that the constraint of the time-indexed formulation can be easily strengthened in order to obtain stronger formulations. Also, the discretization allows to model rather complex physical and logical requirements by linear constraints. One major problem is that the number of variables and constraints can grow very large, especially if the time-discretization is fine, making the solution process very slow. On the other hand, a too gross time discretization may lead to unrealistic solutions which would turn to be infeasible in practice. For an interesting discussion on these and other issues related to the application of time-indexed formulations to train scheduling, we refer the reader to Harrod (2011).

2.3 Strategic timetabling

Being usually more difficult than its tactical and operational siblings, strategic timetabling for independent routes has not received much attention from the academic community. Only recently, two attempts were made into this direction.

Sartor et al. (2023) consider the problem of creating a feasible timetable starting from a given set of required periodic train services. While periodic trains are almost always preferred to non-periodic ones (at least by customers), they create a constraint that can substantially reduce the capacity of a route. In this work, the authors introduce the concept of quasi-periodic strategic timetables, which allows some deviation from strict periodicity in order to expand the set of feasible solutions, while guaranteeing a periodic timetable for the customers. The resulting MILP model is based on a big-M formulation solved by delayed row generation.

Coviello et al. (2023) introduce a new method for the efficient creation sets of strategic (not necessarily periodic) timetables using minimum infrastructure information. The method is based on a multi objective ant colony optimisation algorithm and a mixed integer linear programming (MILP) formulation, and was verified on the real-life instance of the Bergen-Oslo railway line in Norway.

2.4 Tactical timetabling

Single line timetabling using a big-M MILP formulation was first discussed by Carey and Lockwood (1995), proposing a mix of heuristic and branching procedures to speed up the computation. Higgins et al. (1997) later proposed a similar formulation and develop a set of methods based on local search, genetic algorithms, tabu search, and hybrid combinations of these to solve the TTP.

Around the same time, time-indexed formulation for this problem were proposed in Brännlund et al. (1998) and Caprara et al. (2002). A bit later, Caprara et al. (2006) developed a Lagrangian heuristic algorithm tailored to real-world conditions of the Italian railway system. Fischer et al. (2008) employ a Lagrangian

relaxation combined with a cutting plane approach to manage the substantial number of variables and constraints, focusing on a test instance that represents ten percent of the German network. Cacchiani et al. (2008) modify the time-space formulation to consider train paths instead of stations, and develop a column generation algorithm and local search heuristics to solve the TTP. They verify their algorithms on real-life instances of the Italian railway network. Cacchiani et al. (2010a) extend the model to account for the scheduling of extra freight trains on the network and use a similar Lagrangian heuristic as Caprara et al. (2006). Later, Fischer (2015) introduce ordering constraints improving the relaxation of the formulation and tested on instances of the RAS Informs competition (INFORMS, 2012).

More recently, Kloster et al. (2023) proposed an incremental approach to tactical timetabling based on a big-M formulation solved with a custom row-and-column generation algorithm. The idea is to find a new feasible timetable that is as close as possible to a (possibly infeasible) reference timetable. The route planner can then incrementally modify the reference timetable to steer the result toward the desired direction. The authors report a very good feedback from route planners who tested a prototype of such method on the Oslo-Trondheim line in Norway.

The models presented in this section can sometimes be extended to timetabling for dependent routes in case of an acyclic timetable. Often, heuristics are needed to solve the mixed-integer programming formulations in this case. For example, Gestrelius et al. (2017) developed an incremental fix and release heuristic and an improvement heuristic to solve tactical timetabling problems in the Swedish region around Hallsberg.

2.5 Operational timetabling

In operational timetabling, as in dispatching and disruption management, one has at hand an existing timetable T, and needs to modify T to factor in unforeseen or planned deviations. The modification can involve both the schedule of trains at different locations and their routes and one wants to minimize the distance of the modified timetable from T.

The major difference between operational timetabling and *dispatching* is that for the latter the available computation time is very short, typically a few seconds, while in operational timetabling one may have more time at disposal. Indeed, in dispatching, the modifications to the official timetable T are computed in real-time in order to tame the effects of primary delays and minimize knock-on effects. However, dispatching and operational timetabling share constraints and objectives. As a consequence, all the (huge) body of scientific literature devoted to automatic train dispatching and rescheduling can be regarded as a basis also for operational timetabling. Because the literature on train rescheduling is vast, here we will only mention a few survey papers on this topic as Cacchiani et al. (2014), Corman and Meng (2015), Fang et al. (2015), and Lamorgese et al. (2018). Instead, we will consider in more detail the papers specifically devoted to operational timetabling as defined in Section 1.

In this survey we will look only at operational timetabling for some planned activities, and we consider two main streams:

1. Maintenance activities, which involve the closure of some tracks and require rescheduling and rerouting of trains.

2. Addition of individual train service to the current service.

In both cases, the objective will be to minimize the impact on the current timetable. In both cases, the existing body of literature is quite limited.

Before delving into the literature, it is worth mentioning that in the current practice, the operational timetabling process is performed manually and usually alternates between (1) making rough modifications to an existing timetable (e.g., shifting the departure of a train by half an hour) and then (2) making small adjustments to regain feasibility (e.g., reduce or increase the dwell time of some trains in some stations or "bending" the running time). The most time-consuming element of this process is related to the second step, that is to manually eliminate all conflicts that may arise after a timetable has been modified. Currently available commercial tools support the route-planners in various ways, and in particular in identifying train conflicts generated at step (1). However, to our knowledge, there is no available commercial tool yet for conflict resolution to regain global feasibility and possibly optimality with respect to defined target KPIs like e.g., delay, capacity gain or mobility impact on end-customers.

2.5.1 Planning maintenance activities

As observed in by Lidén (2015) and D'Ariano et al. (2019), the scheduling of preventive maintenance activities and the consequent rescheduling of trains are normally treated separately in two successive phases by infrastructure managers. D'Ariano et al. (2019) refer to the overall problem as the Tactical Traffic and Possession Scheduling Problem (TTPSP) where *decisions about retiming, re-sequencing and re-routing of trains have to be taken, before the operational day, with some knowledge of maintenance actions and traffic perturbations.*

There are already a number of surveys on planning maintenance activities in railroads, some as recent as 2021 (see Lidén, 2015; Sedghi et al., 2021). However, these surveys are centred on the maintenance schedule, and papers are classified according to this focus. In the initial works about maintenance the planning of the activities was carried out without an exact re-schedule of train movements. We will only mention a couple of references, and focus instead on the papers which deals also with the re-computation of train timetables.

Only maintenance operations. One of the first works dealing with planning track maintenance activities is probably Higgins et al. (1999). They present a time-indexed formulation for scheduling maintenance operations in a corridor. The main limitation is that they do not produce a modified train timetable to take into account track blockages. The impact of the maintenance schedule on the train schedule is only considered by some approximating terms in the objective function. Also the more recent work such as Quiroga and Schnieder (2010) is affected by the limitation of only considering the scheduling of the maintenance (in this case, tamping) activities, without calculating a new feasible schedule for trains.

Integrating train timetabling and maintenance schedule. Albrecht et al. (2013) apply a probabilistic metaheuristic to find a plan to schedule maintenance works and 50 trains in the 480km long single-track North Coast Line in Queensland, Australia. The basic idea is to handle maintenance activities as fictitious train services, and then schedule real and fictitious train services simultaneously. The method produces several conflict-free timetables, sparing time of human planners which can instead spend it to choose the preferred timetable. Compared with the current practice, it led up to 34% delay reduction.

Fictitious (virtual) trains are also utilized in Luan et al. (2017) to represent maintenance activities. The model is a MILP time-indexed formulation (Section 2.2) equipped with multicommodity flow constraints to represent different routings. The model is single objective, namely the minimization of the deviation of trains schedule from the wanted one. The basic instance is derived from a portion of the Chinese railway network.

Modelling maintenance activities as fictitious, slow moving trains is smart but cannot be always applied. As observed in Lidén and Joborn (2017), this is not sufficient for situations where (a) a work activity can be interrupted (for letting real trains through), (b) the work closes off several tracks or line segments at the same time, or (c) the work inflicts speed restrictions or other operational restrictions on neighboring tracks. Moreover, there are additional features characterizing maintenance activities: (i) work tasks can be co-located, (ii) the costs have a non-linear dependency on the given shift and possession time and (iii) work force restrictions are differently handled. For this reasons, Lidén and Joborn (2017) introduce a comprehensive time-indexed (see Section 2.2) MILP model which considers maintenance activities and train movements as different type of events. The objective is the weighted sum of multiple terms to take into account the deviation of the train timetable from the wanted timetable, and the cost of executing a maintenance works on specific links (track sections) in specific time periods. The model considers an aggregated view of the train movements and therefore neglects the detailed resolution of conflicts and the meet/pass events precise specification. These are left for a hypothetical second phase, where the timetable problem in detail. They finally consider a set of instances both with single and double tracks. Although it is unclear if the instances are real-life or realistic, the authors have made the valuable effort to make them available at https://github.com/TomasLiden/mwo-data.

Also Forsgren et al. (2013) schedule simultaneously train services and maintenance works (track possession), considering also train cancellations and rerouting. The model is an extended, multi-objective big-M formulation (Section 2.1), which is tested on two complex real-life scenario from the Swedish railway system. The objective cost takes into account train cancellations, potential conflicts in the use of tracks, and the deviation from the original timetable. Bababeik et al. (2019) also considers the case of a single-track line, but factoring in with uncertain duration of maintenance works. The works will of course make some tracks unavailable to trains, and the schedule of the trains will be affected. A discrete probability distribution of duration is given for each maintenance operation, and this is used to constraint the minimum wanted duration of the same activity. A massive MILP model (based on the big-M model of Section 2.1) is built, and tested

on a single instance with a 15km line and 8 stations (probably real-life, but the authors do not say anything about that).

Also D'Ariano et al. (2019) address the problem of uncertain duration of future maintenance operation. These are handled by considering several alternative scenarios, weighted with their probability to occur. The basic model is the big-M MILP of Section 2.1 with multi-objective function, considering both the expected cost of delaying trains and the expected benefit of maximizing maintenance operations overlap. The two objectives are then combined in a single function by choosing suitable weights. In a second approach (the ϵ-constraint formulation), one of the objectives is instead modeled as a constraint while the other one is minimized. Another interesting feature of the model is to consider re-routing for the trains, which in turn is modeled as multi-commodity flow. This is particular relevant and realistic, because in many practical cases of maintenance track possessions, trains can and must be re-routed on alternative paths. The test instances were derived from the one used for the RAS 2012 railway challenge (INFORMS, 2012).

2.5.2 Adding extra train services

Adding extra services to a given timetable is a very common exercise carried out regularly by infrastructure managers. This is required to satisfy some extra demand, or to schedule work trains. Despite of this, the scientific literature on the topic seems to be not too large. When inserting new services into a given timetable, one is typically allowed, at some cost, to partially modify the schedule of the original trains. In particular, strict periodicity maybe relaxed, connections between trains may be lost, departure or arrival times at stations can be perturbed.

D. Jovanović and Harker (1991) presents one of the first MILP approaches to operational timetabling, called SCAN. The tool allowed to modify an existing schedule in order to add or delete trains, or meet new requirements. Although the authors claim the SCAN system was implemented and utilized by a class A American railroad company, Norfolk Southern, no details are given on the quality of the resulting plans or the actual quantitative benefits for the company.

In Tan et al. (2021) a MILP big-M model is developed for the problem of adding extra periodic train services to a timetable for high-speed trains. Numerical experiments are performed on 20 instances from Shanghai-Hangzhou High-Speed Railway in China. Besides considering several realistic constraints, such as acceleration and deceleration, the model also factors in periodicity, an important feature when handling passenger trains.

The problem of inserting non periodic extra-trains in a (possibly, partially) periodic timetable is instead addressed by several paper, such as Burdett and Kozan (2009), Erlandson et al. (2023), Ingolotti et al. (2004), Lamorgese et al. (2017), and Tan et al. (2020). These papers differ in the way some constraints are handled, such as losing connections or relaxing periodicity, and the solution methods, which varies from genetic algorithms (Tan et al., 2020) to sequential heuristics (Ingolotti et al., 2004), meta-heuristics (Erlandson et al., 2023, Burdett and Kozan, 2009), and Benders' decomposition (Lamorgese et al., 2017). In some cases, the original timetable cannot be modified at all, at any cost, typically because of contract clauses with the railway undertakings. This particular problem is addressed for instance in Ljunggren et al. (2021), where real-life instances from the Swedish railway network are solved by a variant of the Dijkstra's shortest path algorithm.

A slightly different perspective was investigated by Pellegrini et al. (2017). Here, the authors consider the railway saturation problem and propose the RECIFE-SAT, a MILP-based algorithm based on a hybrid time-indexed formulation with big-M constraints that estimates the capacity of a certain railway line by adding as many trains as possible while maintaining a feasible timetable. This method can then be used to evaluate infrastructure upgrades or other high-level decisions on the amount of trains that should serve a certain line.

3 Timetabling for dependent routes

In complex railway networks, routes cannot be scheduled independently. In addition, planning the stations is a complex tasks. Therefore, timetables are usually constructed in two steps (see Cacchiani et al. (2015)). First, a timetable on the **network** (or macroscopic) level is constructed. In other words, network timetabling aims at creating a feasible schedule with arrival and departure times of all trains at all stations. The considered

	Network	Station
Strategic	Polinder et al. (2021) de Graaf (2021)	Delorme et al. (2001) Sels et al. (2014) Jensen et al. (2017) Bešinović and Goverde (2019) Schmidt et al. (2019) P. Jovanović et al. (2020) Weik et al. (2020)
Tactical	Schrijver and Steenbeek (1993) Nachtigall (1994) Odijk (1996) Peeters (2003) Kroon and Peeters (2003) Liebchen (2006) Liebchen and Möhring (2007) Liebchen (2008) Großmann et al. (2012) Gattermann et al. (2016) Robenek et al. (2017) Farina (2018) Lindner and Liebchen (2019) Matos et al. (2021) Lindner and Liebchen (2022) Martin-Iradi and Ropke (2022) Polinder et al. (2022)	Zwaneveld et al. (1996) Zwaneveld et al. (2001) Lusby, Larsen, Ryan, and Ehrgott (2011) Caimi et al. (2011) Caprara et al. (2011) Dewilde et al. (2013) Dewilde et al. (2014) Caprara et al. (2014) Burggraeve and Vansteenwegen (2017)
Operational	Van Aken et al. (2017a) Van Aken et al. (2017b) Bešinović et al. (2020)	—

Table 1: Classification of timetabling papers for dependent routes in 6 categories

infrastructure information is accounted for by headway times between the events (the arrival and departure of trains) to be scheduled. In a next step, **station planning** at the microscopic level is done for each (major) station. On microscopic level, individual switches and signals are taken into account.

Almost all European countries operating such a complex network also operate a **cyclic** timetable. This is a timetable that repeats itself after its cycle time. The cycle time is typically one hour (e.g. the Netherlands) or two hours (e.g. Germany). Cyclic timetables are generally preferred by passengers as they are easier to remember, leading to an increased demand. Therefore, many railway undertakings operate such timetables. However, cyclic timetables are also associated with high costs since the timetable must be repeated during the day, during and outside of peak hours (Cacchiani & Toth, 2012).

In the remainder of this section, we present the basic PESP formulation for constructing a cyclic timetable on the network level in Section 3.1. Section 3.2 discusses a basic mixed integer programming formulation for the station planning. Afterwards, we discuss the literature on strategic, tactical and operational level for network and station planning in Sections 3.3-3.8. An overview is presented in Table 1.

3.1 Basic PESP formulation

Most cyclic timetabling models are based on the Periodic Event Scheduling Problem (PESP) as defined by Serafini and Ukovich (1989). In PESP, we are given a cycle period T and event-activity network $G = (V, A)$ where V is the set of events to be scheduled and A is the set of activities linking those events. Each event $i \in V$ represent the arrival or departure of each train service at each station. An activity $(i, j) \in A$ is a directed arc that represents the time difference between two events i and j such that $(i, j) \in A$. The lower and upper bounds for the time duration that activities can take is defined by *activity constraints*. In its simplest form, the PESP is to find a feasible schedule of event times $\pi : V \to \{0, \ldots, T-1\}$ satisfying the

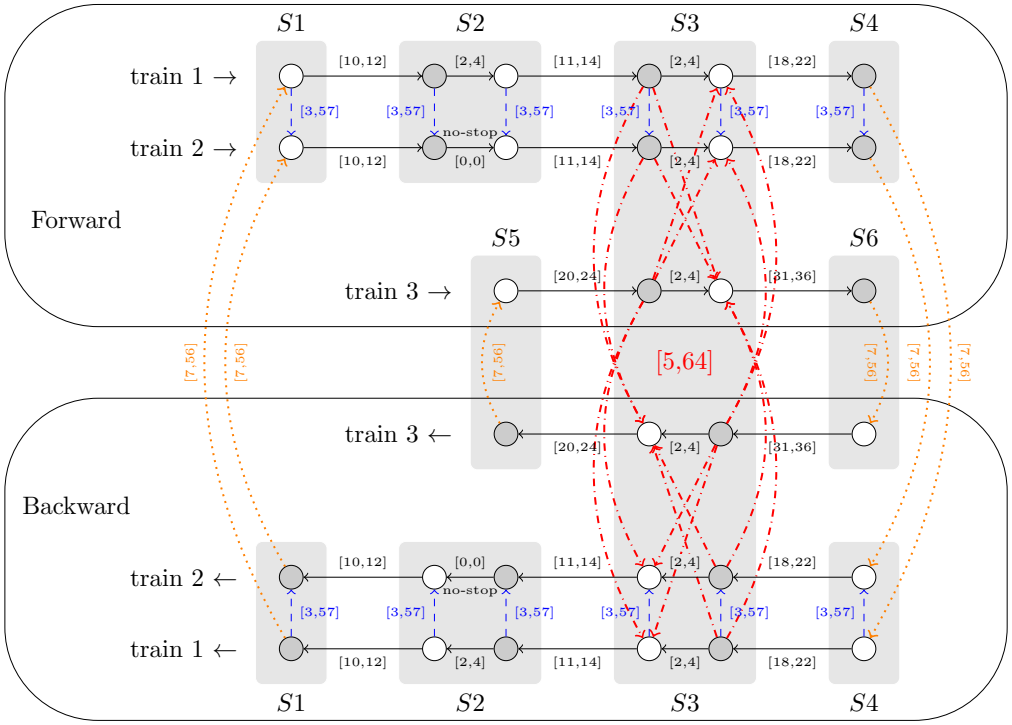


Figure 1: Example of Event Activity Network for a cyclic timetable of cycle period $T = 60$; The black straight arrows represent the drive and dwell activities of a line, the dashed blue arrows represent the headway activities between trains, the red dashed-dotted arrows represent the transfer activities, and the orange dotted arrows represent the turnaround activities.

activity constraints

$$l_{(i,j)} \leq (\pi_j - \pi_i) \bmod (T) \leq u_{(i,j)} \qquad \forall (i,j) \in A$$

where $l_{(i,j)}$ and $u_{(i,j)}$ are respectively the lower and upper bounds on the duration an activity $(i,j) \in A$ can take. Note that $l_{(i,j)}$ and $u_{(i,j)}$ must satisfy that $u_{(i,j)} - l_{(i,j)} \geq T - 1$, else the constraint becomes redundant as all time differences between π_i and π_j are allowed. Finally, it can be noted that the formulation above is non-linear due to the modulo function. Let $p_{(i,j)}$ be an integer variable used to model the modulo function MILP formulation of the PESP is

$$l_{(i,j)} \leq \pi_j - \pi_i + T p_{(i,j)} \leq u_{(i,j)} \qquad \forall (i,j) \in A \tag{10a}$$

$$p_{(i,j)} \in \mathbb{N}^0 \qquad \forall (i,j) \in A \tag{10b}$$

$$\pi_i \in \{0, \ldots, T-1\} \qquad \forall i \in V \tag{10c}$$

Activities are used to model the movement of trains and passengers in the network, but also the infrastructure and safety requirements ensuring the safe operation of the timetable. As exemplification, we present here 5 common types of activities depicted the event-activity network shown in Figure 1 and how to model them using the PESP formulation:

Drive activities represent the time spent by a train travelling from one station to another. The lower bound of a drive activity constraint is defined by the minimum travel time given the distance and the maximum speed of the train between two stations. An upper bound is not necessary but can be defined by the maximum allowed deviation from the lower bound.

Dwell activities represent the time spent by a train at a station. The lower bound of a dwell activity represents the minimum time spent at a station by a train ensuring that the passengers have enough

time to board or disembark the train. Similarly to drive activities, the upper bound can be defined by the maximum allowed deviation from the lower bound.

Transfer activities represent the time allocated for the transfer of a passenger from one train to another. Transfer activity constraints provide a lower bound for transfer times such that passengers have the time to go from one platform to another. The upper bound of a transfer activity (i, j) is defined as $u_{(i,j)} = T - l_{(i,j)} - 1$, such that $u_{(i,j)} - l_{(i,j)} \geq T - 1$ is satisfied.

Turnaround activities represent the reuse of rolling stock after the end of a train service. After the arrival of a train at its terminal station, the same train is often used to serve the line in the other direction. The minimum time restriction on this activity is at least the time that is needed for the driver to get to the other side of the train. An upper bound can be defined to avoid long turnaround times, as a measure aimed at reducing the number of vehicles needed to operate the timetable and thus reduce rolling stock costs.

Safety activities represent infrastructure constraints that guarantee a safe operation if all trains operate according to the timetable. Safety activity constraints define the minimum time difference between the arrival or departure of two trains using the same tracks. Given a headway time $h_{(i,j)}$ between two events i and j, the lower bound is equal to $h_{(i,j)}$ and the upper bound is defined as $u_{(i,j)} = T - h_{(i,j)}$ such that the headway time is accounted for no matter which event happens first during the cycle period.

Many more aspects of the timetable can be modeled using PESP activity constraints. We refer to Liebchen and Möhring (2007) for an extensive description of the modeling capabilities of the PESP for railway timetabling.

3.2 Basic Routing formulation

When train timings are pre-defined (or restricted to a discrete set of possible values), we can model routing decisions for trains as a node packing problem, as first proposed by Zwaneveld et al. (1996). This is of particular interest in a station context, where arrival and departure times may be the output of a network-wide optimization in a previous planning step and the proposed schedule must now be translated into a microscopically feasible train routing.

Train runs in stations are defined by their routes, as well as the traversal times along these routes. The combination of a physical route and the traversal time of a train along it is called *train path* (following Lusby, Larsen, Ehrgott, and Ryan (2011)). Thus, different paths may use the same route but differ in arrival or departure times. To judge whether two paths are in conflict, spatial and temporal constraints induced by the signaling system have to be respected. In particular, *block sections* referring to infrastructure segments which are exclusively usable by one train at a time, are fundamental for microscopically feasible routing concepts in station areas (for an introduction to infrastructure utilization and rail traffic control according to the blocking time see Pachl (2018)). Since different block sections need not be disjoint, e.g. in the case of merging or splitting routes, some approaches (Caimi et al., 2011; Lusby, Larsen, Ryan, & Ehrgott, 2011; Pellegrini et al., 2014) divide them into multiple smaller, non-overlapping *track sections* to simplify conflict modelling. This has the added benefit of allowing a route-lock sectional-release traffic control system, where occupation for all track sections in a block section must start at the same time, but may end early for some track sections that the train has already freed. This extends the standard route-lock route-release traffic control system, where block sections are always occupied and released as a whole (see also Pellegrini et al. (2014) for a discussion of the two interlocking systems).

The train routing problem now consists of assigning a path to each train to be scheduled that connects its entry and exit points in the analyzed infrastructure. This problem is sometimes also referred to as the train platforming problem, particularly in cases where only one route is considered per platform. The set of all paths must be conflict-free according to the blocking time model. To this end, we find all possible physical routes available for a train and – through simulation or analytical evaluation – determine the occupation times of all block and track sections along these routes for the given arrival and departure times. If we denote by T the set of all trains, this yields for each train $t \in T$ a set of potential paths P_t, combined into the set of all potential paths $P = \bigcup_{t \in T} P_t$. Note that, while arrival and departure times are fixed, the occupation times themselves need not be discretized and can be expressed with arbitrary precision.

Not all paths are compatible with each other, which we model through a set of conflicts

$$E = \{\{p, p'\} \,|\, p, p' \in P \text{ are incompatible}\}. \tag{11}$$

There are two types of important incompatibilities: First, all paths belonging to the same train are pairwise incompatible, as only one path can be chosen per train. Secondly, two train paths for different trains that occupy incompatible block sections for some overlapping time interval are also in conflict.

Now, the tuple (P, E) captures our routing problem in the form of a conflict graph. The objective is to select a maximal independent set, i.e. a maximal node set such that no two contained nodes are neighbours, by assigning binary selection variables $x_p \in \{0, 1\}\, \forall p \in P$. A solution with objective value equal to the number of trains guarantees a feasible train routing. In its simplest form, this leads to the optimization problem

$$\max \quad \sum_{p \in P} x_p \tag{12}$$

$$\text{s.t.} \quad x_p + x_{p'} \leq 1, \qquad \forall \{p, p'\} \in E \tag{13}$$

$$\boldsymbol{x} \in \{0, 1\}^{|P|} \tag{14}$$

Despite its intuitiveness, the authors note that this formulation suffers from a weak LP-relaxation and a high number of constraints. To remedy this somewhat, one can replace pairwise exclusions with valid clique inequalities, though these have to be carefully selected as their number is exponential. The authors propose to use cliques of paths belonging to only one or two trains, an approach also found in subsequent works (Caprara et al., 2011; Dewilde et al., 2014, 2013; Zwaneveld et al., 2001).

Finally, we note that the concept of generic train conflicts is a very flexible tool for setting up the optimization problem: Beyond modelling different traffic control systems as mentioned above, it can also encode desired relations between trains (e.g. routing transfer connections to the same platform) by forbidding those path combinations that violate the desired properties.

3.3 Strategic network timetabling

In complex railway networks, infrastructure can often be a limiting factor affecting the feasibility and quality of timetables. Hence, timetables are sometimes used to improve the quality of strategic decisions such as line planning and railway network design. For line planning, some attempts have been made at integrating (parts of) the line planning and timetabling problem to find better timetables (Burggraeve et al., 2017; Correia Duarte et al., 2023; Michaelis & Schöbel, 2009; Schöbel, 2017; Yan & Goverde, 2019). For railway network design, a strategic timetable is generated by abstracting part of the infrastructure requirements, and used as input for planning steps for network design. This method is employed in various Western European nations, such as Switzerland, Germany, and the Netherlands, but rarely covered in scientific literature (Friesen et al., 2024).

To the best of our knowledge, only a few recent papers explicitly focus on the creation of strategic timetables for railway network design. Polinder et al. (2021) study the creation of strategic cyclic timetables that dismiss headway constraints to find "ideal" passenger oriented timetables that can serve as blueprints for railway network design problems. The model, integrating passenger routing, aims at minimising the passenger perceived travel time defined as a weighted sum of passenger waiting time and travel time and was validated on real life instances of the Intercity Network of NS. Furthermore, de Graaf (2021) investigates how timetables generated by the model of Polinder et al. (2021) can facilitate the strategic timetable design process. The research concludes that the emphasis of strategic timetables on waiting time and efficient transfers can provide valuable information that can be used to design tactical timetables, for instance in deciding which transfers to prioritise from a passenger perspective.

3.4 Tactical network timetabling

In the tactical timetabling phase, the goal is to design a yearly timetable considering the network's capacity and service requirements. In complex railway networks, tactical network timetabling is often modeled

as PESP (see Section 3.1) and solved using constraint programming or mixed integer programming methods. Notable early contributions include the development of the DONS decision support system for cyclic timetabling at NS by Schrijver and Steenbeek (1993), the development of the Cycle Periodicity Formulation for the PESP by Nachtigall (1994), and the work of Odijk (1996) on constraint generation algorithms for the PESP.

In particular, the Cycle Periodicity Formulation (CPF) developed by Nachtigall (1994) is the most studied formulation for the PESP. The CPF defines activity duration as variables and uses a new set of constraints to enforce the cyclic nature of the solution within the associated event-activity network. By doing so, fewer integer variables are created and tighter constraints are defined, leading to a stronger formulation for the linear programming formulation than that of Serafini and Ukovich (1989).

Peeters (2003) further studied applications of the CPF to the Dutch Railway Network using different objective functions, including the minimisation of passenger travel times, the maximisation of the timetable robustness, the minimisation of required rolling stock, and the minimisation of violation of soft constraints. Such timetables are generally created assuming that the travel time between stations is fixed. Kroon and Peeters (2003) extend the problem by considering variable travel times instead of fixed travel times, increasing the solution space and facilitating the search for feasible timetables using similar techniques as models using fixed travel times. Liebchen (2006) develop different methods to solve the PESP using Cut-and-Branch Algorithms for Integer Programs, Constraint Programming, and even Genetic Algorithms based on the CPF. Liebchen (2008) further describes how these optimisation methods are used in the creation of a new Berlin subway timetable in December 2004. Few papers also aim at tackling cyclic timetabling for large instances by decomposing the problem into sub-problems. Lindner and Liebchen (2019) introduce a constraint programming based heuristic to generate T-partitions in the event activity graph to solve smaller problems and merge solutions. This approach is further explored by Lindner and Liebchen (2022) in which a timetable merging crossover heuristic is developed.

Polinder et al. (2022) introduces an iterative approach for constructing a tactical timetable with the objective of minimizing total perceived travel time of all passengers together. They iterate between a formulation which computes an ideal timetable without considering infrastructure constraints (Polinder et al., 2021) and a Lagrangian-based heuristic that satisfies the infrastructure constraints by modifying event times as little as possible.

Beyond common constraint programming approaches, some recent work takes advantage of developments in the performance of solvers for the Satisfiability (SAT) problem and develop new SAT-based formulations for network timetabling problems. Großmann et al. (2012) are the first to introduce a reformulation of the PESP as a SAT problem and provide results on the German network that outperform constraint programming solvers in finding feasible solutions. Gattermann et al. (2016) extend the SAT formulation of Großmann et al. (2012) by integrating passenger routing in the form of an objective aiming at minimising passenger travel time, becoming a Maximum Satisfiability (MaxSAT) problem formulation. Matos et al. (2021) later develop a state-of-the-art reinforcement learning approach to solve the new MaxSAT formulation, outperforming existing methods over some of the publicly available instances of the PESPlib library.

Finally, some methods deviate from the PESP in the generation of tactical network timetables. For example, Robenek et al. (2017) consider a model between cyclic and acyclic timetabling and display the interest of keeping regularity for a fixed set of lines instead of all lines to solve the problem on a network level. The model is solved using a simulated annealing method and verified on the Israeli railway network. We further refer to Farina (2018) and Martin-Iradi and Ropke (2022) for recent state-of-the-art approaches to cyclic timetabling based on the acyclic TTP time-space formulation of Cacchiani et al. (2010a) using respectively local search methods and a column generation approach, both tested on parts of the Danish railway network.

3.5 Operational network timetabling

In the operational timetabling phase, the aim is to adjust the tactical timetable to account for planned changes in network infrastructure, such as maintenance or construction works that leave parts of the network unavailable. Since we do not want the whole network to be impacted by local changes, the alternative timetables should deviate as little as possible from the tactical timetable. In the process of adjusting a timetable, different changes from the tactical timetable can be expected, such as train retiming, i.e. changes

in the scheduled event times, train reordering, i.e. changing the order of arrival of trains at stations, train cancellation, i.e. the complete removal of a train service or line from the timetable, or short-turning, i.e. changes in turnaround activities in stations close to the disruption location.

To the best of our knowledge, operational network timetabling has been the focus of only a small number of scholarly works. Van Aken et al. (2017a) extend the classical PESP model (Section 3.1) and introduce the Train Timetable Adjustment Problem (TTAP), in which the goal is to generate a timetable that minimizes the deviation from the original timetable to handle infrastructure maintenance. This model is tested on parts of the Dutch network. In this paper, short-turning is fixed in a preprocessing step. Due to the large complexity of the model, Van Aken et al. (2017b) develop solving methods and reduction techniques to tackle larger and more complex instances of the full Dutch network. This paper further extends the TTAP by introducing and evaluating different short-turning procedures. We also refer to Bešinović et al. (2020) for an extension of the model of Van Aken et al. (2017a) to consider freight trains in addition to the passenger trains in the timetable adjustment process.

3.6 Strategic station planning

In the following, we discuss the current state-of-technique for planning in the station context. In particular, we discuss methodological approaches for determining route plans and platform assignments for given macroscopic train routing requirements.

As local station planning is typically performed as a follow-up task after generating a (macroscopically) feasible timetable, strategic timetabling in the local timetable context is relatively rare. The topic has seen increased interest in the context of capacity planning and bottleneck resolution, as attractive (customer-friendly) network timetable concepts suffer from capacity restrictions in junctions or station areas. Given the change of paradigm in railway infrastructure planning towards timetable-driven network and infrastructure design, long-term station planning is hence also concerned with evaluating the characteristics of the available station infrastructure and developing it to meet growing travel demands.

To this end, station capacity, i.e. the number of trains that can be operated on the infrastructure, needs to be assessed. UIC Code 406 discusses the application of the well-known timetable compression method including quality-related threshold levels to assess infrastructure utilization in stations (International Union of Railways, 2013). An earlier method proposes to saturate the station by maximizing the number of trains operated in a given time frame through heuristic solutions for constraint programming and set packing models (Delorme et al., 2001). Successive insertion of additional trains in a timetable concept has also been used to assess the overall number of trains that can be operated in the station area (Sels et al., 2014). Drawing from scheduling approaches, P. Jovanović et al. (2020) proposes to optimize route sequence and to model utilization within junction areas using a graph-coloring method.

(Max,+)-approaches with fix or flexible timing of resource usage have also been applied in the station context, allowing the use of corresponding stability margins or recovery times as robustness metrics (Bešinović & Goverde, 2019). Another option is to use stochastic petri nets allowing to keep certain precedence constraints and timings (Burkolter, 2005; Schmidt et al., 2019; van der Aalst & Odijk, 1995). In order to abstract from a fix routing or timetable scheme, ensemble-averaging approaches have been described to assess infrastructure usage. Jensen et al. (2017) describes a statistical approach to assess capacity based on randomization of train sequences in the network. Weik et al. (2020) build on this method to describe an application-driven procedure to evaluate station capacity in Sweden.

3.7 Tactical station planning

On the tactical level, determining microscopically feasible platform assignments and station route plans for a given macroscopic network timetable concept is a common problem in railway timetabling. Microscopic feasibility is defined based on the blocking time concept which incorporates the design of the signalling system, as introduced in Section 3.2. There, we also described the node packing formulation (Zwaneveld et al., 1996), which is a common choice of model for this problem that forms the basis of multiple subsequent works (Caprara et al., 2014, 2011; Dewilde et al., 2014, 2013; Zwaneveld et al., 2001).

Other approaches to find an efficiently solvable model formulation focus on the locality of conflicts: Lusby, Larsen, Ryan, and Ehrgott (2011) presents a set packing model where *tints* (time interval track sections)

signify the occupation of a certain track section in a certain time slot. The optimization problem then consists in choosing a set of train paths such that each tint is used by at most one path. Following an adjacent idea, Caimi et al. (2011) represents the problem as a multicommodity-flow model: The possible paths for a train are translated into a resource tree whose edges correspond to occupied track sections. With binary variables for all edges, choosing a path for a train is now equivalent to choosing a unit flow through the resource tree that respects standard flow constraints. Conflict constraints can then be formulated per track section, where the occupation times form an interval graph in which maximal cliques can be found efficiently.

Finally, it is also possible to model occupation and release times themselves as variables of a MILP model, with constraints requiring that for each pair of trains on the same track section, the release time of one precedes the occupation time of the other. This approach is used in Sels et al. (2014) and the timetabling model of Burggraeve and Vansteenwegen (2017). As an outlook, this modelling technique is also found in works on dispatching where it allows adding delays to different parts of the train path (Pellegrini et al., 2014).

3.8 Operational station planning

Operational station planning, i.e. the short-term re-planning in the station context for the purpose of maintenance or a time-limited increase in service, has received significantly less scientific attention. The reason for this presumably is that a local adaptation of the timetable or routing scheme in a single station is rarely sufficient in re-planning traffic. In principle, both manual adaptations of the timetable as well as a complete re-planning using methods of tactical station planning are possible depending on the size of the expected disruption. Robust routing approaches, such as the simultaneous scheduling of backup tracks allowing to cope with temporary inaccessibility of some tracks or platforms, to a certain extent, are applicable in this domain (cf. Caprara et al. (2010)).

4 Considerations of robustness in timetabling

Although the goal is often to construct efficient timetables, such timetables can easily become infeasible in the face of delays occurring at the operational level, leading to further delays and possibly train service cancellations. Therefore, many papers in the literature aim at creating timetables that are not only efficient but also robust to delays and delay propagation. A common approach to robust timetabling is the addition of buffer times in the planning phase, representing empty time slots that can be used to absorb delays. Decisions about the length and place of those buffer times in a timetable is a large part of the robust TTP (Cacchiani & Toth, 2012). See for research on the use of stochastic programming (Fischetti et al., 2009; Kroon, Maróti, et al., 2008), light robustness (Cacchiani et al., 2020; Fischetti et al., 2009; Goerigk et al., 2013), recoverable robustness (Cicerone et al., 2009; Goerigk & Schöbel, 2014; Liebchen et al., 2009), and multi-objective methods (Yan et al., 2019) to tackle robustness in the TTP.

Improving robustness has received similar attention in the station context: While different metrics and concepts of robustness exist (cf. the survey by Lusby et al. (2018)), it is generally understood to mean the reduction or limitation of knock-on delays arising from small initial delays of arriving or departing trains.

One metric that has been proposed for measuring robustness is the maximization of buffers (i.e. time intervals) between trains occupying the same track sections, often weighted such that more importance is given to increasing particularly small buffers. The reasoning is straightforward: Any initial delay that is smaller than the buffer to the next train will not cause a knock-on delay. An early example of this approach is from Caimi et al. (2005), where a local search heuristic attempts to widen the buffers between trains starting from an initial feasible solution. Dewilde et al. (2014, 2013) define robustness as the minimization of weighted real travel times of passengers in case of small disturbances, but also improve it via a weighted buffer maximization heuristic: Through repeated iterative application of a route choice module, a timetabling module (which slightly shifts arrival or departure times) and (in Dewilde et al. (2013)) a platforming module, the algorithm reaches a routing with improved spread of the trains.

Buffer maximization also forms part of the two-step approach by Burggraeve and Vansteenwegen (2017): First, the route choice of trains in the station is optimized with the objective of using the infrastructure evenly. Only in a second step, a timetabling module is used to find an optimal schedule that maximizes the buffers between train pairs on these fixed routes. Note that this approach establishes its own timetable

and is thus not easily compatible with a given network timetable optimizer, but could instead be used as a decomposition step for the timetabling of independent routes.

Other robustness criteria have also been studied in the literature: A heuristic method by Bešinović and Goverde (2019) combines multiple metrics, aiming for small capacity occupation, small average train delay and balanced use of infrastructure resources. These objectives are evaluated in subroutines and a greedy multi-start local search heuristic is used to replace routes based on problem-specific route permutation rules. Finally, Caprara et al. (2014) propose a scenario-based approach to model and minimize worst-case delay propagation directly, following concepts of recoverable robustness.

5 Rolling stock planning

Rolling stock is the most expensive and critical resource for a railway undertaking. Rolling stock refers to the powered and unpowered vehicles required to move passengers or freight. In the early literature on rolling stock scheduling, the rolling stock units often correspond to locomotives. In the last decades, passenger railway operator mainly use Electric Multiple Units (EMUs). An EMU is a powered train unit that can drive by itself and can be combined with other EMUs to form a longer train.

Based on the required train paths, decided in train time timetabling, that must be covered, the goal in rolling stock scheduling is to minimize the total cost to operate the rolling stock. This typically means that the rolling stock units should move empty (typically for repositioning) as little as possible. Another related criterion is the minimization of the number of required rolling stock units required to cover the transportation demands. Various technical and functional constraints need to be satisfied, in particular the available number of rolling stock units of each type and the eligibility of a rolling stock unit to operate on its assigned train paths.

The literature has initially focused on the so-called locomotive assignment problem, which is the most simple version of the deterministic rolling stock scheduling problem. Additional constraints or criteria have later been considered in the problem.

5.1 Freight rolling stock

Ziarati et al. (1997) model the locomotive assignment problem as a multi-commodity flow problem with supplementary constraints, and propose a Dantzig-Wolfe decomposition method. To solve a similar problem, Ziarati et al. (1999) present a branch-and-cut approach. The approaches presented in these two papers are validated using real-life data from the Canadian National railway company. Rouillon et al. (2006) improve the approaches of Ziarati et al. (1997) and Ziarati et al. (1999) by introducing a new backtracking mechanism, associated branching methods and new ways to compute upper bounds.

Fuegenschuh et al. (2008) present a multi-commodity minimum cost flow model, written as an integer linear program, that considers cyclic departures of trains, time windows on starting and arrival times, network load-dependent travel times, and that cars can be transferred between trains. Several improvements of the integer linear program are proposed, together with a solution approach that uses a randomized greedy heuristic combined with a standard mathematical programming solver. The approach is validated on real-world instances of the German railway company Deutsche Bahn.

Following Vaidyanathan, Ahuja, Liu, and Shughart (2008), Vaidyanathan, Ahuja, and Orlin (2008) solve a rather general version of the rolling stock scheduling problem. In particular, the authors consider locomotive fueling constraints and locomotive servicing constraints. Both types of constraints require that every locomotive visits a fueling station or a servicing station every time it has reached a given number of miles. An integer programming model is proposed and, as the model has a huge number of variables for relevant instances, an aggregation-disaggregation based algorithm is developed to solve the problem in a few minutes. Computational experiments validate the proposed algorithm on the real-life data of a U.S. railway company.

Recently, Ortiz-Astorquiza et al. (2021) study the locomotive assignment problem for the Canadian national railway company. Two integer linear programming models are proposed, whose originalities lie in the way in which the decisions on the operating mode of each train is modeled. Various improvements are presented for one of the models, as well as a Benders decomposition–based algorithm to determine feasible solutions. These contributions are validated by convincing computational experiments.

5.2 Passenger rolling stock

Cordeau et al. (2000) and Cordeau, Desaulniers, et al. (2001) solve the problem of assigning locomotives and cars in passenger transportation, Cordeau, Soumis, and Desrosiers (2001) propose a multi-commodity network flow-based model. A branch-and-bound method that relies on a Benders decomposition approach is introduced, and validated on real-life instances, in particular from VIA Rail Canada.

Rolling stock scheduling problems in passenger transportation are also studied in Abbink et al. (2004), Fioole et al. (2006) and Alfieri et al. (2006), with computational experiments relying on industrial instances of NS (the main Dutch Railway operator). Abbink et al. (2004) present a model that optimizes the allocation of train units to the railway lines, and whose application shows better results compared to manual planning. This model is used in the tactical planning phase. Alfieri et al. (2006) focus on the optimizing the numbers of rolling stock units of different types based on an integer multicommodity flow model. The coupling and uncoupling of rolling stock units to meet the demand in number of passengers are taken into account, together with the shunting constraints in stations. Focusing on the optimizing the rolling stock circulation in the operational phase, Fioole et al. (2006) consider multiple objectives, in particular the maximization of service quality and reliability instead of only the total operational cost. Moreover, the proposed model allows trains to be combined and split. The method developed in this paper resulted in an application that is used on a regular bases at NS (Kroon et al., 2009). This application was part of the work NS won the Franz Edelman Award in 2008.

About a decade later, DB reached the Edelman final with their application of optimizing algorithms for rolling stock scheduling. Their approach, based on a hypergraph model (see Borndörfer et al. (2016)), allowed to not only take the order of the train units into account but also their orientation. The latter is especially relevant for scheduling ICE train units, where first class carriages and second class carriages are on the opposite sides of the train. In addition, maintenance and regularity constraints are considered.

Cacchiani et al. (2010b), Cacchiani et al. (2013) and Cacchiani et al. (2019) propose a heuristic to solve a real-world rolling stock scheduling problem where a set of train units, each with a cost and a capacity in terms of number of available seats, must be assigned to a set of trips.

5.3 Robust rolling stock scheduling

The robustness of rolling stock plans for passenger trains is investigated in Nielsen et al. (2012), Cacchiani et al. (2012) and Kroon et al. (2015). In these three papers, the proposed approaches are validated on real-life instances of NS. Nielsen et al. (2012) introduce a generic framework for real-time disruption management of rolling stock schedules. A rolling horizon approach is proposed, where schedules are regularly adjusted, and computational experiments are conducted on a set of disruptions. Also to deal with major disruptions, Cacchiani et al. (2012) present a two-stage optimization model, formalized as a mixed integer linear program, to determine robust rolling stock schedules. Benders decomposition is used to solve the linear relaxation of the mixed integer linear program, and to derive a heuristic to determine robust schedules. The computational experiments show that it is much easier to recover from robust rolling stock schedules than from non-robust rolling stock schedules. Kroon et al. (2015) also focus on real-time rolling stock rescheduling in case of large-scale disruptions, and present a simulation-optimization framework. Note that simulation and optimization are often combined to tackle railway rescheduling problems. The proposed framework takes dynamic passenger flows into account, as passengers search for alternative routes in case of disruptions. The numerical results show that the average delay of the passengers can significantly be reduced.

Tréfond et al. (2017) also study the problem of robust rolling stock scheduling for passenger trains. After characterizing robustness indicators, the paper focuses on the determination of the turning times of rolling stock units at stations in a schedule to absorb potential delays. An approach with multiple steps is proposed to determine robust rolling stock schedules. Using a simulation model and real-life instances from the French national railway company, the robustness indicators are significantly improved while maintaining low operating costs and satisfying the maintenance requirements.

More details can be found in the review on robustness in railway planning of Lusby et al. (2018), where a section is dedicated to robustness in rolling stock planning.

5.4 Rolling stock stabling

Passenger train units have to be parked and serviced during the night. This problem is called rolling stock stabling, which is a relevant and hard problem to solve when infrastructure is limited. This is especially the case in dense urban areas. Freling et al. (2005) were the first to study the so-called train units shunting problem, where arrival and departing units need to be matched, parked, cleaned, and routed from the station to a parking track and back. They solved all these subproblems in a sequential way. Kroon, Lentink, and Schrijver (2008) considered a partial integration of the matching and parking step and solved this problem with a mixed-integer programming model. van den Broek et al. (2022) considered a further integration of all steps and solved it with a local search heuristic.

5.5 Integrated planning

Train timetabling remains currently most often not concerned by rolling stock planning, typically because infrastructure managers do not own or are not in charge of the rolling stock. However, a better integration of these two types of decisions could help to improve the profitability of train operators but also, and maybe more importantly, to better use infrastructure and rolling stock by offering both a cheaper and more effective service. The integration of train timetabling and locomotive assignment is studied in Xu et al. (2018). A three-dimensional state-space-time network is introduced, and the problem is modeled as a minimum cost multi-commodity network flow problem with incompatible arcs and integer flow restrictions. A Lagrangian heuristic is proposed, and computational results are reported to illustrate the benefits of integrating timetabling and locomotive planning.

An interesting remark in Lusby et al. (2018) is that rolling stock planning "is one area of railway planning in which the robustness introduced in the earlier planning problems, in particular line planning and timetabling, can be adversely affected." This motivates the research on the integration of timetabling and rolling stock scheduling decisions, to better balance the buffers required to handle disruptions.

To our knowledge, Dauzère-Pérès et al. (2015) are the only ones to have tackled the integration of rolling stock scheduling and crew scheduling decisions. A Lagrangian relaxation heuristic is proposed, and promising results are obtained on real-life instances of a French region.

6 Conclusions and directions for future research

In this paper, we looked back at 50 years of applying Operations Research methods in railway planning. The two most important and domain specific planning problems in railways are timetabling and rolling stock planning. We classified the literature in railway timetabling in decision levels (strategic, tactical and operational) and by the type of network infrastructure. For the latter, we consider two extreme cases, namely independent routes where the timetable of one route is completely independent of the other routes like in a star network and dependent routes, where complex railway networks with many inter-dependencies are considered. In these networks, a distinction between network level and station level is necessary to be able to solve real-world instances. Moreover, such networks often consider cyclic timetables. We think that this distinction is not only useful for classifying the literature as we did, but also from other perspectives. For instance, when introducing new European legislation the topology of the network in different countries could be considered. Moreover, we considered the state-of-the-art in rolling stock planning including its link with infrastructure planning in the rolling stock stabling problem.

Both timetabling and rolling stock planning received much attention since the 1990s, when the first optimization models and solution methods for these problems were developed. These initial approaches covered the tactical level, i.e. construction of the annual timetable or rolling stock plan. At that time, only small-scale instances could be solved.

In the first decade of the 21st century some major achievements have been accomplished, for instance the mathematical model used to construct the timetable of the Berlin subway in December 2004 and the use of OR methods for both timetabling and rolling stock planning to develop and introduce the completely new timetable in the Netherlands in December 2006.

In the last 10 years, new models and solution methods have been introduced in the literature to tackle the strategic and operational planning level. However, to the best of our knowledge, these methods have hardly

been applied in practice, in particular, in the context of complex networks with dependent routes. Therefore, we consider this as a major challenge for the European railway sector.

Optimization methods on the strategic level are becoming more important, because if we look at the coming decades where on one hand rail plays a major role in the European Green Deal, but on the other hand financial sources for large investments in the railway infrastructure are limited, an efficient use of the railway infrastructure is required.

At the operational level, many challenges are resulting from the ageing infrastructure that needs a lot of maintenance in the coming years.

Other important topics for future research are the further evaluation and development of cross-links between network timetabling and station planning in complex networks, and the integrating of different planning problems. Especially, the integration of timetabling and rolling stock planning has the potential to reduce the capital and operational costs of European railways. The split in many European countries between infrastructure manager and railway undertaking makes this also an organizational challenge. Fairness considerations and game-theoretic approaches could therefore be interesting research directions as well.

Acknowledgements

Most research leading to this paper has been part of EU-Rail flagship project MOTIONAL, funded by the European Union under Grant Agreement 101101973 (call HORIZON-ER-JU-2022-01).

References

Abbink, E., Van den Berg, B., Kroon, L., & Salomon, M. (2004). Allocation of railway rolling stock for passenger trains. *Transportation Science*, *38*(1), 33–41.

Albrecht, A. R., Panton, D. M., & Lee, D. H. (2013). Rescheduling rail networks with maintenance disruptions using problem space search. *Computers & Operations Research*, *40*(3), 703–712.

Alfieri, A., Groot, R., Kroon, L., & Schrijver, A. (2006). Efficient circulation of railway rolling stock. *Transportation Science*, *40*(3), 378–391.

Bababeik, M., Zerguini, S., Farjad-Amin, M., Khademi, N., & Bagheri, M. (2019). Developing a train timetable according to track maintenance plans: A stochastic optimization of buffer time schedules. *Transportation Research Procedia*, *37*, 27–34.

Bešinović, N., & Goverde, R. M. (2019). Stable and robust train routing in station areas with balanced infrastructure capacity occupation. *Public Transport*, *11*, 211–236.

Bešinović, N., Widarno, B., & Goverde, R. M. (2020). Adjusting freight train paths to infrastructure possessions. *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 1–6.

Borndörfer, R., Eßer, T., Frankenberger, P., Huck, A., Jobmann, C., Krostitz, B., Kuchenbecker, K., Mohrhagen, K., Nagl, P., Peterson, M., et al. (2021). Deutsche bahn schedules train rotations using hypergraph optimization. *INFORMS Journal on Applied Analytics*, *51*(1), 42–62.

Borndörfer, R., Reuther, M., Schlechte, T., Waas, K., & Weider, S. (2016). Integrated optimization of rolling stock rotations for intercity railways. *Transportation Science*, *50*(3), 863–877.

Brännlund, U., Lindberg, P. O., Nou, A., & Nilsson, J.-E. (1998). Railway timetabling using lagrangian relaxation. *Transportation Science*, *32*(4), 358–369.

Burdett, R. L., & Kozan, E. (2009). Techniques for inserting additional trains into existing timetables. *Transportation Research Part B: Methodological*, *43*(8-9), 821–836.

Burggraeve, S., Bull, S. H., Vansteenwegen, P., & Lusby, R. M. (2017). Integrating robust timetabling in line plan optimization for railway systems. *Transportation Research Part C: Emerging Technologies*, *77*, 134–160.

Burggraeve, S., & Vansteenwegen, P. (2017). Robust routing and timetabling in complex railway stations. *Transportation Research Part B: Methodological*, *101*, 228–244.

Burkolter, D. (2005). *Capacity of railways in station areas using petri nets* [Doctoral dissertation, ETH Zurich].

Cacchiani, V., Caprara, A., Galli, L., Kroon, L., Maróti, G., & Toth, P. (2012). Railway rolling stock planning: Robustness against large disruptions. *Transportation Science*, *46*(2), 217–232.

Cacchiani, V., Caprara, A., & Toth, P. (2008). A column generation approach to train timetabling on a corridor. *4OR*, *6*, 125–142.

Cacchiani, V., Caprara, A., & Toth, P. (2019). An effective peak period heuristic for railway rolling stock planning. *Transportation Science*, *53*(3), 746–762.

Cacchiani, V., Caprara, A., & Toth, P. (2013). A lagrangian heuristic for a train-unit assignment problem. *Discrete Applied Mathematics*, *161*(12), 1707–1718.

Cacchiani, V., Caprara, A., & Toth, P. (2010a). Scheduling extra freight trains on railway networks. *Transportation Research Part B: Methodological*, *44*(2), 215–231.

Cacchiani, V., Caprara, A., & Toth, P. (2010b). Solving a real-world train-unit assignment problem. *Mathematical Programming*, *124*, 207–231.

Cacchiani, V., Galli, L., & Toth, P. (2015). A tutorial on non-periodic train timetabling and platforming problems. *Euro Journal on Transportation and logistics*, *4*(3), 285–320.

Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., & Wagenaar, J. (2014). An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, *63*, 15–37. https://doi.org/https://doi.org/10.1016/j.trb.2014.01.009

Cacchiani, V., Qi, J., & Yang, L. (2020). Robust optimization models for integrated train stop planning and timetabling with passenger demand uncertainty. *Transportation Research Part B: Methodological*, *136*, 1–29.

Cacchiani, V., & Toth, P. (2012). Nominal and robust train timetabling problems. *European Journal of Operational Research*, *219*(3), 727–737.

Caimi, G., Burkolter, D., & Herrmann, T. (2005). Finding delay-tolerant train routings through stations. *Operations Research Proceedings 2004: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR). Jointly Organized with the Netherlands Society for Operations Research (NGB) Tilburg, September 1–3, 2004*, 136–143.

Caimi, G., Chudak, F., Fuchsberger, M., Laumanns, M., & Zenklusen, R. (2011). A new resource-constrained multicommodity flow model for conflict-free train routing and scheduling. *Transportation Science*, *45*(2), 212–227.

Caprara, A., Fischetti, M., & Toth, P. (2002). Modeling and solving the train timetabling problem. *Operations Research*, *50*(5), 851–861.

Caprara, A., Galli, L., Kroon, L., Maróti, G., & Toth, P. (2010). Robust train routing and online re-scheduling. *10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'10)*.

Caprara, A., Galli, L., Stiller, S., & Toth, P. (2014). Delay-robust event scheduling. *Operations Research*, *62*(2), 274–283.

Caprara, A., Galli, L., & Toth, P. (2011). Solution of the train platforming problem. *Transportation Science*, *45*(2), 246–257.

Caprara, A., Monaci, M., Toth, P., & Guida, P. L. (2006). A lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Applied Mathematics*, *154*(5), 738–753.

Carey, M., & Lockwood, D. (1995). A model, algorithms and strategy for train pathing. *Journal of the Operational Research Society*, *46*(8), 988–1005.

Cicerone, S., D'Angelo, G., Di Stefano, G., Frigioni, D., Navarra, A., Schachtebeck, M., & Schöbel, A. (2009). Recoverable robustness in shunting and timetabling. *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*, 28–60.

Cordeau, J.-F., Desaulniers, G., Lingaya, N., Soumis, F., & Desrosiers, J. (2001). Simultaneous locomotive and car assignment at via rail canada. *Transportation Research Part B: Methodological*, *35*(8), 767–787.

Cordeau, J.-F., Soumis, F., & Desrosiers, J. (2000). A benders decomposition approach for the locomotive and car assignment problem. *Transportation Science*, *34*(2), 133–149.

Cordeau, J.-F., Soumis, F., & Desrosiers, J. (2001). Simultaneous assignment of locomotives and cars to passenger trains. *Operations Research*, *49*(4), 531–548.

Corman, F., & Meng, L. (2015). A review of online dynamic models and algorithms for railway traffic management. *IEEE Trans. Intell. Transp. Syst.*, *16*(3), 1274–1284. https://doi.org/10.1109/TITS.2014.2358392

Correia Duarte, P. J., Schmidt, M., Huisman, D., & Veelenturf, L. P. (2023). Fewer trains for better timetables: The price of fixed line frequencies in the passenger-oriented timetabling problem. *23rd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2023)*.

Coviello, N., Medeossi, G., Nash, A., Nygreen, T., Pellegrini, P., & Rodriguez, J. (2023). Multiobjective timetable development tool for railway strategic planning in norway. *Transportation Research Record*, *2677*(1), 720–729.

D'Ariano, A., Meng, L., Centulio, G., & Corman, F. (2019). Integrated stochastic optimization approaches for tactical scheduling of trains and railway infrastructure maintenance. *Computers & Industrial Engineering*, *127*, 1315–1335.

Dauzère-Pérès, S., De Almeida, D., Guyon, O., & Benhizia, F. (2015). A lagrangian heuristic framework for a real-life integrated planning problem of railway transportation resources. *Transportation Research Part B: Methodological*, *74*, 138–150.

de Graaf, Y. (2021). *Strategic passenger-oriented timetable design: Long-term timetable designs with minimised passenger inconvenience* [Master's thesis, KTH Royal Institute of Technology].

Delorme, X., Rodriguez, J., & Gandibleux, X. (2001). Heuristics for railway infrastructure saturation. *Electronic Notes in Theoretical Computer Science*, *50*(1), 39–53.

Dewilde, T., Sels, P., Cattrysse, D., & Vansteenwegen, P. (2014). Improving the robustness in railway station areas. *European Journal of Operational Research*, *235*(1), 276–286.

Dewilde, T., Sels, P., Cattrysse, D., & Vansteenwegen, P. (2013). Robust railway station planning: An interaction between routing, timetabling and platforming. *Journal of Rail Transport Planning & Management*, *3*(3), 68–77.

Erlandson, W., Häll, C. H., Peterson, A., & Schmidt, C. (2023). Meta-heuristic for inserting a robust train path in a non-cyclic timetable. *Transportation Planning and Technology*, *46*(7), 842–863.

Fang, W., Yang, S., & Yao, X. (2015). A survey on problem models and solution approaches to rescheduling in railway networks. *IEEE transactions on intelligent transportation systems*, *16*(6), 2997–3016.

Farina, F. (2018). *Optimization of operations in public transportation* [Doctoral dissertation, Technical University of Denmark].

Fioole, P.-J., Kroon, L., Maróti, G., & Schrijver, A. (2006). A rolling stock circulation model for combining and splitting of passenger trains. *European Journal of Operational Research*, *174*(2), 1281–1297.

Fischer, F. (2015). Ordering constraints in time expanded networks for train timetabling problems. *15th workshop on algorithmic approaches for transportation modelling, optimization, and systems (ATMOS 2015)*.

Fischer, F., Helmberg, C., Janßen, J., & Krostitz, B. (2008). Towards solving very large scale train timetabling problems by lagrangian relaxation. *8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'08)*.

Fischetti, M., Salvagnin, D., & Zanette, A. (2009). Fast approaches to improve the robustness of a railway timetable. *Transportation Science*, *43*(3), 321–335.

Ford, L. R., & Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, *8*, 399–404.

Forsgren, M., Aronsson, M., & Gestrelius, S. (2013). Maintaining tracks and traffic flow at the same time. *Journal of Rail Transport Planning & Management*, *3*(3), 111–123.

Freling, R., Lentink, R. M., Kroon, L., & Huisman, D. (2005). Shunting of passenger train units in a railway station. *Transportation Science*, *39*(2), 261–272.

Friesen, N., Sander, T., Büsing, C., Nachtigall, K., & Nießen, N. (2024). The complexity of the timetable-based railway network design problem. *Networks*, *83*(2), 289–299.

Fuegenschuh, A., Homfeld, H., Huck, A., Martin, A., & Yuan, Z. (2008). Scheduling locomotives and car transfers in freight transport. *Transportation Science*, *42*(4), 478–491.

Gattermann, P., Großmann, P., Nachtigall, K., & Schöbel, A. (2016). Integrating passengers' routes in periodic timetabling: A sat approach. *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*.

Gestrelius, S., Aronsson, M., & Peterson, A. (2017). A milp-based heuristic for a commercial train timetabling problem. *Transportation Research Procedia*, *27*, 569–576.

Goerigk, M., Schachtebeck, M., & Schöbel, A. (2013). Evaluating line concepts using travel times and robustness: Simulations with the lintim toolbox. *Public Transport*, *5*, 267–284.

Goerigk, M., & Schöbel, A. (2014). Recovery-to-optimality: A new two-stage approach to robustness with an application to aperiodic timetabling. *Computers & Operations Research*, *52*, 1–15.

Großmann, P., Hölldobler, S., Manthey, N., Nachtigall, K., Opitz, J., & Steinke, P. (2012). Solving periodic event scheduling problems with sat. *Advanced Research in Applied Artificial Intelligence: 25th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2012, Dalian, China, June 9-12, 2012. Proceedings 25*, 166–175.

Harris, T., & Ross, F. (1955). *Fundamentals of a method for evaluating rail net capacities* (tech. rep.). Rand Corporation.

Harrod, S. (2011). Modeling network transition constraints with hypergraphs. *Transportation Science*, *45*(1), 81–97.

Harrod, S. (2012). A tutorial on fundamental model structures for railway timetable optimization. *Surveys in Operations Research and Management Science*, *17*, 85–96.

Heil, J., Hoffmann, K., & Buscher, U. (2020). Railway crew scheduling: Models, methods and applications. *European Journal of Operational Research*, *283*(2), 405–425.

Higgins, A., Ferreira, L., & Lake, M. (1999). Scheduling rail track maintenance to minimise overall delays. *Transportation And Traffic Theory: Proceedings of the 14th International Symposium on Transportation and Traffic Theory*, 779–796.

Higgins, A., Kozan, E., & Ferreira, L. (1997). Heuristic techniques for single line train scheduling. *Journal of Heuristics*, *3*, 43–62.

INFORMS. (2012). Ras problem solving competition 2012 archive. Retrieved October 31, 2023, from https://connect.informs.org/railway-applications/new-item3/problem-solving-competition681/new-item6

Ingolotti, L., Barber, F., Tormos, P., Lova, A., Salido, M. A., & Abril, M. (2004). An efficient method to schedule new trains on a heavily loaded railway network. *Advances in Artificial Intelligence–IBERAMIA 2004: 9th Ibero-American Conference on AI, Puebla, Mexico, November 22-26, 2004. Proceedings 9*, 164–173.

International Union of Railways. (2013). UIC Code 406.

Ireland, P., Case, R., Fallis, J., Van Dyke, C., Kuehn, J., & Meketon, M. (2004). The canadian pacific railway transforms operations by using models to develop its operating plans. *Interfaces*, *34*(1), 5–14.

Jensen, L. W., Landex, A., Nielsen, O. A., Kroon, L., & Schmidt, M. (2017). Strategic assessment of capacity consumption in railway networks: Framework and model. *Transportation Research Part C: Emerging Technologies*, *74*, 126–149.

Jovanović, D., & Harker, P. T. (1991). Tactical scheduling of rail operations: The scan i system. *Transportation Science*, *25*(1), 46–64.

Jovanović, P., Pavlović, N., Belošević, I., & Milinković, S. (2020). Graph coloring-based approach for railway station design analysis and capacity determination. *European Journal of Operational Research*, *287*(1), 348–360.

Kloster, O., Luteberget, B., Mannino, C., & Sartor, G. (2023). An optimization-based decision support tool for incremental train timetabling. *Operations Research Forum*, *4*.

Kroon, L., 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., Lentink, R. M., & Schrijver, A. (2008). Shunting of passenger train units: An integrated approach. *Transportation Science*, *42*(4), 436–449.

Kroon, L., Maróti, G., Helmrich, M. R., Vromans, M., & Dekker, R. (2008). Stochastic improvement of cyclic railway timetables. *Transportation Research Part B: Methodological*, *42*(6), 553–570.

Kroon, L., Maróti, G., & Nielsen, L. (2015). Rescheduling of railway rolling stock with dynamic passenger flows. *Transportation Science*, *49*(2), 165–184.

Kroon, L., & Peeters, L. W. (2003). A variable trip time model for cyclic railway timetabling. *Transportation science*, *37*(2), 198–212.

Lamorgese, L., Mannino, C., & Natvig, E. (2017). An exact micro–macro approach to cyclic and non-cyclic train timetabling. *Omega*, *72*, 59–70.

Lamorgese, L., Mannino, C., Pacciarelli, D., & Krasemann, J. T. (2018). Train dispatching. *Handbook of Optimization in the Railway Industry*, 265–283.

Lidén, T. (2015). Railway infrastructure maintenance-a survey of planning problems and conducted research. *Transportation Research Procedia*, *10*, 574–583.

Lidén, T., & Joborn, M. (2017). An optimization model for integrated planning of railway traffic and network maintenance. *Transportation Research Part C: Emerging Technologies*, *74*, 327–347.

Liebchen, C. (2006). *Periodic timetable optimization in public transport* [Doctoral dissertation, Technical University Berlin].

Liebchen, C. (2008). The first optimized railway timetable in practice. *Transportation Science*, *42*(4), 420–435.

Liebchen, C., Lübbecke, M., Möhring, R., & Stiller, S. (2009). The concept of recoverable robustness, linear programming recovery, and railway applications. *Robust and online large-scale optimization: Models and techniques for transportation systems*, 1–27.

Liebchen, C., & Möhring, R. H. (2007). The modeling power of the periodic event scheduling problem: Railway timetables—and beyond. *Algorithmic Methods for Railway Optimization: International Dagstuhl Workshop, Dagstuhl Castle, Germany, June 20-25, 2004, 4th International Workshop, ATMOS 2004, Bergen, Norway, September 16-17, 2004, Revised Selected Papers*, 3–40.

Lindner, N., & Liebchen, C. (2019). New perspectives on pesp: T-partitions and separators. *19th Symposium on algorithmic approaches for transportation modelling, optimization, and systems (ATMOS 2019)*.

Lindner, N., & Liebchen, C. (2022). Timetable merging for the periodic event scheduling problem. *EURO Journal on Transportation and Logistics*, *11*, 100081.

Ljunggren, F., Persson, K., Peterson, A., & Schmidt, C. (2021). Railway timetabling: A maximum bottleneck path algorithm for finding an additional train path. *Public Transport*, *13*, 597–623.

Luan, X., Miao, J., Meng, L., Corman, F., & Lodewijks, G. (2017). Integrated optimization on train scheduling and preventive maintenance time slots planning. *Transportation Research Part C: Emerging Technologies*, *80*, 329–359.

Lusby, R. M., Larsen, J., & Bull, S. (2018). A survey on robustness in railway planning. *European Journal of Operational Research*, *266*(1), 1–15.

Lusby, R. M., Larsen, J., Ehrgott, M., & Ryan, D. (2011). Railway track allocation: Models and methods. *OR spectrum*, *33*, 843–883.

Lusby, R. M., Larsen, J., Ryan, D., & Ehrgott, M. (2011). Routing trains through railway junctions: A new set-packing approach. *Transportation Science*, *45*(2), 228–245.

Martin-Iradi, B., & Ropke, S. (2022). A column-generation-based matheuristic for periodic and symmetric train timetabling with integrated passenger routing. *European Journal of Operational Research*, *297*(2), 511–531.

Mascis, A., & Pacciarelli, D. (2002). Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, *143*(3), 498–517.

Matos, G. P., Albino, L. M., Saldanha, R. L., & Morgado, E. M. (2021). Solving periodic timetabling problems with sat and machine learning. *Public Transport*, *13*(3), 625–648.

Michaelis, M., & Schöbel, A. (2009). Integrating line planning, timetabling, and vehicle scheduling: A customer-oriented heuristic. *Public Transport*, *1*, 211–232.

Nachtigall, K. (1994). *A branch and cut approach for periodic network programming*. Univ., Inst. für Mathematik.

Nielsen, L. K., Kroon, L., & Maróti, G. (2012). A rolling horizon approach for disruption management of railway rolling stock. *European Journal of Operational Research*, *220*(2), 496–509.

Odijk, M. A. (1996). A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B: Methodological*, *30*(6), 455–464.

Ortiz-Astorquiza, C., Cordeau, J.-F., & Frejinger, E. (2021). The locomotive assignment problem with distributed power at the canadian national railway company. *Transportation Science*, *55*(2), 510–531.

Pachl, J. (2018). *Railway operation and control*. VTD Rail Publishing.

Peeters, L. (2003). *Cyclic railway timetable optimization* (Publication No. EPS-2003-022-LIS) [Doctoral dissertation, Erasmus Univeristy Rotterdam].

Pellegrini, P., Marlière, G., & Rodriguez, J. (2014). Optimal train routing and scheduling for managing traffic perturbations in complex junctions. *Transportation Research Part B: Methodological*, *59*, 58–80.

Pellegrini, P., Marlière, G., & Rodriguez, J. (2017). Recife-sat: A milp-based algorithm for the railway saturation problem. *Journal of Rail Transport Planning & Management*, *7*(1-2), 19–32.

Polinder, G.-J., Cacchiani, V., Schmidt, M., & Huisman, D. (2022). An iterative heuristic for passenger-centric train timetabling with integrated adaption times. *Computers & Operations Research*, *142*, 105740.

Polinder, G.-J., Schmidt, M., & Huisman, D. (2021). Timetabling for strategic passenger railway planning. *Transportation Research Part B: Methodological*, *146*, 111–135.

Quiroga, L. M., & Schnieder, E. (2010). A heuristic approach to railway track maintenance scheduling. *WIT Transactions on The Built Environment*, *114*, 687–699.

Robenek, T., Azadeh, S. S., Maknoon, Y., & Bierlaire, M. (2017). Hybrid cyclicity: Combining the benefits of cyclic and non-cyclic timetables. *Transportation Research Part C: Emerging Technologies*, *75*, 228–253.

Rouillon, S., Desaulniers, G., & Soumis, F. (2006). An extended branch-and-bound method for locomotive assignment. *Transportation Research Part B: Methodological*, *40*(5), 404–423.

Sartor, G., Mannino, C., Nygreen, T., & Bach, L. (2023). A MILP model for quasi-periodic strategic train timetabling. *Omega*, *116*, 102798.

Schmidt, M., Weik, N., Zieger, S., Schmeink, A., & Nießen, N. (2019). A generalized stochastic petri net model for performance analysis of trackside infrastructure in railway station areas under uncertainty. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 3732–3737.

Schöbel, A. (2017). An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation. *Transportation Research Part C: Emerging Technologies*, *74*, 348–365.

Schrijver, A. (2002). On the history of the transportation and maximum flow problems. *Mathematical Programming*, *91*, 437–225.

Schrijver, A., & Steenbeek, A. (1993). Dienstregelingontwikkeling voor nederlandse spoorwegen ns rapport fase 1. *Centrum voor Wiskunde en Informatica (Oktober 1993)*.

Sedghi, M., Kauppila, O., Bergquist, B., Vanhatalo, E., & Kulahci, M. (2021). A taxonomy of railway track maintenance planning and scheduling: A review and research trends. *Reliability Engineering & System Safety*, *215*, 107827.

Sels, P., Vansteenwegen, P., Dewilde, T., Cattrysse, D., Waquet, B., & Joubert, A. (2014). The train platforming problem: The infrastructure management company perspective. *Transportation Research Part B: Methodological*, *61*, 55–72.

Serafini, P., & Ukovich, W. (1989). A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, *2*(4), 550–581.

Sousa, J. P., & Wolsey, L. A. (1992). A time indexed formulation of non-preemptive single machine scheduling problems. *Mathematical Programming*, *54*, 353–367.

Tan, Y., Li, Y.-X., & Wang, R.-X. (2020). Scheduling extra train paths into cyclic timetable based on the genetic algorithm. *IEEE Access*, *8*, 102199–102211.

Tan, Y., Xu, W., Jiang, Z., Wang, Z., & Sun, B. (2021). Inserting extra train services on high-speed railway. *Periodica Polytechnica Transportation Engineering*, *49*(1), 16–24.

Tolstoı, A. (1939). Metody ustraneniya neratsional'nykh perevozok pri planirovanii [russian; methods of removing irrational transportation in planning]. *Sotsialisticheskiı Transport*, *9*, 28–51.

Tréfond, S., Billionnet, A., Elloumi, S., Djellab, H., & Guyon, O. (2017). Optimization and simulation for robust railway rolling-stock planning. *Journal of Rail Transport Planning & Management*, *7*(1-2), 33–49.

Vaidyanathan, B., Ahuja, R. K., Liu, J., & Shughart, L. A. (2008). Real-life locomotive planning: New formulations and computational results. *Transportation Research Part B: Methodological*, *42*(2), 147–168.

Vaidyanathan, B., Ahuja, R. K., & Orlin, J. B. (2008). The locomotive routing problem. *Transportation Science*, *42*(4), 492–507.

Van Aken, S., Bešinović, N., & Goverde, R. M. (2017a). Designing alternative railway timetables under infrastructure maintenance possessions. *Transportation Research Part B: Methodological*, *98*, 224–238.

Van Aken, S., Bešinović, N., & Goverde, R. M. (2017b). Solving large-scale train timetable adjustment problems under infrastructure maintenance possessions. *Journal of Rail Transport Planning & Management*, *7*(3), 141–156.

van den Broek, R., Hoogeveen, H., van den Akker, M., & Huisman, B. (2022). A local search algorithm for train unit shunting with service scheduling. *Transportation Science*, *56*(1), 141–161.

van der Aalst, W., & Odijk, M. (1995). Analysis of railway stations by means of interval timed coloured petri nets. *Real-Time Systems*, *9*, 241–263.

Weik, N., Warg, J., Johansson, I., Bohlin, M., & Nießen, N. (2020). Extending UIC 406-based capacity analysis–new approaches for railway nodes and network effects. *Journal of Rail Transport Planning & Management*, *15*, 100199.

Xu, X., Li, C.-L., & Xu, Z. (2018). Integrated train timetabling and locomotive assignment. *Transportation Research Part B: Methodological*, *117*, 573–593.

Yan, F., Bešinović, N., & Goverde, R. M. (2019). Multi-objective periodic railway timetabling on dense heterogeneous railway corridors. *Transportation Research Part B: Methodological*, *125*, 52–75.

Yan, F., & Goverde, R. M. (2019). Combined line planning and train timetabling for strongly heterogeneous railway lines with direct connections. *Transportation Research Part B: Methodological*, *127*, 20–46.

Ziarati, K., Soumis, F., Desrosiers, J., Gélinas, S., & Saintonge, A. (1997). Locomotive assignment with heterogeneous consists at cn north america. *European Journal of Operational Research*, *97*(2), 281–292.

Ziarati, K., Soumis, F., Desrosiers, J., & Solomon, M. M. (1999). A branch-first, cut-second approach for locomotive assignment. *Management Science*, *45*(8), 1156–1168.

Zwaneveld, P. J., Kroon, L., Romeijn, H. E., Salomon, M., Dauzere-Peres, S., Van Hoesel, S. P., & Ambergen, H. W. (1996). Routing trains through railway stations: Model formulation and algorithms. *Transportation science*, *30*(3), 181–194.

Zwaneveld, P. J., Kroon, L., & Van Hoesel, S. P. (2001). Routing trains through a railway station based on a node packing model. *European Journal of Operational Research*, *128*(1), 14–33.

12.2. Appendix B

It provides additional information to Section 6.1.

Long-term timetabling in Task 6.2 in WP6

APPENDIX

October 7, 2024

1 Extended periodic event-activity network

This section describes the extended periodic event-activity network and the notations used to model the problem. We follow the extended event-activity network construction method proposed in Schiewe and Schöbel (2020). Specifically, the railway operations at the macroscopic level and passenger dynamics are modeled as a directed graph $N = (V, A)$, in which V is the set of nodes (events) and A is the set of arcs (activities). The nodes represent train arrival and departure events with a periodicity $T \in \mathbb{N}$, as well as the start and end events for passenger boarding and alighting. The activities include driving, dwelling, transfer, and additional activities, where additional activities link the start nodes to possible departure events and the end nodes to possible arrival events. For a detailed introduction to the extended event-activity network, we refer to Schiewe and Schöbel (2020).

Our goal is to assign a time $\pi_i \in \mathbb{Z}$ to each event $i \in V$, such that the duration of activities $y_a \ \forall a \in A$ are within the desired time interval $[l_a, u_a]$. The optimal timetable is the one that minimizes passengers' perceived travel costs, under the assumption that passengers choose the route with the shortest perceived travel costs under that timetable. To integrate the passenger routing into timetabling, let an OD-matrix $\mathcal{OD}$ be given, providing for each OD-pair $k \in \mathcal{OD}$ an estimate of the average number of passengers d_k that want to travel from the origin to the corresponding destination per period.

2 Model formulations

2.1 Passenger routing and perceived travel time

We denote the set of routes for each OD pair $k \in \mathcal{OD}$ as $\mathcal{R}^k$, where each route $r \in R := \bigcup_{k \in \mathcal{OD}} \mathcal{R}^k$ is a simple directed path through the event-activity network N. Each route r is an ordered subset of A, and the perceived travel time of this route (denoted as $Y_r(\pi)$) is the weighted sum of times on the included activities, the adaption time between the desired departure time t and the first event $\sigma(r)$ on route r (denoted as $\mathrm{at}_r^t(\pi)$), the penalty γ_w that models the perception of adaption time in relation to perceived duration of the route, and the penalty γ_t for each transfer due to the inconvenience. Hence, the perceived travel time on route r under timetable π is formulated as follows:

$$\hat{Y}_r(\pi) := \sum_{a \in r} y_a + \gamma_w \cdot \mathrm{at}_r^t(\pi) + \gamma_t \cdot 1_t(a), \tag{1}$$

where the function $1_t(a)$ is an indicator function to indicate whether activity $a \in A$ is a transfer activity or not.

The adaption time can be expressed as follows:

$$\mathrm{at}_r^t(\pi) := \left(\pi_{\sigma(r)} - t\right) \bmod T. \tag{2}$$

Under the assumption that passenger demand is distributed uniformly over the period, and that each passenger chooses a route which minimizes his perceived travel time, we can thus formulate the objective function as minimizing the total perceived travel time of passengers under a timetable π, that is

$$tt(\pi) := \sum_{k \in \mathcal{OD}} \sum_{t=1}^{T} \frac{d_k}{T} \cdot \min_{r \in \mathcal{R}^k} \hat{Y}_r(\pi). \tag{3}$$

2.2 Integer nonlinear programming formulation

We introduce an integer decision variable p_{ij} to represent the periodical offset of activity. We can now formalize the periodic timetabling with integrated passenger routing problem as follows:

$$\text{minimize} \quad \sum_{k \in \mathcal{OD}} \frac{d_k}{T} \sum_{v \in V^k} L_v^k \cdot (\gamma_w \cdot W_v^k + Y_v^k) \tag{4a}$$

$$\text{subject to} \quad y_{ij} = \pi_j - \pi_i + T p_{ij} \qquad \forall (i,j) \in A, \tag{4b}$$

$$l_{ij} \le y_{ij} \le u_{ij} \qquad \forall (i,j) \in A, \tag{4c}$$

$$Y_r = \sum_{a \in r} \Big(y_a + \gamma_t \cdot 1_t(a) \Big) \qquad \forall r \in \mathcal{R}, \tag{4d}$$

$$L_v^k = \min_{v' \in V^k \setminus \{v\}} \Big\{ \pi_v - \pi_{v'} + T \alpha_{v',v} \Big\} \qquad \forall k \in \mathcal{OD}, v \in V^k, \tag{4e}$$

$$\alpha_{v,v'} + \alpha_{v',v} = 1 \qquad \forall k \in \mathcal{OD}, v \in V^k, v' \in V^k \setminus \{v\}, \tag{4f}$$

$$W_v^k = \frac{1}{2} L_v^k \qquad \forall k \in \mathcal{OD}, v \in V^k, \tag{4g}$$

$$Y_v^k = \min_{v' \in V^k} \min_{r \in \mathcal{R}_{v'}^k} \Big\{ Y_r + \gamma_w \cdot (\pi'_v - \pi_v + T \alpha_{v,v'}) \Big\} \qquad \forall k \in \mathcal{OD}, v \in V^k, \tag{4h}$$

$$L_v^k \in [0, T] \qquad \forall k \in \mathcal{OD}, v \in V^k, \tag{4i}$$

$$W_v^k \in [0, T/2] \qquad \forall k \in \mathcal{OD}, v \in V^k, \tag{4j}$$

$$Y_r, Y_v^k \in \mathbb{R}_{\ge 0} \qquad \forall r \in \mathcal{R}, k \in \mathcal{OD}, v \in V^k, \tag{4k}$$

$$\pi_v \in \{0, \dots, T-1\} \qquad \forall v \in V, \tag{4l}$$

$$p_{ij} \in \{0, 1\} \qquad \forall (i,j) \in A, \tag{4m}$$

$$\alpha_{v,v'} \in \{0, 1\} \qquad \forall k \in \mathcal{OD}, v \in V^k, v' \in V^k \setminus \{v\}. \tag{4n}$$

Objective function (4a) aims to minimize the total perceived travel time of passengers. Constraints (4b) and (4c) are the timetabling constraints, which are formulated to ensure

the periodicity of generated timetables. Constraints (4d) determine the perceived duration of each route under timetable π. Constraints (4e) determines the length of time slice S_v^k, as the time distance to the previous periodic departure event in V^k. Hereby, constraints (4f) are required to impose an order between events, even when two departures happen at the same time. Constraints (4g) and (4h) determine average waiting time until the end of time slice S_v^k and perceived travel time on an optimal route from the end of time slice S_v^k, respectively. Lastly, constraints (4i) - (4n) define the domains of the variables.

2.3 Linearization

Note that the proposed model (4) is nonlinear, where the objective function (4a) has quadratic terms, constraints (4e) and (4h) contain one or two minimums. To facilitate straightforward computations using the commercial solvers, we linearized the nonlinear constraints in this section.

First, to linearize the objective function, we introduce a binary variable $x_{v,d}^k$ for all $k \in \mathcal{OD}, v \in V^k, d \in \{1, 2, \cdots, T\}$ to indicate whether $L_v^k \geq d$ or not. If yes, this variable is equal to one; otherwise, zero. We also define a variable $R_{v,d}^k = Y_v^k \cdot x_{v,d}^k$ for all $k \in \mathcal{OD}, v \in V^k, d \in \{1, 2, \cdots, T\}$. On this basis, the linear form of the objective function can be expressed as follows:

$$\sum_{k \in \mathcal{OD}} \frac{d_k}{T} \sum_{v \in V^k} \sum_{d=1}^{T} \left[\frac{\gamma_w}{2} (2d - 1) \cdot x_{v,d}^k + R_{v,d}^k \right]. \tag{5}$$

and additional restrictions have to be added, i.e.,

$$x_{v,d}^k \leq x_{v,d-1}^k \qquad \forall k \in \mathcal{OD}, v \in V^k, d \in \{2, \cdots, T\}. \tag{6}$$

$$R_{v,d}^k \leq u_v^k \cdot x_{v,d}^k \qquad \forall k \in \mathcal{OD}, v \in V^k, d \in \{1, \cdots, T\}. \tag{7}$$

$$R_{v,d}^k \geq l_v^k \cdot x_{v,d}^k \qquad \forall k \in \mathcal{OD}, v \in V^k, d \in \{1, \cdots, T\}. \tag{8}$$

$$R_{v,d}^k \leq Y_v^k - l_v^k \cdot (1 - x_{v,d}^k) \qquad \forall k \in \mathcal{OD}, v \in V^k, d \in \{1, \cdots, T\}. \tag{9}$$

$$R_{v,d}^k \geq Y_v^k - u_v^k \cdot (1 - x_{v,d}^k) \qquad \forall k \in \mathcal{OD}, v \in V^k, d \in \{1, \cdots, T\}, \tag{10}$$

where l_v^k and u_v^k are the lowest and highest possible values for Y_v^k, respectively.

Thereafter, we define $Q_{v,v'}$ as the periodic time difference between events v and v', i.e.,

$$Q_{v,v'} = \pi_{v'} - \pi_v + T\alpha_{v,v'} \qquad \forall k \in \mathcal{OD}, v \in V^k, v' \in V^k \backslash \{v\}. \tag{11}$$

Then we can replace constraints (4e) by the following linear restrictions

$$L_v^k \leq Q_{v,v'} \qquad \forall k \in \mathcal{OD}, v \in V^k, v' \in V^k \backslash \{v\}. \tag{12}$$

$$\sum_{v \in V^k} L_v^k = T \qquad \forall k \in \mathcal{OD}. \tag{13}$$

Finally, to linearize constraints (4h), we define a binary variable $z^k_{v,v',r}$ which corresponds to the route that is chosen. If passengers wait from event v to v', and take route route r, starting at v', we have $z^k_{v,v',r} = 1$. Constraints (4h) can be replaced by the following set of linear restrictions:

$$Y^k_v \le Y_r + \gamma_w \cdot Q_{v,v'} \qquad \forall k \in \mathcal{OD}, v \in V^k, v' \in V^k, r \in \mathcal{R}^k_{v'}. \quad (14)$$

$$Y^k_v \ge Y_r + \gamma_w \cdot Q_{v,v'} - M^k_v \cdot (1 - z^k_{v,v',r}) \qquad \forall k \in \mathcal{OD}, v \in V^k, v' \in V^k, r \in \mathcal{R}^k_{v'}. \quad (15)$$

$$\sum_{v' \in V^k} \sum_{r \in \mathcal{R}^k_{v'}} z^k_{v,v',r} = 1 \qquad \forall k \in \mathcal{OD}, v \in V^k. \quad (16)$$

2.4 Integer linear programming formulation

We can now formalize the integer linear programming model as follows:

$$\text{minimize} \quad \sum_{k \in \mathcal{OD}} \frac{d_k}{T} \sum_{v \in V^k} \sum_{d=1}^{T} \left[\frac{\gamma_w}{2}(2d-1) \cdot x^k_{v,d} + R^k_{v,d} \right] \quad (17a)$$

$$\text{subject to} \quad (4b) - (4d), (4f) - (4g), (4i) - (4n), (6) - (16). \quad (17b)$$

3 Decomposition framework

We design a decomposition method that divides the integer linear programming model (17) into two subproblems. The core idea is to find a timetable that is feasible concerning the timetabling constraints, and then evaluate its quality through the passenger routing part. Specifically, the timetabling subproblem decides the integer variable $\boldsymbol{\pi}$ and generates a feasible train timetable. The passenger routing subproblem optimizes passengers' routes and evaluates the quality of the timetable generated by MP. For clarity, the formulation (17) is expressed as the following problem, i.e.,

$$\min_{\boldsymbol{\pi},\boldsymbol{\alpha},\mathbf{p},\mathbf{L},\mathbf{W},\mathbf{x},\mathbf{R},\mathbf{Q},\mathbf{z}} \left\{ (17a) \middle| f(\pi) \ge 0, g(\boldsymbol{\varsigma}) \ge 0, \boldsymbol{\pi} \in \Pi, \boldsymbol{\varsigma} \in \Xi \right\},$$

where $\boldsymbol{\varsigma} = \{(\boldsymbol{\alpha}, \mathbf{p}, \mathbf{L}, \mathbf{W}, \mathbf{x}, \mathbf{R}, \mathbf{Q}, \mathbf{z})\}$, $\Pi = \{\boldsymbol{\pi} \in \{0, 1, \cdots, T-1\} \mid \text{Eqs.}(4l)\}$, and $\Xi = \{\boldsymbol{\alpha}, \mathbf{p}, \mathbf{x}, \mathbf{z} \in \{0,1\}, \mathbf{L}, \mathbf{W}, \mathbf{R}, \mathbf{Q} \in \mathbb{Z}_{\ge 0} \mid \text{Eqs.}(4b), (4f), (4g), (4i), (4j), (4m), (4n), (6)-(13)\}$.

Specifically, let $\hat{\pi}$ denote the solution of the timetabling subproblem. The passenger routing subproblem can be formulated as an integer linear program, which can be expressed as follows:

$$\min_{\boldsymbol{\alpha},\boldsymbol{\varsigma}} \left\{ (17a) \middle| g(\hat{\pi}, \boldsymbol{\varsigma}) \ge 0, \boldsymbol{\varsigma} \in \Xi \right\}.$$

4 Strategies for updating operators' scores and weights in the ALNS algorithm

We define ϵ_k^d and ζ_k^d as the score and weight of the k-th operator, respectively. The weights are updated according to the formula:

$$\zeta_k^d = (1-\lambda)\zeta_k^d + \lambda \frac{\epsilon_k^d}{\sum_{k=1}^{3} \epsilon_k^d} \quad \forall k \in \{1,2,3\},$$

where $\lambda \in [0,1]$ acts as a scaling factor, which controls how sensitive the weights are to the changes in the operators' performance.

5 Information of the studied sub-network and analysis of passenger distribution

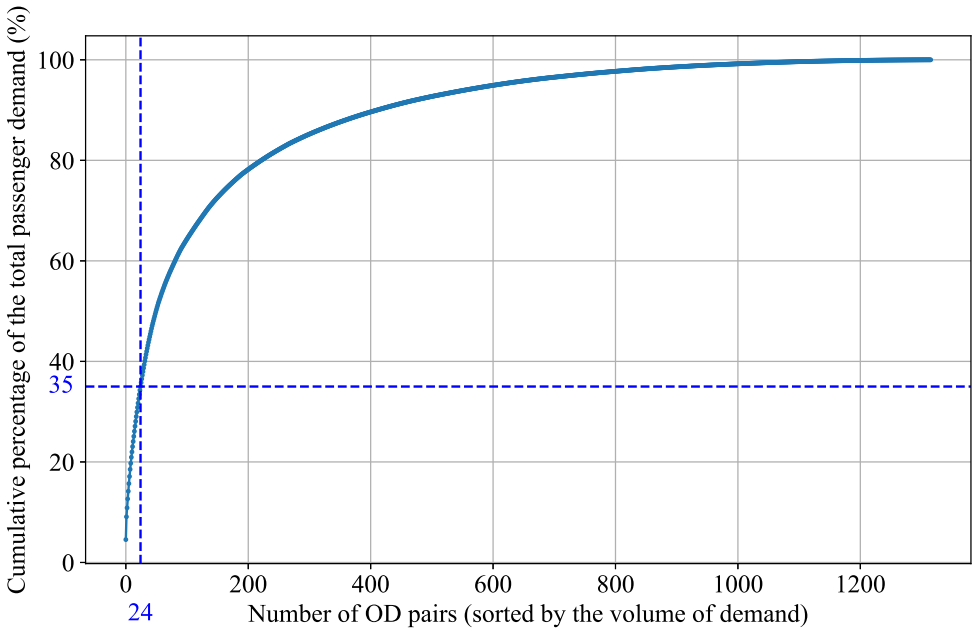
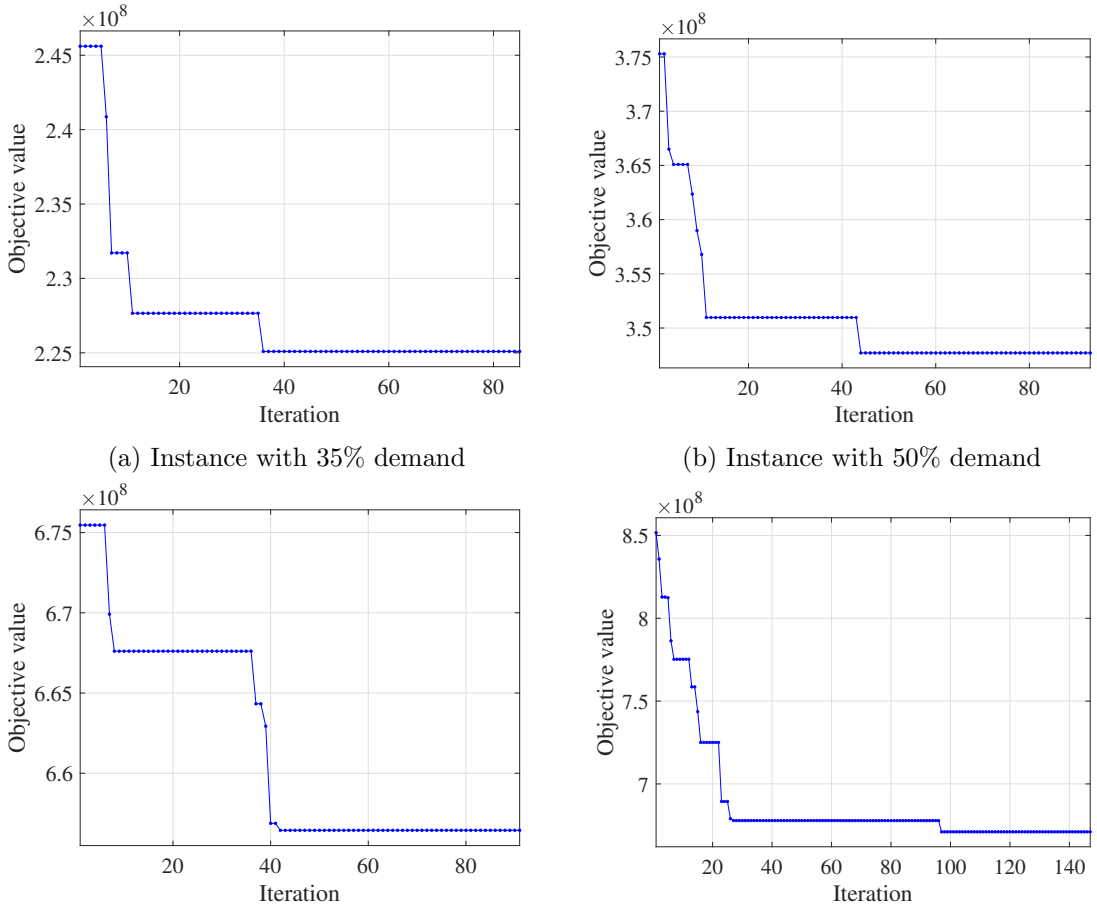


Figure 1: Cumulative percentage of the total passenger demand versus the number of OD pairs in the subnetwork.

Table 1: Line plan of the sub-network comprising four intercity lines.

Line index	Stations	Frequency
1	Vs, Mdb, Gs, Bgn, Rsd, Ddr, Rtd, Sdm, Gv, Laa, Ledn, Hlm, Asd	2
2	Nm, Ah, Ed, Db, Ut, Asa, Asd, Ass, Zd, Amr, Hwd, Sgn, Ana, Hdrz, Hdr	2
3	Gn, Asn, Zl, Amf, Ut, Gd, Rta, Rtd	1
4	Lw, Hr, Swk, Mp, Zl, Amf, Ut, Gd, Rta, Rtd	1

6 Detailed results of the case studies



(a) Instance with 35% demand

(b) Instance with 50% demand

(c) Instance with 80% demand

(d) Instance with all demand

Figure 2: The convergence curves of our proposed algorithm for instances based on the subnetwork with varying percentages of passenger demand.

References

Schiewe, P., Schöbel, A., 2020. Periodic timetabling with integrated routing: Toward applicable approaches. Transportation Science 54, 1714–1731. doi:10.1287/trsc.2019.0965.

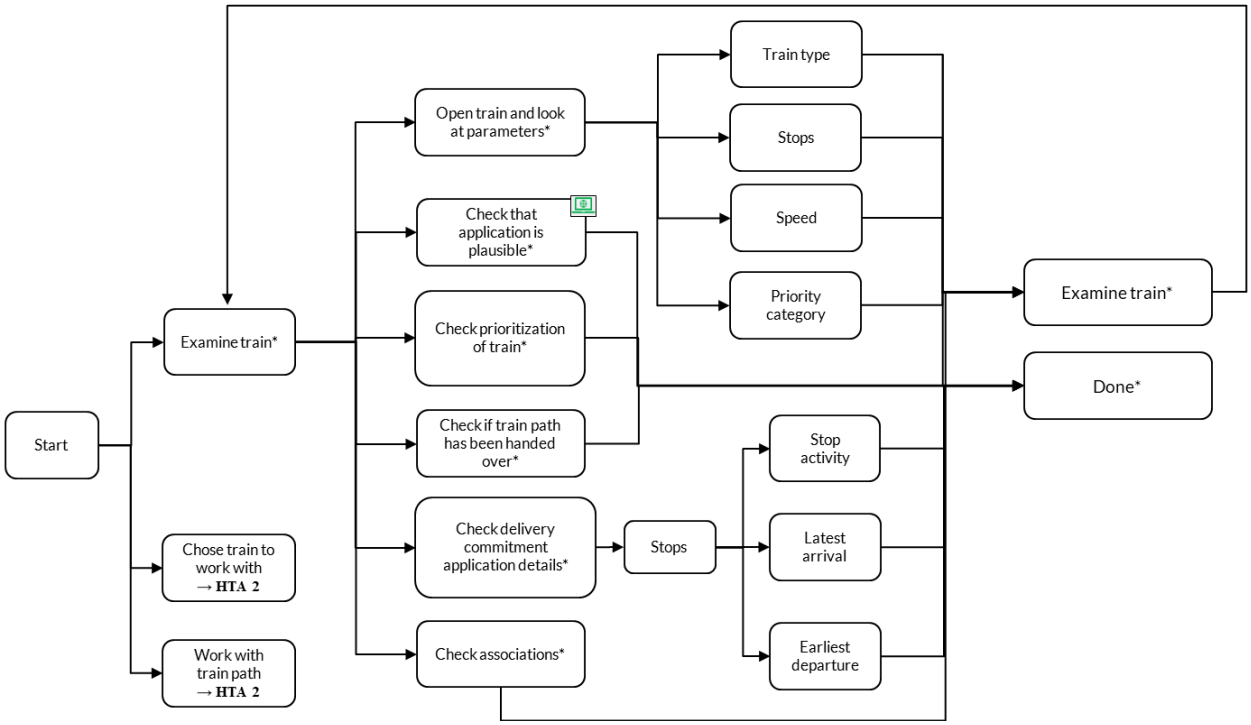
12.3. Appendix C

It provides additional information to Section 6.5.

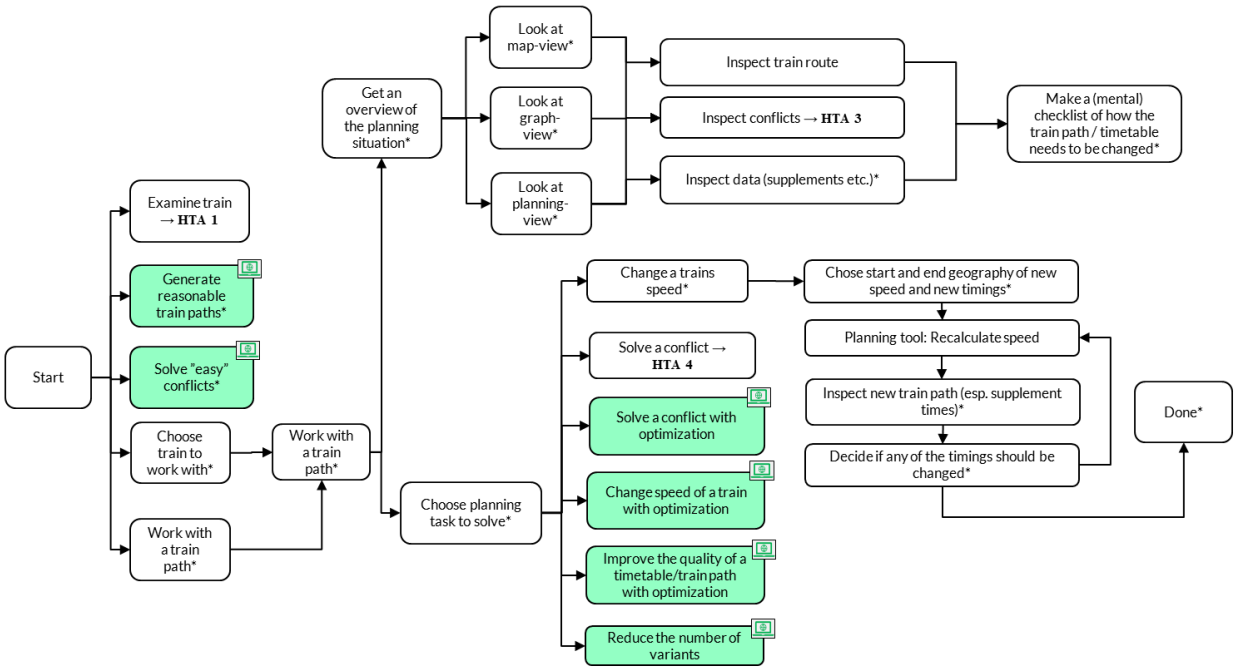
Appendix C

Below are the HTAs from after the second iteration. A green computer symbol indicates that the task can be supported by (optimization) algorithms, and green tasks are (new) tasks that will be performed quite differently if optimization algorithms are used as support. An asterix after a task name means that that task had a slide with functionality suggestions that could serve as a discussion basis during the interview.

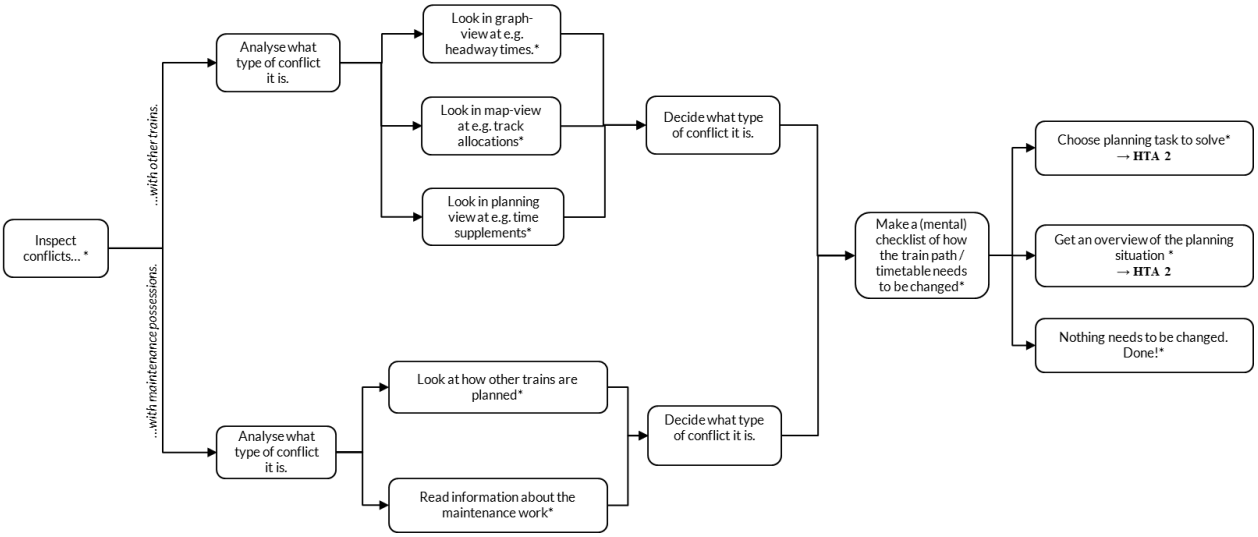
HTA 1 – Examine trains



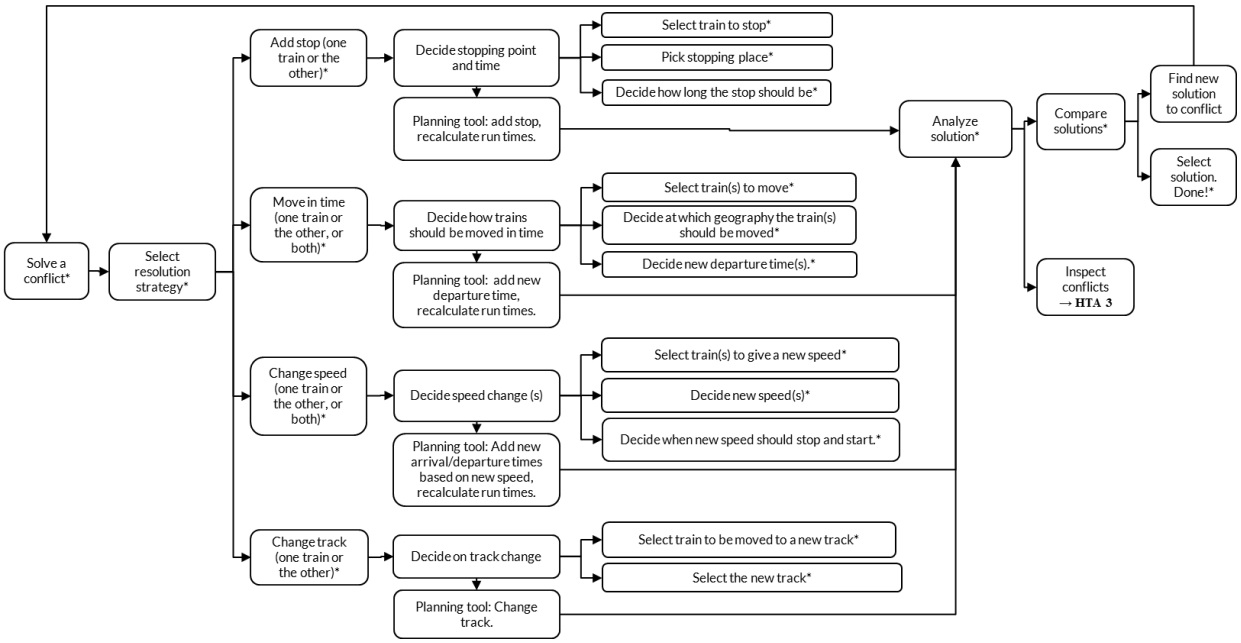
HTA 2 – Chose train to work with



HTA 3 – Inspect conflicts



HTA 4 - Solve a conflict



12.4. Appendix D

It provides additional information to Section 7.2.

Appendix Linköping University

A.1. Model and implementation

In this section we present the mathematical model used in the algorithm. First we describe the initial (free-running) MILP used in Step 1 of iteration 1, then how we check for violations (Step 3) and finally which constraints are added to remove these violations (in Step 5).

A.1.1. Sets, variables and parameters

This section describes the sets, variables and parameters used in the model. Some are used in the MILP presented in the next section, and some needed for the constraints added in later sections.

Sets:.

A	set of trains
R^S	set of station resources
R^T	set of track resources
R	set of all resources, $R = R^S \cup R^T$
R_a	set of resources in the route of train a

Variables:.

t_r^a	≥ 0, time train a enters resource r.
$t_{r_{out}}^a$	≥ 0, time train a leaves the last station R_a
$d_{a,r}$	absolute difference of when train a enters resource r compared to the original timetable
z_r^a	$= \begin{cases} 1, & \text{if train } a \text{ stops at station (resource) } r. \\ 0, & \text{otherwise} \end{cases}$
$z_{a,r}^{both}$	$= \begin{cases} 1, & \text{if train } a \text{ stops at both the preceding and subsequent station of track (resource) } r \\ 0, & \text{otherwise} \end{cases}$

$$p_r^{ab} = \begin{cases} 1, & \text{if train } a \text{ exits resource } r \text{ before train } b \text{ enters it} \\ 0, & \text{otherwise} \end{cases}$$

$$m_r^{ab} = \begin{cases} 1, & \text{if train } a \text{ meets train } b \text{ in resource } r \\ 0, & \text{otherwise} \end{cases}$$

$$e_r^{ab} = \begin{cases} 1, & \text{if train } a \text{ arrives at resource } r \text{ before train } b. \\ 0, & \text{otherwise} \end{cases}$$

Parameters:.

$\bar{t}_r^a$ original arrival time for train a at resource r

δ_r^a How much earlier train a may may be scheduled to resource r

Δ_r^a How much later train a may may be scheduled to resource r

Θ^a maximum travel time of train a

$d_{a,r}^{min}$ minimum time train a needs to pass station (includes required stopping time) r, $r \in R^S \cap R_a$

$d_{a,r}^{passing}$ maximum time train a can use to pass station r, $r \in R^S \cap R_a$ without stopping

$d_{a,r}^{FF}$ minimum time needed to pass track segment r, if not stopping at station before, or after

$d_{a,r}^{FS}$ extra time needed to pass track segment r, if not stopping at station before but do stop at station after track segment r

$d_{a,r}^{SF}$ extra time needed to pass track segment r, if stopping at station before but not at station after track segment r

$d_{a,r}^{SS}$ extra time needed to pass track segment r, if stopping at both station before and after track segment r

M large positive number

α time representing "safety margin". Could be different values for precedence, headway, overtaking and consecutive trains entering a station with $C_r = 1$.

C_r capacity of resource $r \in R$

A.1.2. Base-model

This section presents the base model (MILP) that the algorithm starts with in Step 1.

Formulation:.

$$\min \sum_{a \in A} \sum_{r \in R_a} \frac{d_{a,r}}{|R_a|} \tag{A.1}$$

Subject to:

$$t^a_r - \bar{t}^a_r \leq d_{a,r} \qquad a \in A, r \in R_a \tag{A.2}$$

$$\bar{t}^a_r - t^a_r \leq d_{a,r} \qquad a \in A, r \in R_a \tag{A.3}$$

$$t^a_{r+1} - t^a_r \geq \tag{A.4}$$

$$d^{FF}_{a,r} + d^{FS}_{a,r}\,(z^a_{r+1} - z^{both}_{a,r}) + d^{SF}_{a,r}\,(z^a_{r-1} - z^{both}_{a,r}) + d^{SS}_{a,r}\,z^{both}_{a,r} \quad a \in A, r \in R_a \cap R^T$$

$$z^a_{r-1} + z^a_{r+1} \leq 1 + z^{both}_{a,r} \quad a \in A, r \in R_a \cap R^T \tag{A.5}$$

$$z^a_{r-1} + z^a_{r+1} \geq 2 \cdot z^{both}_{a,r} \quad a \in A, r \in R_a \cap R^T \tag{A.6}$$

$$t^a_{r+1} - t^a_r \leq d^{passing}_{a,r} + M z^a_r \quad a \in A, r \in R_a \cap R^S \tag{A.7}$$

$$t^a_{r+1} - t^a_r \geq d^{min}_{a,r} \quad a \in A, r \in R_a \cap R^S \tag{A.8}$$

$$\bar{t}^a_r - t^a_r \leq \delta^a_r \quad a \in A, r \in R_a \tag{A.9}$$

$$t^a_r - \bar{t}^a_r \leq \Delta^a_r \quad a \in A, r \in R_a \tag{A.10}$$

$$t^a_{r_{out}} - t^a_{r_{in}} \leq \Theta^a \quad a \in A \tag{A.11}$$

$$t^a_r \geq 0 \quad a \in A, r \in R_a \tag{A.12}$$

$$t^a_{r_{out}} \geq 0 \quad a \in A, r \in R_a \tag{A.13}$$

$$d_{a,r} \geq 0 \quad a \in A, r \in R_a \tag{A.14}$$

$$z^a_r \in \{0,1\} \quad a \in A, r \in R_a \cap R^S \tag{A.15}$$

$$z^{both}_{a,r} \in \{0,1\} \quad a \in A, r \in R_a \cap R^T \tag{A.16}$$

The objective function, as described in (A.1) represents the total change of arrival times to all resources for all trains. The changes are normalized with the length of the train path of each train, to ensure that each train is given equal

weight regardless of the length of the train path. However, if all resources are included, the algorithm is likely to produce strange results. Which resources should be included in the objective function is something needed to be considered for each planning situation. One general recommendation is to only include important station-resources, e.g. stations where train a has a fixed stop in the original timetable.

Constraints (A.2) – (A.3) keep track of how much the original arrival times are changed. So, these only update the help-variables used in the objective function. Constraint (A.4) ensures that the minimum running time of train a on resource r is fulfilled. The minimum running time on track-resource r depends on whether the train stops at the station before and/or after the track-resource, or passes at full speed. This means that the four different stop patterns are; Full-Full, Full-Stop, Stop-Full, Stop-Stop. Constraints (A.5) – (A.6) guarantees that if train a stops at both station (before and after track-resource r), the correct minimum running time is used. Constraint (A.7) says that if a train spends more time than a given value in a station area, the train has stopped.

Constraint (A.8) says that train a needs to spend at least a minimum amount of time (including required stopping time) at station r. Constraints (A.9) – (A.10) limit the time for when a train may enter a specific resource, based on the time in the original timetable. Constraint (A.11) limits the total travel time of trains, and constraints (A.12) – (A.16) sets our continuous and binary variables. In general, constraints (A.8) – (A.11) and (A.15) are ways through which the planner can steer the output timetable from the algorithm.

The base-model presented above, only includes ”free-running constraints”, i.e. no interactions between trains are considered so far. This is the base of our model, that the algorithm starts with in Step 1. In Step 3, violations of operational rules are checked and constraints needed to resolve these violations are added. How this is done is described in the following section.

A.1.3. Checking for violations

As mentioned in Section 2.3, we need to find all existing conflicts (violations to rules) and augment the MILP with the constraints needed to prohibit the conflicts. We do this by identifying three types of conflicts (interactions as mentioned in Section 2.2): *Precedence conflicts*, *Headway and overtaking conflicts*, and *Capacity conflicts*. For *Precedence conflicts*, and *Headway and overtaking conflicts*, a conflict is considered between pairs of trains and for *Capacity conflicts* a set of conflicting trains are considered. For more detailed descriptions of why these rules are needed, we refer to Kloster et al. (2023).

A.1.3.1. Precedence Conflicts

With the assumptions given in Section 2.1, precedence conflicts can only occur on resources with capacity 1 (single track sections, and single track stations). Constraints (A.17) – (A.19) are added to resolve this type of conflict. For subsequent trains in the same direction these constraints apply to stations (with capacity 1) and for trains going in opposite directions they apply to both tracks and stations.

If any of these conflicts are identified involving train a and train b on resource r, the binary variables p_r^{ab} and p_r^{ba} are created, and constraints (A.17) – (A.19) are added.

$$p_r^{ab} + p_r^{ba} = 1 \tag{A.17}$$

$$(t_{r+1}^a + \alpha) - t_r^b \leq M(1 - p_r^{ab}) \tag{A.18}$$

$$(t_{r+1}^b + \alpha) - t_r^a \leq M(1 - p_r^{ba}) \tag{A.19}$$

Constraint (A.17) says that one of the trains has to leave the resource before the other enters, and constraints (A.18) – (A.19) includes a safety margin, parameter α. The value of α depends on whether the conflict involves two trains going in the same or opposite directions.

A.1.3.2. Headway and overtaking conflicts

Without any constraints defining the order of trains, or headway between trains, solutions can be given where a train caches up with, or even passes, another train on a track resource. Variables e_r^{ab} are created for all identified violations of that type between train a and train b on resource r. To prohibit conflicts when a train catch up with the train in front, and thereby violates the minimum headway requirement, constraints (A.20) – (A.22) are then added.

$$e_r^{ab} + e_r^{ba} = 1 \tag{A.20}$$

$$(t_r^a + \alpha) - t_r^b \leq M(1 - e_r^{ab}) \tag{A.21}$$

$$(t_r^b + \alpha) - t_r^a \leq M(1 - e_r^{ba}) \tag{A.22}$$

If a violation of the rules occur where a train actually overtakes another on resource r, then also constraints (A.23) – (A.26) are added.

$$e^{ab}_{r+1} + e^{ba}_{r+1} = 1 \tag{A.23}$$
$$(t^a_{r+1} + \alpha) - t^b_{r+1} \leq M(1 - e^{ab}_{r+1}) \tag{A.24}$$
$$(t^b_{r+1} + \alpha) - t^a_{r+1} \leq M(1 - e^{ba}_{r+1}) \tag{A.25}$$
$$e^{ab}_r = e^{ab}_{r+1} \tag{A.26}$$

A.1.3.3. Capacity conflicts

For stations with capacity > 1, we must also check for any capacity violations, i.e. making sure there are never more trains in a station than the capacity allows. When such situations are identified, variables p^{ab}_r, p^{ba}_r and m^{ab}_r are created and constraints (A.27) – (A.31) are added.

$$p^{ab}_r + p^{ba}_r + m^{ab}_r = 1 \tag{A.27}$$
$$(t^a_{r+1} + \alpha) - t^b_r \leq M(1 - p^{ab}_r) \tag{A.28}$$
$$(t^b_{r+1} + \alpha) - t^a_r \leq M(1 - p^{ba}_r) \tag{A.29}$$
$$(t^a_r + \alpha) - t^b_{r+1} \leq M(1 - m^{ab}_r) \tag{A.30}$$
$$(t^b_r + \alpha) - t^a_{r+1} \leq M(1 - m^{ab}_r) \tag{A.31}$$

Constraint (A.27) makes sure that for all trains involved in a capacity conflict the following is true: either train a leaves the station before train b enters it, train b leaves the station before train a enters it, or they meet in the station. An α safety margin is included in constraints (A.28) – (A.31) but is treated as a model-technical margin ε.

Also, for any set of trains, S, involved in a capacity conflict in a station, we create all subsets S' such that $|S'| = C_r + 1$. For each such subset, constraint (A.32) is added. This limits the number of simultaneous meetings that can occur in a station.

$$\sum_{\{a,b\} \in S'} m^{ab}_r \leq \binom{|S'|}{2} - 1 \tag{A.32}$$

B.2. Parameter settings

Parameter	Value (seconds)	Description
δ_r^a	3600	Limit move earlier
Δ_r^a	$3600 * 3$	Limit move later
Θ^a	$2 * (original\_travel\_time)$	Max travel time
$d_{a,r}^{min}$	1	Min time to pass a station
$d_{a,r}^{passing}$	1	Max time at station without stopping
$\alpha_{opposite}$	180	Time separation of trains going in opposite directions
$\alpha_{following}$	60	Time separation of trains going in the same direction (following)
$\alpha_{capacity}$	1	Meeting time separation in capacity constraint

Table B.1: Parameter settings in test case.

References

Kloster, O., Luteberget, B., Mannino, C., Sartor, G., 2023. An optimization-based decision support tool for incremental train timetabling. Operations Research Forum 4.

12.5. Appendix E

It provides additional information to Section 7.3.

APPENDIX Task 6.3: Short-term Timetable Planning at NSR, Optimisation for Maintenance Possessions

Gábor Maróti, Marije Siemann

October 16, 2024

1 Introduction

Preventive maintenance of the railway infrastructure necessitates the closure of some parts of the networks for a few days, forcing adjustments to the annual timetable. In the Netherlands, the scheduling of the maintenance works starts 6 to 12 months before the operations, followed by the actual adjustments of the timetable until 8 weeks before the operations, after which the rolling stock and crew schedules are adjusted. The problem of finding an adjusted timetable arises quite often: almost every weekend has maintenance works at multiple locations. The timetable adjustment process is a cooperation between the infrastructure manager ProRail and the passenger and freight operators.

The Dutch railway timetable is periodic with a period of one hour. The timetable adjustment process considers the basic shape of a day's timetable, assuming that it arises by repeatedly carrying out a one-hour timetable. Therefore, the problem setting of this research features one-hour timetables.

The timetable adjustment process is composed of two main parts. The first part decides which maintenance works should be carried out on which days. These decisions must respect both the availability of the engineering equipment and crews, and the by-train reachability of the major population centres. The first part focuses on capacity planning, without working out the trains' adjusted departure and arrival times.

The second part of the process produces a conflict-free adjusted hourly timetable (AHT) in which the services of the annual timetable may be adjusted by being fully cancelled, partially cancelled, re-routed, and shifted in time. In addition, bus services are ordered if a temporarily out-of-service railway link has no reasonable detour option in the remaining railway network.

The AHT is valid for the duration of a given set of infrastructure maintenance works. This means that most weekends need multiple AHTs since many works finish by Saturday evening and many others start on Sunday morning. Moreover, the maintenance works are scattered throughout the country and tend to affect the flow on multiple corridors of a highly interconnected railway network (see Figure 1). Therefore, it is desirable to consider the entire country's AHT, rather than splitting up the problem geographically.

In this work we focus on supporting the second part of the process. The goal is to design an AHT that facilitates the passenger and freight flows on the available infrastructure. It is preferred to minimise cancellations and to adjust the departure and arrival times as little as possible. The latter goal helps limit the impact on the rolling stock and crew schedules as well as on connections with other modes of public transport.

There is no decision support tool available to help computing the adjusted timetable; it is an entirely manual process. Considering the large number of AHTs, the planning

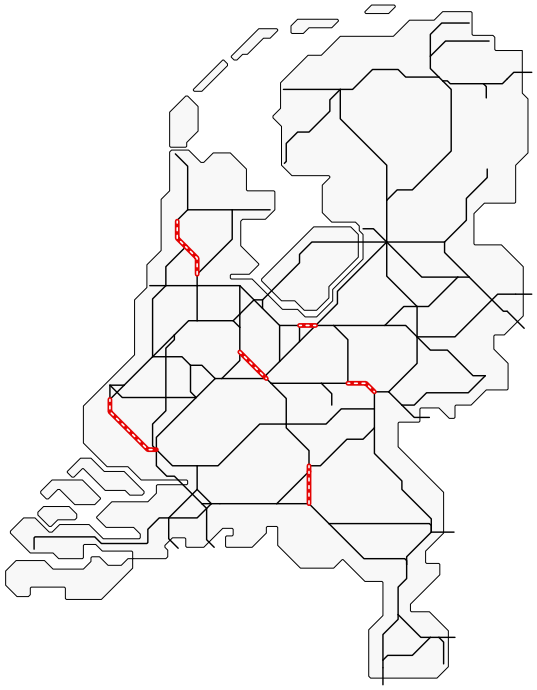


Figure 1: An example for a set of infrastructure segments in the Dutch railway network that are unavailable on a particular day due to preventive maintenance.

department is under constant time pressure which gives room for working out the details of one AHT, without being able to explore alternative ideas.

This research aims at solving problem instances that are relevant for real-life decision making. In the highly interconnected Dutch railway system, this necessitates considering the entire network, including all passenger and freight services. Our approach is designed for the AHT problem on a macroscopic level. We do admit that the solution may not be conflict-free on a microscopic level. We are not aware of any approach that would allow for optimising the entire country's microscopic railway network. This limitation can be mitigated in two ways. First, macroscopy is likely to be sufficient in the early stages of the process where the possible maintenance possessions are scheduled or where a first sketch of an AHT is constructed with sufficient time to resolve the microscopic conflicts. Second, our future research will investigate the prospects of post-processing tools that transform a macroscopically feasible timetable into a microscopically feasible one.

2 Outline of the proposed method

In this research we build upon the mixed integer linear programming model of Van Aken et al. (2017a,b) for the macroscopic AHT problem. Their model is an extension of the Periodic Event Scheduling Problem (PESP) model; it decides on the cancellation of some services and on the departure and arrival times of the remaining services. The objective is to minimise train cancellations as well as the deviation from the annual timetable's departure and arrival times.

We propose the Network Adjusted Hourly Timetable (NAHT, see Section 4) model which is an extension of the approach of Van Aken et al. (2017a,b). We enhance the

model's scope to include decisions on the trains' turning patterns at their terminal stations. Moreover, we propose a solution method that enables us to scale up the AHT calculations for problems instances of practically relevant size and complexity.

In addition, we propose the Macroscopic Track Usage (MTU, see Section 5) model which is inspired by the work of Van Lieshout (2021) on the integration of periodic timetabling with rolling stock scheduling. The MTU model takes the outcome of NAHT, and determines the necessary number of tracks, thereby identifying the stations where the NAHT solution exceeds the available track capacity.

In this work we propose an interactive approach for the AHT problem where we combine the NAHT and MTU models with the insights of field experts, as depicted in Figure 2. Our approach is iterative, and each iteration consists of two stages. In the first stage of an iteration, we solve the NAHT model, i.e., the macroscopic timetabling problem ignoring the stations' limited platform capacity. In the second stage, we solve the MTU model to identify the track bottlenecks of the NAHT solution. Then the field experts suggest modifications of the NAHT model's specification to resolve the capacity bottlenecks. The modifications can include the cancellations of some services, re-arranging the turning patterns and requiring a certain time separation between the departure and arrival events. The modified NAHT problem specifications apply to the next iteration's calculations. The interactive algorithm terminates when all track capacity violations are resolved.

While an integrated optimisation model with network- and station-level aspects is desirable from a theoretical point of view, this research chooses for an iterative approach for pragmatic reasons. Our goal is to find AHT solutions of a good quality in a few minutes, even if the solutions may not be optimal. However, the integrated model is unlikely to be suitable for this goal because NAHT problem instances can be very challenging to solve even without track capacity. The addition of track capacity constraints to NAHT escalates both the memory consumption and the computing times.

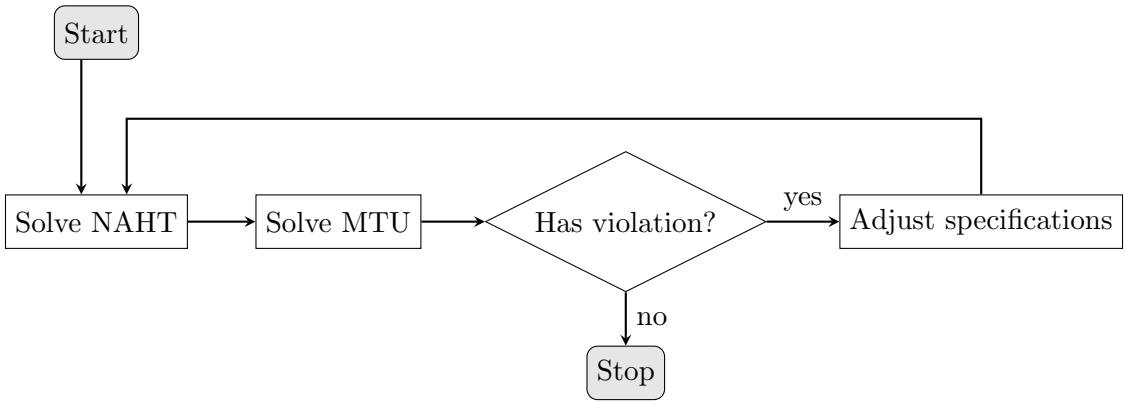


Figure 2: The structure of the proposed iterative framework.

3 Problem formulation

We consider instances of the adjusted hourly timetabling (AHT) problem in the following form. Given is the available infrastructure, consisting of *nodes* and *segments* between the nodes.

In our application, the nodes are the railway stations and the junctions; we consider them as black boxes by disregarding their internal routing limitations. However, we do have information about the number of available platform tracks and non-platform tracks.

If a passenger service calls at a station (including its origin and destination stations), it has to use a platform track. Non-calling passenger services and freight services can use any track.

Each infrastructure segment connects two nodes and has a given number of parallel tracks. We distinguish single-track, double-track, and multi-track segments, i.e., those with at least three tracks.

We are also given the list of passenger and freight services, as well as the *nominal hourly timetable*. The nominal timetable specifies the departure and arrival times of each service at each node it visits. For the sake of uniformity, the passage of a train at a node is modelled by a pair of an arrival and a departure event with a zero-minute time difference. The services are sequences of departure and arrival *events*, connected by alternating *driving* and *dwelling* activities.

In periodic timetabling, the departure and arrival times are minutes of the clock-face. That is, a service may enter a segment at 58 minutes of the hour, and leave it at 2 minutes. In accordance with practice, we use a planning time granularity of 1/10th of a minute, thus each event is selected from the set $\{0.0, 0.1, \ldots, 59.9\}$.

Our problem setting assumes that the maintenance possessions are given as a list of infrastructure limitations: some segments and nodes have fewer available tracks available than in the nominal timetable. We want to emphasise that a segment can be blocked fully or partially.

The AHT problem amounts to modifying the nominal hourly timetable by the following decisions:

- Services can be fully or partially cancelled. For each service, some of its visited nodes are marked as a possible cutting point. The cutting points divide the service into *portions*. Then the portions between any neighbouring pair of cutting points can be cancelled independently from each other. In particular, it is possible to cancel the middle part of a service, and to keep operating the parts on either side of a maintenance possession.

  The non-cancelled portions of a service form *contiguous parts* which can be viewed as a set of new, shorter services. The *starting events* (or *finishing events*) of a service are the first (or last) events of its contiguous parts.

  In our problem setting, freight services have no cutting points: either they run on their full route or they are cancelled.

  In order to avoid unrealistic patterns, railway practice may require that, after cancellations, each service remains operating on at most two contiguous parts.

- The nominal departure and arrival times of the services can be adjusted.

- Services can be re-routed by selecting a route from an explicitly given list of candidate routes. This option is particularly important for freight services that need to be able to reach the harbours and industrial sites.

- Whenever a passenger service has a finishing event at a node, this event will be connected to the starting event of another service at the same node by a *turning* activity. A nominal service can give rise to several turnings in the AHT problem, attached to each of its contiguous parts.

The adjusted timetable needs to fulfil various requirements. The first group of requirements concerns individual trains. The travel times and dwell times must lie between given lower and upper bounds that arise for technical and commercial reasons.

The second group of constraints concerns the interaction between the services on the limited railway infrastructure. For double-track segments, we assume that one track is used for each traffic direction. Our model includes *headway* restrictions to ensure a safe time separation between services that travel in the same direction on double-track segments. In addition, another set of headway constraints expresses that services are separated if they use a single-track segment in any direction.

Multi-track segments offer a considerable traffic capacity and routing flexibility, however, the precise modelling requires excessive complications in the macroscopic view of our application, both for data processing and for the mathematical formulations. Therefore, this proof-of-concept research omits the headway constraints between services on a multi-track segment.

Our models connect at each infrastructure node the services' finishing events to the starting events by choosing the turnings by a perfect matching. The selected turnings must be sufficiently long so that the drivers have enough time to walk from one end of a train to the other end. A maximum turning time can also be set. The turnings themselves are constrained by the allowed rolling stock types of the services (such as high speed, Inter-City or regional rolling stock).

Our models decide the starting and finishing events by cancelling some portions of the service. The matching structure of the turnings results in solutions with an equal number of starting and finishing services at each node. This allows for a steady flow of rolling stock and crews when the one-hour timetable is carried out repeatedly.

We are given a lower and an upper bound for each event time in order to limit the deviation between the nominal and adjusted event times.

The solution must not exceed the nodes' track capacity, both for the platform tracks and for the total number of tracks. Track capacity is consumed by dwelling services, by passing services as well as by turnings.

In our application, freight services are merely reserved paths that are going to be filled in a few days or weeks before the operations. In contrast, the adjusted hourly timetable is computed several months earlier.

The nominal timetable contains many freight services that share the time slots on a large part of their routes before diverging to different destinations; obviously, at most one of these paths will be used in each concrete hour. Therefore, we need special treatment for freight trains. In our models, we omit all headway constraints between freight services, however, we do keep such constraints between passenger services as well as between a passenger service and a freight service. Intuitively, our models ensure that the passenger services are macroscopically conflict-free, and that each freight service can run in the time slots between the passenger services. In addition, freight services are not included in turnings because they do not form a strictly repeating hourly pattern.

The optimisation objective is to minimise penalties for train cancellations and for shifting the nominal event times. In essence, we penalise the deviation from the nominal timetable. According to our objective function, the nominal timetable is the unique optimal solution of the AHT problem if there is no maintenance possession.

4 Network Adjusted Hourly Timetable (NAHT) model

In this section we describe the Network Adjusted Hourly Timetable (NAHT) model which is an extension of the work of Van Aken et al. (2017a,b).

Throughout this paper we consider period time T, and we assume that all event times belong to the interval $[0, T_{\max}]$ where $0 < T_{\max} < T$. The gap $T - T_{\max}$ is the time granularity of the underlying application; the Dutch timetables have a granularity of 1/10th of a minute. Although the models below admit any fractional event time, real-life timetabling will need an appropriate rounding to discrete time instants, namely to integer multiples of the granularity.

The NAHT model is defined with an event-activity network (E, A) where E is the set of events, and $A \subseteq E \times E$ is the set of activities. The events in set E consist of the departure and arrival events of the services. The activities arise from the restrictions on the travel times, on the dwell times ans on the turning times, as well as from the headway constraints. The key modelling idea is that all travel time, dwell, turning and headway restrictions can be expressed by imposing a lower and an upper bound on the (modulo T) time difference of two events.

4.1 The generic PESP model

A PESP instance is given by an event-activity network (E, A). The PESP model asks for a time instant for each event in such a way that the (modulo T) time difference of each activity is between the lower and upper bounds.

Formally, PESP amounts to finding a function $\nu : E \to [0, T_{\max}]$ such that the modulo T time difference of each activity lies within a given periodic interval. That is,

$$\nu_{e'} - \nu_e \mod T \in [\ell_{ee'}, u_{ee'}]_T \qquad \text{for each } ee' \in A. \tag{1}$$

The difference is taken modulo T due to the periodic time axis. We note that $\ell_{ee'} > u_{ee'}$ is possible, e.g., it is the case when 0 is an interior point of the periodic interval $[\ell_{ee'}, u_{ee'}]_T$.

Constraints (1) are commonly reformulated by introducing integer-valued auxiliary variables $p_{ee'} \in \{0, 1, 2\}$ and by requiring

$$\ell_{ee'} \le \nu_{e'} - \nu_e + T \cdot p_{ee'} \le u_{ee'} \qquad \text{for } ee' \in A \text{ with } \ell_{ee'} \le u_{ee'}, \tag{2}$$

$$\ell_{ee'} \le \nu_{e'} - \nu_e + T \cdot p_{ee'} \le T + u_{ee'} \qquad \text{for } ee' \in A \text{ with } \ell_{ee'} > u_{ee'}. \tag{3}$$

4.2 The PESP layer of the NAHT model

Our NAHT model is an extension of the Periodic Event Scheduling Problem (PESP) formulation. The most crucial difference to the classic PESP formulation is the possibility of cancelling services.

We introduce the following decision variables:

- Cancellation variables $y_{m,k} \in \{0, 1\}$ where m is a service and $k \in \{0, 1, \ldots, \#(m)-1\}$ is a non-negative integer index. The indices refer to the portions between the service's cutting points. The symbol $\#(m)$ denotes the number of portions of service m. The choice $y_{m,k} = 1$ indicates cancellation.
- Event time variables $\nu_e \in [0, T_{\max}]$.
- Auxiliary variable $\beta_e \in \{0, 1\}$ for each event e, used for the lower and upper bounds on the event times.
- Auxiliary variable $p_{ee'} \in \{0, 1, 2\}$ for each activity ee', used for the lower and upper bounds on the duration of the activities.

The PESP constraints, and their relation to the cancellation variables, are implemented by the following constraints. In what follows, the symbol $e \sim (m,k)$ means that event e belongs to portion k of service m, that is, that the cancellation variable $y_{m,k}$ directly affects event e.

Lower and upper bounds on the event times

Each event e is supplied with a lower bound ℓ_e and an upper bound u_e on its scheduled event time. Then we require

$$\nu_e + \ell_e \cdot y_{m,k} \geq \ell_e \qquad \text{for } e,m,k \text{ with } \ell_e \leq u_e \text{ and } e \sim (m,k), \quad (4)$$

$$\nu_e + u_e \cdot y_{m,k} \leq u_e \qquad \text{for } e,m,k \text{ with } \ell_e \leq u_e \text{ and } e \sim (m,k), \quad (5)$$

$$\nu_e + \ell_e \cdot y_{m,k} + T \cdot \beta_e \geq \ell_e \qquad \text{for } e,m,k \text{ with } \ell_e > u_e \text{ and } e \sim (m,k), \quad (6)$$

$$\nu_e + (u_e + T) \cdot y_{m,k} + T \cdot \beta_e \leq u_e + T \qquad \text{for } e,m,k \text{ with } \ell_e > u_e \text{ and } e \sim (m,k). \quad (7)$$

These constraints express that either event e is cancelled (in which case ν_e is set to zero), or ν_e belongs to the periodic interval $[\ell_e, u_e]_T$.

The PESP constraints themselves

For any activity ee' with lower bound $\ell_{ee'}$ and upper bound $u_{ee'}$, we define the constant

$$u^*_{ee'} = \begin{cases} u_{ee'} & \text{if } \ell_{ee'} \leq u_{ee'}, \\ u_{ee'} + T & \text{if } \ell_{ee'} > u_{ee'}. \end{cases} \quad (8)$$

Then we require for each ee', m, k, m', k' with $e \sim (m,k)$ and $e' \sim (m',k')$ the constraints

$$\nu_{e'} - \nu_e + T \cdot p_{ee'} + \ell_{ee'} \cdot (y_{m,k} + y_{m',k'}) \geq \ell_{ee'}, \quad (9)$$

$$\nu_{e'} - \nu_e + T \cdot p_{ee'} + (u^*_{ee'} - T_{\max}) \cdot (y_{m,k} + y_{m'k'}) \leq u^*_{ee'}. \quad (10)$$

Constraints (9) - (10) express that if none of the events e and e' is cancelled then their time difference belongs to the periodic interval $[\ell_{ee'}, u_{ee'}]_T$.

We note that all driving and dwelling activities ee' have $\ell_{ee'} \leq u_{ee'}$. In addition, all driving activities and many of the dwelling activities have the property that their events e and e' are related to the same cancellation variable $y_{m,k}$. If this is the case, constraints (9) and (10) can be replaced for these activities by the slightly tighter constraints

$$\nu_{e'} - \nu_e + T \cdot p_{ee'} + \ell_{ee'} \cdot y_{m,k} \geq \ell_{ee'}, \quad (11)$$

$$\nu_{e'} - \nu_e + T \cdot p_{ee'} + u_{ee'} \cdot y_{m,k} \leq u_{ee'}. \quad (12)$$

4.3 Turning of the services

At each infrastructure node, the NAHT model connects the finishing events of the services to the starting events. The potential turnings are pairs ee' of events. Here e is an arrival event at a cutting point of its service; this event may become a finishing event by cancellations. Similarly, e' is a departure event at the same node which is a cutting point for its service.

We introduce the following decision variables.

- Binary variable $k_{ee'} \in \{0,1\}$ to express whether a potential turning ee' is selected.
- Auxiliary variable $p_{ee'} \in \{0,1,2\}$ for each potential turning ee'.

The constraints below control the time duration of the selected turnings, and they link the turning variables to the cancellation variables.

Time duration of a turning

Using the notation (8), the following constraints ensure that the time duration of each selected turning lies between the specified minimal and maximal turning times.

$$\nu_{e'} - \nu_e + T \cdot p_{ee'} \geq \ell_{ee'} \cdot k_{ee'} \tag{13}$$

$$\nu_{e'} - \nu_e + T \cdot p_{ee'} \leq u^*_{ee'} + T_{\max} \cdot (1 - k_{ee'}) \tag{14}$$

Matching constraints

Consider a service m, and let e be the arrival event at the end of kth portion. Let

$$y^*_e = \begin{cases} y_{m,k+1} & \text{if } k < \#(m) - 1, \\ 1 & \text{if } k = \#(m) - 1. \end{cases} \tag{15}$$

Then we require the following constraints.

$$\sum_{e'} k_{ee'} \geq y^*_e - y_{m,k}, \tag{16}$$

$$y_{m,k} + \sum_{e'} k_{ee'} \leq 1, \tag{17}$$

$$\sum_{e'} k_{ee'} \leq y_{m,k+1} \qquad \text{if } k < \#(m) - 1. \tag{18}$$

Constraints (16) ensure that at least one turning will leave every finishing event. Constraints (17) and (18) express that at most on turning will leave e, but no turning at all if e is not a finishing event after the cancellations.

An analogous set of constraints handles in-coming turnings of the potential starting events. Altogether, the selected turnings form a perfect matching between the starting and finishing events.

4.4 Identifying and counting the contiguous parts

After partial cancellations in the middle of its geographical path, a service splits into multiple contiguous parts. Our objective function in Section 4.7 will discourage the fragmentation of the services by incurring a penalty term for each contiguous part after the first one.

For counting the contiguous parts of service m, we observe that whenever a departure event e of m becomes a starting event in a NAHT solution, event e has exactly one incoming turning activity $e'e$ for which the turning variable $k_{e'e}$ is 1. Therefore, letting $D(m)$ denote the set of departure events of service m, the number of contiguous parts of m is equal to

$$\sum_{e \in D(m)} \sum_{e'e} k_{e'e}. \tag{19}$$

This sum is zero if service m is fully cancelled.

The number of contiguous parts after the first one is counted by the non-negative integer-valued auxiliary variable cp_m and by the constraint

$$cp_m \geq \sum_{e \in D(m)} k_{e'e} - 1. \tag{20}$$

More precisely, cp_m is an upper bound on the number of contiguous parts. However, this relation is sufficient as cp_m is going to be minimised.

4.5 Route choices

Suppose that service m of the nominal timetable has alternative routes $m'_1, \ldots, m'_t$; the nominal service itself is one of these alternatives. The alternative routes differ from each other in that they may take different geographical paths, or they may call at a different set of stations. Each alternative is naturally modelled as a service. Then the NAHT model will have to replace service m by at most one of the alternative services $m'_1, \ldots, m'_t$, and to fully cancel all other alternatives. We note that, once an alternative service m'_j is chosen, partial cancellation can still apply to its portions.

Routing decisions are implemented in the NAHT model by adding the alternative routes to the list of services, and by requiring the same sets of constraints (such as for travel time, dwell time, headway and turning) as we did for all nominal services. In addition, we need to express the logic of selecting at most one alternative route. In order to do so, we introduce the binary decision variables

$$\begin{aligned} &\widetilde{x}_m \in \{0,1\}, \\ &x_{m'_j} \in \{0,1\} \qquad\qquad \forall j \in \{1,\ldots,t\}. \end{aligned}$$

Variable $\widetilde{x}_m$ has the value of 1 if none of the alternative routes of service m runs at all. Variables $x_{m'_j}$ indicate which alternative route is chosen.

The following constraints link the new decision variables to each other and to the partial cancellation variables $y_{m'_j,k}$ of the alternative routes, defined for $j \in \{0,1\}$ for each $j \in \{1,\ldots,t\}, k \in \{0,\ldots,\#(m'_j)-1\}$.

$$\widetilde{x}_m + \sum_{j=1}^{t} x_{m'_j} = 1 \tag{21}$$

$$x_{m'_j} + y_{m'_j,k} \geq 1 \qquad \forall j \in \{1,\ldots,t\}, k \in \{0,\ldots,\#(m'_j)-1\} \tag{22}$$

$$\sum_{k=0}^{\#(m'_j)-1} y_{m'_j,k} \leq \#(m'_j) - x_{m'_j} \qquad \forall j \in \{1,\ldots,t\} \tag{23}$$

Constraints (21) ensure that at most one routing alternative is chosen. By constraints (22), all portions of a not chosen alternative are cancelled. Constraints (23) ensure that not all portions of the chosen alternative are cancelled.

The cancellation variables $y_{m'_j,k}$ cannot see whether the partial cancellation takes place in a chosen or a non-chosen alternative route. This ambiguity causes problems for assessing the cancellation costs in the objective function. Therefore, we introduce the binary *punishable cancellation variables* $\widetilde{y}_{m'_j,k} \in \{0,1\}$ for each $j \in \{1,\ldots,t\}, k \in \{0,\ldots,\#(m'_j)-1\}$. The variables $\widetilde{y}$ indicate whether a portion of a chosen alternative route is cancelled. The model needs the following additional constraints.

$$\widetilde{y}_{m'_j,k} \leq x_{m'_k} \qquad \forall j \in \{1,\ldots,t\}, k \in \{0,\ldots,\#(m'_j)-1\} \tag{24}$$

$$\widetilde{y}_{m'_j,k} \leq y_{m'_j,k} + 1 - x_{m'_k} \qquad \forall j \in \{1,\ldots,t\}, k \in \{0,\ldots,\#(m'_j)-1\} \tag{25}$$

$$y_{m'_j,k} \leq \widetilde{y}_{m'_j,k} + 1 - x_{m'_k} \qquad \forall j \in \{1,\ldots,t\}, k \in \{0,\ldots,\#(m'_j)-1\} \tag{26}$$

By constraints (24), non-chosen alternative routes have zero-valued punishable cancellation variables. By constraints (25) - (26), chosen alternative routes will satisfy $y_{m'_j,k} = \widetilde{y}_{m'_j,k}$.

We note that re-routing is particularly important for freight services. Our problem setting does not admit a partial cancellation of freight services. Then the presented modelling of alternative routes can be simplified and sharpened slightly.

4.6 Measuring the event shifts

The deviation of the adjusted event times from the nominal event times needs to be measured, as well. We introduce the following decision variables.

- Non-negative continuous variables d_e^+ to measure the positive deviation of ν_e from the nominal event time π_e.
- Non-negative continuous variables d_e^- to measure the negative deviation of ν_e from the nominal event time π_e.
- Auxiliary binary variables α_e.

Then we require the following constraints.

$$\nu_e + d_e^- + T \cdot \alpha_e + \pi_e \cdot y_{m,k} = \pi_e + d_e^+ \qquad \text{for each } e, m, k \text{ with } e \sim (m,k). \tag{27}$$

By these constraints, the difference $d_e^+ - d_e^-$ is equal to the difference between the nominal and adjusted event times (or to zero if the event is cancelled). Optimal solutions will have the property that at most one of the deviation variables is non-zero as they will appear in the objective function with positive coefficients. The term $T \cdot \alpha_e$ accounts for the modulo T calculations of the time.

We note that the lower and upper bounds on ν_e need to be sufficiently tight so that positive and negative deviations can be distinguished. Indeed, a positive shift of 40 minutes in the hourly timetable is equivalent to a negative shift of 20 minutes.

4.7 The objective function

The objective function minimises the sum of the penalties for undesirable features such as the cancellation of a service, and a shift of an event.

In addition, discussions with practitioners revealed that they prefer solutions where partial cancellations create few contiguous parts. We incur a penalty for each contiguous part after the first one. That is, no penalty is given to the contiguous parts for full cancellation and for the case where partial cancellations leave one contiguous part of the service intact; one unit of penalty is incurred if an underway cancellation creates two contiguous parts; and so on. Moreover, we incur a penalty proportionally to the geographic distance of the cancelled portion.

The time shift penalties are proportional to the magnitude of the shift. We consider different penalties for forward (positive) and for backward (negative) shifts.

Cancellation penalty for services

Let m be a service without alternative routes. Let $H_{m,k}$ denote the length of the kth portion of m, and let w_m be a non-negative weight for service m. Then the objective function includes the expression

$$w_m \left(f \cdot cp_m + g \cdot \sum_k H_{m,k} \cdot y_{m,k} \right). \tag{28}$$

The first term is the penalty for creating new contiguous parts. The second penalises the total length of the cancelled portions. The penalty terms include the weights f and g.

A service m with alternative routes is handled similarly; we just need to consider its alternative routes m'_j as well the corresponding punishable cancellation variables $\widetilde{y}_{m'_j,k}$. The objective function will include

$$w_m \left(f \cdot \sum_j cp_{m'_j} + g \cdot \sum_j \sum_k H_{m'_j,k} \cdot \widetilde{y}_{m'_j,k} + h_m \cdot \widetilde{x}_m \right). \tag{29}$$

The term $h_m \cdot \widetilde{x}_m$ accounts for the full cancellation of the service.

Event shift penalty

Let g_m denote the weight of service m, and let b^+ and b^- denote the penalties for shifting an event by 1 minute in positive and negative direction, resp. Then the services' total event shift penalty is

$$\sum_m \sum_e g_m \left(b^+ d_e^+ + b^- d_e^- \right). \tag{30}$$

5 Macroscopic Track Usage (MTU) model

The NAHT model considers the railway nodes as black boxes. In this section, we describe the Macroscopic Track Usage (MTU) model that takes the outcome of NAHT and determines whether the stations' track capacity is sufficient.

A macroscopic hourly timetable violates the station's capacity if its dwelling and turnings activities consume too many platform tracks or through-going tracks at a node. We distinguish platform tracks and non-platform tracks. If a passenger service calls at the station, it must use a platform track. Non-calling passenger services and freight services can use any track.

Platform track capacity often turns out to be a problem at railway nodes just outside the maintenance possession that become turning locations due to cancellations. These nodes can be small stations with few platform tracks; if one chooses too long turning times, the platform tracks cannot accommodate all services.

Based on the cancellations and time shifts of the NAHT solution, MTU computes how many platform tracks and non-platform tracks are minimally needed to operate the timetable. More precisely, we compute the necessary number of tracks *on top* of the already available tracks. Lacking any interaction between the stations, the track capacity problem decomposes into independent sub-problems, namely one at each infrastructure node. The track usage model is inspired by the techniques proposed by Van Lieshout (2021).

As mentioned in Section 3, our approach treats passenger services and freight services differently. The MTU model determines the turnings between passenger services. Together with the dwellings of the services, these turnings form the track occupation intervals of the passenger services. Then the MTU model will determine which occupation interval is followed immediately by which other occupation interval, implicitly defining the track allocation. Moreover, the model will ensure that the passage of each freight fits in between at least one consecutive pair of occupation intervals.

5.1 Problem setting, notations and graph representation

We describe the MTU model for a single infrastructure node. Let V_{arr} and V_{dep} be the set of arrival and departure events, resp., of the passenger services. Let $V = V_{\text{arr}} \cup V_{\text{dep}}$.

We define the following sets of arcs.

- A *dwelling* arc from the arrival event to the departure event of a service. A zero-minute dwelling arc is used to represent services that pass through the node without calling.
- A *turning* arc from each finishing arrival event of a service to each starting departure event. These arcs represent the possible turnings. Notice that each event is incident either with exactly one dwelling arc, or with an arbitrary number of turning arcs. Depending on the application, one may restrict the set of turning arcs, e.g., by requiring that Inter-City services can turn onto each other, but not onto regional services, due to the different rolling stock types used in different categories of services.
- A *follow-up* arc from each departure event to each arrival event. A follow-up arc indicates the possibility that the arrival event is the first service to use the track after the departure event's service. The set of follow-up arcs can be restricted to elements that are likely to be feasible in practice, e.g., to those with a sufficiently large time separation.

Let A denote the set of all dwelling, turning and follow-up arcs. Let A_{platform} be the set of dwelling and turning arcs that require track occupation at a platform track; this is the case when at least one of its departure and arrival events represent the call of a passenger service. Figure 3 shows an example of the graph.

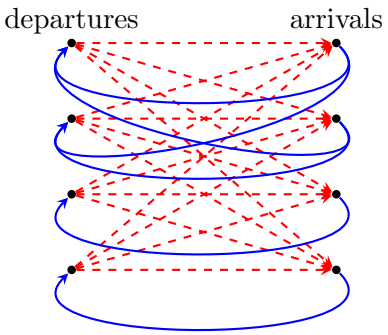


Figure 3: The directed graph behind the MTU model. The dwelling and turning arcs are represented by solid blue arrows; the follow-up arcs by red dashed arrows.

Let $\ell(a)$ denote the modulo T time duration of any arc a. Notice that the length is a well-defined constant since NAHT had already determined all departure and arrival times. Let Q denote the passages of the freight services, and let $\Gamma(q)$ be the set of follow-up arcs whose time gap between its departure and arrival event can accommodate the freight service of q. The definition of $\Gamma(q)$ ensures the sufficient time separation between the freight service and the passenger services.

5.2 Main modelling idea

The macroscopic track usage of passenger services is modelled by selecting the turning arcs as well as the succession of track occupations (i.e., the follow-up arcs). The selected turning arcs and the dwelling arcs form a perfect matching, and the selected follow-up arcs form a perfect matching, as well.

The arc set of the union of these matchings forms a network flow that decomposes into disjoint directed cycles. The sum of the lengths in each directed cycle is an integer multiple of T. Dividing the sum by T yields the number of tracks needed to accommodate the track occupations in the cycle. If a cycle needs more than one track then the services are placed on the tracks in a cyclic fashion as the hourly timetable is executed repeatedly. For example, if a directed cycle uses 3 tracks and a service in it is placed on track 1 in the first hour, then it will be placed on track 2 in the second hour, on track 3 in the third hour, and on track 1 again in the fourth hour.

The passage of each freight service should fit into at least one of the idle intervals (i.e., a follow-up arc) between the passenger services. Moreover, our model also needs to allow a track dedicated to freight services, since this possibility is not visible through the arcs of the passenger services. It is worth noting that one single dedicated freight track can accommodate *all* freight services according to the assumed lack of timetabling conflicts between freight trains.

5.3 Mathematical formulation

The mathematical formulation of the MTU model contains the following decision variables.

- $\phi^{\mathrm{P}} \in \{0,1\}^A$ and $\phi^{\mathrm{NP}} \in \{0,1\}^A$ indicate the selected dwelling, turning and follow-up arcs for platform tracks and for non-platform tracks, resp.
- γ^{P} and γ^{NP} are binary indicators for using a platform track and a non-platform track dedicated to freight services, resp.
- The non-negative integer variables ω^{P} and ω^{NP} count the number of used platform tracks and non-platform tracks, resp.
- The non-negative integer slack variables σ^{P} and σ^{NP} count how many tracks are used on top of the available tracks, resp.

Letting s^{P} and s^{NP} denote the number of available platform tracks and non-platform tracks, resp., the MTU model reads as follows.

$$\min \ \sigma^{\mathrm{P}} + \sigma^{\mathrm{NP}} \tag{31}$$

$$\text{s.t} \sum_{a \in \delta^{\mathrm{out}}(v)} \phi^{\mathrm{P}}(a) + \sum_{a \in \delta^{\mathrm{out}}(v)} \phi^{\mathrm{NP}}(a) = 1 \qquad \text{for each } v \in V \tag{32}$$

$$\sum_{a \in \delta^{\mathrm{out}}(v)} \phi^{\mathrm{P}}(a) = \sum_{a \in \delta^{\mathrm{in}}(v)} \phi^{\mathrm{P}}(a) \qquad \text{for each } v \in V \tag{33}$$

$$\sum_{a \in \delta^{\mathrm{out}}(v)} \phi^{\mathrm{NP}}(a) = \sum_{a \in \delta^{\mathrm{in}}(v)} \phi^{\mathrm{NP}}(a) \qquad \text{for each } v \in V \tag{34}$$

$$\phi^{\mathrm{NP}}(a) = 0 \qquad \text{for each } a \in A_{\mathrm{platform}} \tag{35}$$

$$\gamma^{\mathrm{P}} + \gamma^{\mathrm{NP}} + \sum_{a \in \Gamma(q)} \big(\phi^{\mathrm{P}}(a) + \phi^{\mathrm{NP}}(a)\big) \geq 1 \qquad \text{for each } q \in Q \tag{36}$$

$$\omega^{\mathrm{P}} = \gamma^{\mathrm{P}} + \frac{1}{60} \sum_{a} \ell(a) \phi^{\mathrm{P}}(a) \tag{37}$$

$$\omega^{\mathrm{NP}} = \gamma^{\mathrm{NP}} + \frac{1}{60} \sum_{a} \ell(a) \phi^{\mathrm{NP}}(a) \tag{38}$$

$$\sigma^{\mathrm{P}} \geq \omega^{\mathrm{P}} - s^{\mathrm{P}} \tag{39}$$

$$\sigma^{\mathrm{NP}} \geq \omega^{\mathrm{NP}} - s^{\mathrm{NP}} \tag{40}$$

$$\phi^{\mathrm{P}} \in \{0,1\}^A, \phi^{\mathrm{NP}} \in \{0,1\}^A \tag{41}$$

$$\gamma^{\mathrm{P}} \in \{0,1\}, \gamma^{\mathrm{NP}} \in \{0,1\} \tag{42}$$

$$\omega^{\mathrm{P}} \geq 0, \omega^{\mathrm{NP}} \geq 0 \tag{43}$$

$$\sigma^{\mathrm{P}} \geq 0, \sigma^{\mathrm{NP}} \geq 0 \tag{44}$$

Constraints (32), (33) and (34) describe binary 2-commodity network flows with flows ϕ^{P} and ϕ^{NP} with arc capacities of 1. By constraint (35), calls of passenger services take place at platform tracks.

Constraints (36) ensure that either there is a dedicated freight track or there is a suitable follow-up arc for each freight service.

Constraints (37) and (38) compute the number of used platform tracks and non-platform tracks, resp. Constraints (39) and (40) determine the values of the slack variables σ^{P} and σ^{NP}.

The objective function minimises the sum of the slack variables. The available tracks are sufficient to accommodate all services (under our simplified assumptions about feasibility) if and only of the optimal objective value is zero.

One may wonder about the possibilities of combining the MTU model with NAHT in a single mathematical model. We are not aware of any easy way of integrating the models due to three challenges, all of them stemming from the fact that the event times (and thereby the arc lengths $\ell(a)$) are constant in MTU but variable in an integrated model. First, the admissibility of a follow-up arc depends on the arc lengths in that too short successions of track occupations are not allowed. Second, constraints (37) and (38) include the products $\ell(a)\phi^{\mathrm{P}}(a)$ and $\ell(a)\phi^{\mathrm{NP}}(a)$ which become quadratic terms in the integrated model. Third, the sets $\Gamma(q)$ of suitable follow-up arcs for freight services are defined based on the planned event times.

The integration of the network-level NAHT model and the station-level MTU model is a subject of our future research.

6 Iterative framework with the NAHT and MTU models

We design an interactive and iterative framework to find good solutions to the adjusted hourly timetabling problem. In each iteration, the network-level NAHT model is solved without station capacity limitations. Its outcome serves as input to the station-level MTU model to identify track capacity violations. A human decision maker analyses the violations, if any, and suggests modifications to the NAHT model specifications of the next iteration. The high-level overview of the approach is shown in Figure 2.

In our experiments we use the following feedback actions to adjust the NAHT problem's specification.

- *Forced cancellation* of a portion of a service. For example, the NAHT model's objective prefers to let the service run on all segments that do not undergo maintenance. One may decide that the portion right next to a maintenance possession (i.e., a nearly hit portion) could be cancelled to relieve a track capacity violation without any substantial negative effect on the passenger flows and the rolling stock schedules.

- Adjusted *turning patterns*. The NAHT specification is rather conservative about turning possibilities. In-bound services are normally limited to turn onto serviced on the same geographic corridor. Relaxing this inflexibility can lead to shorter turning times, thereby resolving a track capacity violation.

- Adjusting the *departure and arrival times* so that a sufficient time separation is created. Simultaneous track occupations, e.g., the passage of two service at the same time, require two tracks. By delaying one of the services slightly, the services pass the station at the same track.

Forced cancellation and adjusting turning pattern are added to the NAHT model by setting the values of the corresponding binary decision variables to 0 or to 1. The departure and arrival time changes are achieved by adding a PESP constraint to the model, thereby ensuring the desirable time separation.

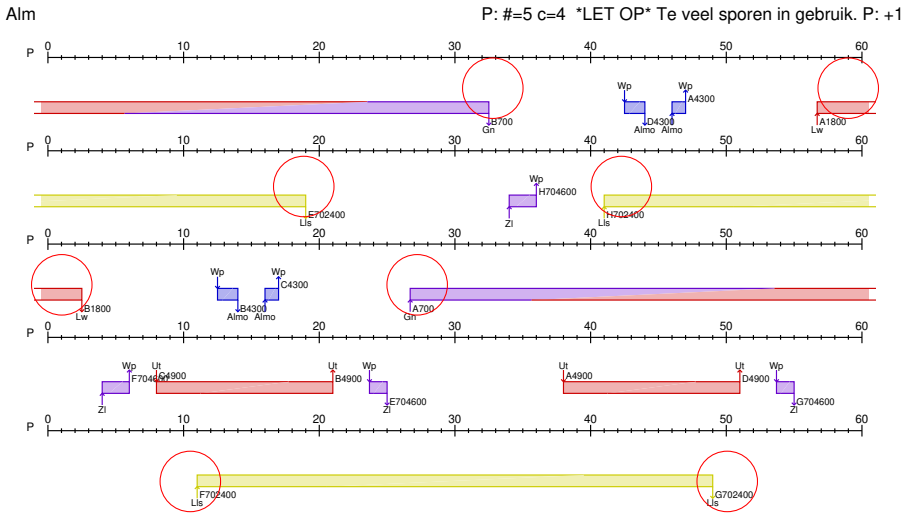
A NAHT solution may have capacity violations at multiple stations. In such a case, the human decision makers can choose to address all violated stations in one single feedback action; alternatively, they can focus on the most aching station in the feedback, leaving the other violations for subsequent iterations. Both strategies have their advantages and disadvantages. Focusing on a single station in the feedback is a conceptually easier task. However, it may commit to decisions that turn out to cause additional violations at other stations.

An example of such feedback considerations is shown in Figure 4. Each coloured rectangle indicates a track occupation, starting with an arrival event and ending with a departure event. The track occupations of station 'Alm' in the first iteration has some turnings that are longer than 30 minutes (see Figure 4(a)). These turnings may be there because they are the only allowed ones in NAHT problem specification, or the NAHT solver may have chosen them from many possible turnings. For whichever reason the long turnings exist, their effect is clear: the solution requires 5 tracks, however, the station has only 4 tracks. The human decision maker can suggest using alternative turnings at this station that better fit to the departure and arrival times. The adjusted turnings involve the circled events. Figure 4(b) is the track occupation in the second iteration. The shorter turnings save two tracks.

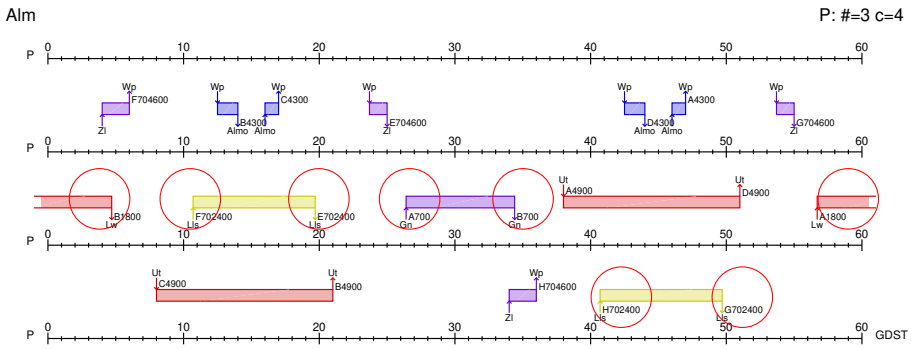
7 Hot starts for the NAHT model

The NAHT model is a big-M extension of the computationally already quite challenging PESP formulation. Particularly difficult are the instances with partially blocked infrastructure segments. If a segment is fully blocked, all originally scheduled services are cancelled on this segment; commercial mixed integer linear programming (MILP) solvers (such as CPLEX or Gurobi) can easily find rather sharp lower bounds and (near-)optimal feasible solutions. In contrast, partially blocked segments leave non-trivial decisions about which services are to be cancelled. Then the MILP solvers often struggle to find good feasible solutions and tight bounds.

A common speed-up technique for MILP solvers is to compute feasible solutions, and to provide these solutions as hot starts to the MILP solver. If the hot start solutions are of sufficiently good quality, they cut off large parts of the branch-and-bound tree, and they can help the solver's internal search algorithms to find better solutions. All in all, hot starts often have a dramatic improvement on the solution times of large and complex optimisation problems.



(a) The track occupation of the first iteration's NAHT solution, using 5 tracks, thus violating the station capacity of 4.



(b) The track occupation in the second iteration with updated turning possibilities, using only 3 tracks.

Figure 4: Track occupation at station 'Alm' in the first and in the second iteration.

We generate hot start solutions by imposing heuristic restrictions on the NAHT model's solution space. We are motivated by the intuition that the maintenance works are unlikely to lead to cancellations or time shifts in services that operate in other geographic parts of the country.

To compute a hot start solution, we identify which services are deemed to be affected by the maintenance possessions. Then we solve the NAHT model subject to the additional restriction that the unaffected services keep their nominal timetable without any cancellation or time shift. Thanks to the smaller solution space, we expect the MILP solver to find good solutions quickly for the restricted hot start models.

Our proof-of-concept implementation contains several ways to compute hot starts; they differ in their definition for a service to be affected, by the maintenance possessions. We use the following concepts (see Figure 5). A service is *directly affected* if it passes in the nominal timetable through an infrastructure segment under maintenance. A service is *indirectly affected* if it is not directly affected, but it shares a station with a directly affected service. For a directly affected service, a portion is said to be *directly hit* if is contains a segment under maintenance. A portion of a directly affected service is *nearly hit* if it is not directly hit but it is neighbouring a directly hit portion.

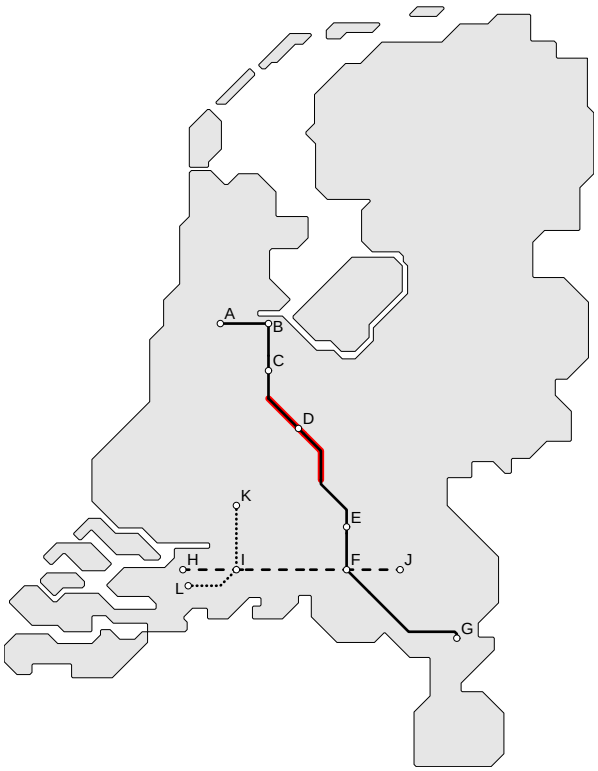


Figure 5: An example of affected services and portions in the map of the Netherlands. The maintenance takes place in the marked area around station D. Service 1 (solid line between stations A and G) is *directly affected*; its portions CD and DE are *directly hit*, its portions BC and EF are *nearly hit*. Service 2 (dashed line between stations H and J) is *indirectly affected*. Service 3 (dotted line between stations K and L) is not affected at all.

We use the following variants of hot starts.

CompleteClosure: All services are cancelled on all infrastructure segments under maintenance. That is, partial blockage is replaced by complete blockage.

AreaAndTime: For directly affected services, the departure and arrival times may be shifted by at most 5 minutes. Moreover, cancellation is forbidden on portions that are not directly hit. All other services must keep their nominal timetable.

LargerAreaAndTime: For directly affected services, the departure and arrival times may be shifted by at most 5 minutes. Moreover, cancellation is forbidden on portions that are not directly or nearly hit. For indirectly affected services, the departure and arrival times may be shifted by at most 5 minutes, but cancellation is forbidden. All other services must keep their nominal timetable.

LineAndTime: For directly affected services, the departure and arrival times may be shifted by at most 5 minutes. For indirectly affected services, the departure and arrival times may be shifted by at most 5 minutes, but cancellation is forbidden. All other services must keep their nominal timetable.

Time: For affected services, the departure and arrival times may be shifted by at most 5 minutes.

In our numerical tests, we compare two methods for solving NAHT problem instances.

- The *Direct Method* (DM) amounts to solving the NAHT model by a commercial MILP solver without hot starts.
- The *Hot Start Method* (HSM) amounts to solving the hot start calculation models with 'CompleteClosure', 'AreaAndTime', 'LargerAreaAndTime', 'LineAndTime' and 'Time' (in this particular order), followed by solving the unrestricted NAHT model. When solving this sequence of six integer programs, each of them taking the solutions of all preceding models as hot starts.

We note that the solution space of the hot start models grows in general as we progress in the HSM sequence. Indeed, HSM starts with the strongest heuristic restrictions, and proceeds towards less and less strong ones, finishing with no restrictions at all. Therefore, each hot start calculation benefits from the preceding hot starts, as well.

We want to emphasise that both DM and HSM are exact solution methods. The heuristic considerations arise only in the process of generating some feasible solutions before the MILP solver's branch-and-bound procedure starts.

8 Computational tests

We test the proposed solution approach on problem instances that are derived from the real-life 2024 timetable of the Netherlands. The test data consist of 292 passenger services of NSR and 108 freight services, altogether with over 9500 departure and arrival events on a network of 293 stations and 207 junctions. These services form the nominal timetable.

We consider 9 test instances that arise from the nominal timetable by selecting one or more segments for a maintenance possession (see Table 1). The selected segments have a reduced number of available tracks. Fully blocked segments have no available track, otherwise the segment is partially blocked.

The maintenance possession scenarios are synthetic, they do not correspond to real-life cases. We selected the possessions in such a way that non-trivial computational or conceptual challenges appear in solving the NAHT nodes, in resolving the capacity violations, or

both. The possessions have a varying impact on the nominal timetable. The number of directly affected services ranges from 16 to as high as 97. The coefficients in the objective function are given in Table 2.

The practical value of the proposed algorithms is partly determined by the computation times. An efficient workflow necessitates calculations to be performed in a few minutes. Therefore, we limited the computation time of the NAHT model. Both the Direct Method (DM) and the Hot Start Method (HSM) have a time limit of 15 minutes in every iteration. DM can spend up to 15 minutes on solving the single NAHT model, while HSM has 2 minutes for each hot start model and 5 minutes for the sixth (unrestricted) NAHT model. We note that the MTU model poses little computational challenge: it is solvable to optimality for all stations in a total of about 20 seconds.

Our test implementation was written in the Java programming language, using the commercial mixed integer linear programming solver CPLEX 22.0.1, and run on a standard desktop computer.

Table 1: Characteristics of the 9 test instances.

A	• One segment (Mrn-Klp) reduced from 2 tracks to 0. • Directly affects 16 services.
B	• One segment (Rtz-Rlb) reduced from 4 tracks to 0. • Directly affects 37 services.
C	• One segment (Gdpa-Wp) reduced from 4 tracks to 0. • Directly affects 41 services.
D	• One segment (Asb-Ac) reduced from 4 tracks to 0. • Directly affects 46 services.
E	• One segment (Asdz-Shl) reduced from 4 tracks to 0. • Directly affects 52 services.
F	• One segment (Har-Klp) reduced from 2 tracks to 1. • Directly affects 12 services.
G	• One segment (Ah-Nm) reduced from 2 tracks to 1 track. • Directly affects 22 services.
H	• Three segments with reduced track capacity: Gvc-Ledn from 4 to 2, Bkl-Asb from 4 to 2, and Ed-Ah from 2 to 1. • Directly affects 66 services.
I	• Four segments with reduced track capacity: Ah-Nm from 2 to 1, Ehv-Btl from 4 to 2, Ut-Utvr from 8 to 6, and Utvr-Htn from 4 to 2. • Directly affects 97 services.

8.1 Tractability of the NAHT model

In our first tests we wonder to what extent the network-level NAHT model is solvable for large-scale instances. We apply both DM and HSM on test instances ‘A’ to ‘I’. In these experiments, we do not attempt to resolve any track capacity violation.

The numerical results are presented in Table 3. DM solves instances ‘A’ to ‘E’ to optimality; we note that the actual solution time is between 10 and 20 seconds. This observation is in line with the intuition that possessions with fully blocked segments lead to

Table 2: Penalty coefficients in the objective function of the NAHT model.

Cancellation of a freight service	1000
New starting event of a passenger service	100
Cancellation of a passenger service (per kilometre)	10
Positive time shift for passenger services (per event, per minute)	1
Negative time shift for passenger services (per event, per minute)	2
Positive time shift for freight services (per event, per minute)	0.01
Negative time shift for freight services (per event, per minute)	0.01

easy optimisation problems. Indeed, the NAHT model must cancel all portions of a service that has an overlap with the maintenance possessions. Then the theoretically best solution would let each remaining portion of a service run with its nominal departure and arrival times. The only constraints that can be violated by this naive idea are the restrictions on the minimal or maximal turning times at the boundary of the maintenance possessions. Modern MILP software is able to cope with these violations efficiently. The resulting optimal solutions have very few time shifts or cancellations on top of the mandatory cancellation, indeed.

Applying HSM does not have any benefit: HSM and DM find the same optimal solution, although HSM is slower due to the burden of solving five more optimisation problems.

In contrast, instances 'F' to 'I' feature partially blocked segments, and they behave very differently. The commercial MILP solver fails to prove optimality for any of these instances. For test instance 'F', both DM and HSM find attractive solutions. With merely 12 affected services, instance 'F' is the easiest one of partially blocked instances.

The benefits of HSM are highlighted by instances 'G', 'H' and 'I'. HSM finds good solutions within the allotted 15 minutes of computation time. The solutions by DM do not have any practical value at all; more than two thirds of the services are cancelled.

Instances with partially blocked segments (i.e., instance 'F' to 'I') have weak lower bounds. This behaviour is very common in optimisation models with layers of big-M constraints such as the NAHT model. The bounds from DM are somewhat higher then those from HSM, simply because HSM spends merely 5 minutes on the unrestricted NAHT model, while DM has 15 minutes for it. The additional time is sufficient to derive sharper bounds.

We do claim, though, that the HSM solutions of 'F' to 'I' are of good quality from a practical point of view. Indeed, an informal assessment revealed that the amount of cancellation is in line with the traffic that the available infrastructure is likely to be capable of facilitating.

The amount of time shifts is also acceptable. Instance 'H' has by far the highest amount of shift with its 1290 minutes. Out of the 9564 departure and arrival events in total, 8577 events are not shifted, and the remaining 987 events have an average shift of 1.31 minutes, ranging between 0 and 6 minutes. Mind that event shifts are counted not only at stations where the services call, but also at each station and junction they pass.

For the harder test instances 'F' to 'I', the solution quality of DM does not benefit from increasing the time limit from 15 minutes to 4 hours. Table 4 shows the objective values and the proven lower bounds for DM with 15 minutes, for DM with 4 hours and for HSM. For test instance 'F', the solution found by HSM is proven to be optimal (in 6200 seconds). In the other cases, the DM solutions are still worse (in some cases much worse) than the HSM solutions.

Table 3: Comparison of the solutions obtained from DM and HSM. The first row of each instance indicates the total objective value as well as the contribution of passenger service cancellations, freight service cancellations and time shifts. The subsequent rows contain the proven lower bound on the objective value, the number of (fully or partially) cancelled services, the total geographic length of the cancelled passenger services, and the total amount of shift in minutes.

Instance	DM				HSM			
	Total	Pass.	Freight	Shifts	Total	Pass.	Freight	Shifts
A	3860	3860	0	0	3860	3860	0	0
	$\geq$ 3860	#=16	#=0	0 min	$\geq$ 3860	#=16	#=0	0 min
		266 km				266 km		
B	7940	2940	5000	0	7940	2940	5000	0
	$\geq$ 7940	#=32	#=5	0 min	$\geq$ 7940	#=32	#=5	0 min
		174 km				174 km		
C	12648	7648	5000	0	12648	7648	5000	0
	$\geq$ 12648	#=36	#=5	0 min	$\geq$ 12648	#=36	#=5	0 min
		485 km				485 km		
D	21360	7360	14000	0	21360	7760	14000	0
	$\geq$ 21360	#=32	#=14	0 min	$\geq$ 21360	#=32	#=14	0 min
		416 km				416 km		
E	6660	6660	0	0	6660	6660	0	0
	$\geq$ 6660	#=52	#=0	0 min	$\geq$ 6660	#=52	#=0	0 min
		506 km				506 km		
F	756	514	0	242	744	514	0	230
	$\geq$ 334	#=2	#=0	242 min	$\geq$ 214	#=2	#=0	163 min
		31 km				31 km		
G	181408	118756	61000	1652	2900	2900	0	0
	$\geq$ 753	#=204	#=61	1652 min	$\geq$ 553	#=20	#=0	0 min
		11796 km				250 km		
H	184669	122244	62000	425	5805	4725	0	1080
	$\geq$ 1171	#=204	#=62	425 min	$\geq$ 968	#=24	#=0	1280 min
		12144 km				272 km		
I	183394	119252	62000	2142	4459	4056	0	403
	$\geq$ 911	#=204	#=62	2142 min	$\geq$ 847	#=28	#=0	943 min
		11845 km				325 km		

Table 4: Comparison of the solutions of instances 'F' to 'I' obtained from HSM as well as from DM with a time limit of 15 minutes and 4 hours.

Instance	DM (15 min)		DM (4h)		HSM	
	Objective	Bound	Objective	Bound	Objective	Bound
F	756	$\geq$ 334	744	optimal	744	$\geq$ 214
G	181408	$\geq$ 753	5432	$\geq$ 851	2900	$\geq$ 553
H	184669	$\geq$ 1171	32250	$\geq$ 1201	5805	$\geq$ 968
I	183394	$\geq$ 911	*9541	$\geq$ *941	4459	$\geq$ 847

* Out of memory after 5100 seconds.

8.2 Resolving track capacity violations

In the second group of our tests, we investigate the performance of the iterative solution approach that combines the NAHT and MTU models with each other as well as with the human experts' insights. In particular, we are interested to see how many iterations are necessary to eliminate all track capacity violations. Moreover, we wonder at which costs (in terms of cancellation or time shift) the track capacity violations of the initial NAHT solution can be repaired.

Based on the observations in Section 8.1, we use DM in each iteration of instances 'A' to 'E' (i.e., the ones with fully blocked segments), and we use HSM in each iteration of instances 'F' to 'I' (i.e., the ones with partially blocked segments).

In these tests, the interactive feedback step was performed by the developers of the methods, after extensively discussing the problem setting with field experts. We use the feedback strategy to resolve one station's capacity violations in each iteration.

We summarise the results in Table 5. It turns out that a handful of iterations are sufficient to resolve the capacity violations. Instance 'G' does not require any feedback action, since the first iteration's NAHT solution has no capacity violation. The capacity violations in instances 'B', 'D' and 'I' are resolved in one single iteration. Instances 'A', 'C' and 'H' need multiple iterations because the initial solution has violations at multiple stations. Instance 'E' and 'F' are examples where the feedback action shifts the violations from one station to another a few times in a row; however, all violations disappear after no more than 5 iterations.

Table 5 also compares the objective value and some crucial characteristics of the solutions. The resolution of the capacity violation is achieved at a moderately higher objective value than that in the first iteration, and very little additional cancellation is used. As for time shifts, even the sharp increase in instance 'E' is acceptable in practice: 278 events (out of 9564) are shifted with an average of 3.9 minutes, and with shifts as high as 13 minutes.

Instance 'I' has a slightly lower objective value in the last iteration than in the first one, indicating the first iteration's solution is suboptimal. In instance 'B', the last iteration's solution decides to apply partial cancellations in 28 services (as opposed to 32 services in the first iteration). However, the combined length of the cancellations is higher, resulting in a slightly more expensive solution.

9 Conclusions and future work

In this research we consider the problem of adjusting a nominal timetable due to preventive maintenance possessions. Our goal is to solve large and complex networks such as the Dutch railway system. Next to the macroscopic network view, we want to account for the limited track capacity of the stations.

We propose an optimisation model for solving the network-level timetabling problem as well as a station-level macroscopic track usage optimisation model that in essence evaluates the outcome of the network-level model by counting the number of tracks. The network-level and station-level models are coupled by a feedback step in which human decision makers provide their expert judgement.

We carry out computational experiments in a test environment. Our tests use the real-life Dutch timetable and infrastructure data. We consider all passenger services of NSR as well as all freight services.

Table 5: Comparison between the first iteration's solution (containing track capacity violation) and last iteration's solution (containing no track capacity violation). The first row of each instance indicates the total objective value as well as the contribution of passenger service cancellations, freight service cancellations and time shifts. Also, the number of feedback actions is shown. The subsequent rows contain the number of (fully or partially) cancelled services, the total geographic length of the cancelled passenger services, and the total amount of shift in minutes.

Instance	First iteration				Last iteration				#Feedback
	Total	Pass.	Freight	Shifts	Total	Pass.	Freight	Shifts	
A	3860	3860 #=16 266 km	0 #=0	0 0 min	4303	4208 #=16 301 km	0 #=0	95 94 min	3
B	7940	2940 #=32 174 km	5000 #=5	0 0 min	9040	4040 #=28 324 km	5000 #=5	0 0 min	1
C	12648	7648 #=36 485 km	5000 #=5	0 0 min	13235	8112 #=36 651 km	5000 #=5	123 114 min	3
D	21360	7360 #=32 416 km	14000 #=14	0 0 min	21418	7360 #=32 416 km	14000 #=14	58 58 min	1
E	6660	6660 #=52 506 km	0 #=0	0 0 min	8852	7676 #=52 608 km	0 #=0	1176 1098 min	5
F	744	514 #=2 31 km	0 #=0	230 163 min	1097	1028 #=4 63 km	0 #=0	69 68 min	2
G	2900	2900 #=20 250 km	0 #=0	0 0 min	–	– – –	– –	– –	0
H	5805	4725 #=24 272 km	0 #=0	1080 1290 min	6607	5836 #=24 383 km	0 #=0	771 630 min	3
I	4459	4056 #=28 325 km	0 #=0	403 943 min	4436	3652 #=24 285 km	0 #=0	784 1094	1

The proposed iterative approach turns out to be able to solve the considered network-level timetable optimisation problem combined with macroscopic station capacity limitations. Our approach consistently provides solutions with a good practical value. Both the solution time of the individual optimisation steps (limited at 15 minutes) and the small number of iterations are in the range that is desirable in the timetable design workflow.

The proposed approach is a good candidate to become the core of a decision support system. Our further work aims at helping this transition by inviting practitioners for an evaluation of our models on real-life problem instances, in order to refine models and to assess the true value of our methods in practice. These investigations are planned to be a part of Work Package 7.

The network-level timetable optimisation model is challenging to solve when the maintenance possessions block some segments partially. We are working on alternative mathematical formulations and solution methods to reduce the solution times and to find tighter optimality gaps.

The current feedback step from the station-level model to the network-level model is based on expert judgement. By including the experts in the solution process, we expect to enhance to practical value of the solutions. For example, the experts can choose services whose cancellations would cause the least harm to the passenger and freight flows. While the experts' insights are likely to be valuable in future extensions of our algorithm, we are looking for ways to automate the feedback, and to integrate some or all ideas of the station-level model into the network-level optimisation model.

We acknowledge that the practical feasibility of a timetable depends on mesoscopic or microscopic aspects such as the routes of the services inside the stations. We are going to investigate the link from macroscopic timetable optimisation to microscopic simulation, so that the simulation outcome can help the optimisation models obtain practically feasible solutions.

References

S. Van Aken, N. Bešinović, and R. M. Goverde. Designing alternative railway timetables under infrastructure maintenance possessions. *Transportation Research Part B: Methodological*, 98:224–238, 2017a.

S. Van Aken, N. Bešinović, and R. M. Goverde. Solving large-scale train timetable adjustment problems under infrastructure maintenance possessions. *Journal of Rail Transport Planning & Management*, 7(3):141–156, 2017b.

R. N. Van Lieshout. Integrated periodic timetabling and vehicle circulation scheduling. *Transportation Science*, 55(3):768–790, 2021.

12.6. Appendix F

It provides additional information to Section 7.4.

Appendix SNCF EMSE

OSRD links:

Official website: https://osrd.fr/en/

OSRD documentation : https://osrd.fr/en/docs/

STDCM documentation : https://osrd.fr/en/docs/reference/design-docs/stdcm/

GitHub repository: https://github.com/OpenRailAssociation/osrd

12.7. Appendix G

It provides additional information to Section 8.2.

A Mathematical description of the yard selection model

The mathematical description of the yard selection model can be read here.

A.1 Notation and assumptions

The following notational conventions are used throughout the description of the model:

- The set of train units is denoted by U.
- The set of yards is denoted by Y.
- The set of gateways is denoted by G.
- The set of arriving trains is denoted by A.
- The set of departing trains is denoted by D.
- A_u denotes the arrival of unit u.
- D_u denotes the departure of unit u.
- The yard to which a gateway $g \in G$ belongs is denoted by Y_g.

A.2 Decision variables

The model contains the following decision variables.

1. $G_a \in [0, \text{number of gateways} - 1]$ the index of the chosen gateway for arrival a
2. $G_d \in [0, \text{number of gateways} - 1]$ the index of the chosen gateway for departure d
3. $Y_a \in [0, \text{number of yards} - 1]$ the index of the chosen yard for arrival a
4. $Y_d \in [0, \text{number of yards} - 1]$ the index of the chosen yard for departure d

A.3 Constraints

The following constraints enforce that the yard of each arrival and departure are set based on the chosen gateway:

$$Y_a = Y_{G_a} \quad \forall a \in A$$
$$Y_d = Y_{G_d} \quad \forall d \in D$$

A.3.1 Reachability

Of course, each gateway should be reachable, which is expressed in the following hard constraint:

$$\text{there is a route to } G_a\text{, possibly blocked by through-traffic} \quad \forall a \in A,$$

$$\text{there is route to } G_d\text{, possibly blocked by through-traffic} \quad \forall d \in D.$$

A.3.2 Service

The arrival and departure yard must be chosen such that each service is possible on at least one of the yards. Let S_u be the set of required services for unit u:

$$\text{OR}(s \text{ is possible on yard } Y_{A_u}, s \text{ is possible on yard } Y_{D_u}) \quad \forall u \in U, s \in S_u,$$

where OR denotes the built-in or-constraint of many CP solvers.

A.3.3 Capacity

As transfers between yards are calculated as a post-processing step, the capacity of the yards is only taken into account on the yards from which the trains departs. This is enforced by the following constraint:

$$\sum_{u \in U} [\text{arrivalTime}(A_u) \leq ti < \text{departureTime}(D_u)] \cdot l_u \cdot (y = Y_{D_u}) \\ \leq l_y \cdot c \quad \forall y \in Y, \forall ti \in \text{timepoints}(y),$$

where l_u is the length of train unit u, l_y the total stabling track length at yard y and c a fixed *cutting-loss factor*. This factor is used to take into account losses when trains are parked on the individual tracks. In most cases, it is not possible to completely use the capacity of a yard. The *timepoints* function returns all unique timepoints for yard y at which the capacity should be constrained such that at no point in time the total length of trains in the yard exceed its total capacity. This is handled using the built-in cumulative constraint of the CP solver.

A.4 Objective

The problem is formulated as a minimization problem, where the objective consists of three weighted components:

1. the total number of reversals in the routes of the chosen gateways. This is included in order to minimize unnecessary movements,
2. the number of transfers between yards,
3. a penalty each time an arriving / departing shunt train cannot reach the gateway without crossing with through-traffic. Since the through-traffic is fixed, this possiblity can be determined in advance for each arriving / departing shunt train and gateway.

B Mathematical description of the stabling algorithm

The mathematical description of the stabling algorithm model can be read here.

B.1 Notation and assumptions

The following notational conventions are used throughout the description of the model:

- The set of arriving train units is denoted by U and the set of departuring train units by U'. Assumed is that $|U| = |U'|$.
- The set of arriving train compositions is denoted by C and the set of departure train compositions is denoted by C'.

- Functions or variables defined on the domain of arriving units and trains never carry an $'$. If the domain consists of departing trains or units, the same function or variable is used as for the arriving case but with an $'$.
- Each track has two sides, labelled by A and B, and the labelling can be chosen such that one always moves from the A-side of one track to the B-side of a directly adjacent track, and never from A to A or B to B. Thus far, this has shown to be a valid assumption.
- The train units in a train composition are always listed from A to B.
- Side A is identified with -1 and side B with $+1$.
- It is assumed that any split of a train composition happens as soon as the train arrives at its arrival stabling track, and any combine happens just before the combined train composition departs from its departure stabling track. Consequently, the train composition of a relocation is always known, because a relocation always happens after the split and before the combine, if any.
- It is assumed that train units are never blocked by train units parked on other tracks.

B.2 Variables and data structures

B.2.1 Train unit decision variables

Each train unit is parked at at most two stabling tracks. The number of lock-ins on a track can be determined if it is known when each unit has to depart (determined by the matching) and when each unit arrives at the track, departs from the track and from which side. This results in the following integral decision variables on the arrival units:

$$
\begin{aligned}
Na_u &\in [0, M], \quad u \in U \quad \text{(order of arrival)},\\
Ndr_u &\in [0, M], \quad u \in U \quad \text{(order of departure for relocation)},\\
Nar_u &\in [0, M], \quad u \in U \quad \text{(order of arrival after relocation)},\\
Nd_u &\in [0, M], \quad u \in U \quad \text{(order of departure)},\\
Sa_u &\in \{-1, 1\}, \quad u \in U \quad \text{(arrival side)},\\
Sdr_u &\in \{-1, 1\}, \quad u \in U \quad \text{(departure side for relocation)},\\
Sar_u &\in \{-1, 1\}, \quad u \in U \quad \text{(arrival after relocation)},\\
Sd_u &\in \{-1, 1\}, \quad u \in U \quad \text{(departure side)},\\
ta_u &\in [0, \text{number of tracks} - 1], \quad u \in U \quad \text{(arrival track)},\\
td_u &\in [0, \text{number of tracks} - 1], \quad u \in U \quad \text{(departure track)},\\
M_u &\in [0, |U| - 1], \quad u \in U \quad \text{(the matched departure unit)},
\end{aligned}
$$

where $M = 2 \cdot (|U| + \text{maximum amount of relocations}) - 1$. The amount of relocations is always bounded from above in order to reduce the domains of the order variables.

B.2.2 Auxiliary unit variables

The following departure unit variables are used in some of the expressions in the model. Our experience led us to believe that the use of these variables, instead of the corresponding variables

of the matched arrival unit, shows siginificant performance improvements.

$$\begin{aligned}
Nd'_{u'} &\in [0, M], \quad u' \in U' \text{(order of departure')}, \\
Sd'_{u'} &\in \{0, 1\}, \quad u' \in U', \quad \text{(departure side')}, \\
td'_{u'} &\in [0, \text{number of tracks} - 1], \quad u' \in U' \quad \text{(departure track')}, \\
\text{attachedAToBd}(u_1, u_2) &\in \{\text{True}, \text{False}\}, \quad u_1, u_2 \in U \quad \text{(attached at departure)}.
\end{aligned}$$

Here, attachedAToBd is *True* if and only if the B-side of unit u_1 is attached to the A-side of unit u_2 in an arrival compositions. This is a variable in the model because its value depends on the matching variables.

B.3 Train auxliary variables

Although formulated above at the level of units, the arrival and departure sides and tracks of units can actually be defined on the level of arrival and departure train compositions. The following auxiliary variables are introduced on the level of train compositions and will help to formulate some of the constraints efficiently:

$$\begin{aligned}
ta_{tr} &\in [0, \text{number of tracks} - 1], \quad tr \in C \quad \text{(arrival track train)}, \\
td'_{tr'} &\in [0, \text{number of tracks} - 1], \quad tr' \in C' \quad \text{(departure track train')}, \\
Sa_{tr} &\in \{-1, 1\}, \quad tr \in C \quad \text{(arrival side train)}, \\
Sd'_{tr'} &\in \{-1, 1\}, \quad tr \in C' \quad \text{(departure side train')}.
\end{aligned}$$

B.3.1 Unit information

Some data structures are introduced to keep track of some important information of each arriving and departing unit.

$$\begin{aligned}
Ta_u &\in \mathbb{N}, \quad u \in U \quad \text{(arrival time)}, \\
Td'_{u'} &\in \mathbb{N}, \quad u' \in U' \quad \text{(departure time')}, \\
Ca_u &\in [0, |C|], \quad u \in U \quad \text{(arrival composition)}, \\
Cd'_{u'} &\in [0, |C'|], \quad u' \in U' \quad \text{(departure composition')},
\end{aligned}$$

where Ca_u and $Cd'_{u'}$ are the arrival composition containing u and departure composition containing u', respectively.

The following boolean matrices store which train units are attached to each other in the train compositions:

$$\begin{aligned}
&\text{attachedAToBa}(u_1, u_2) \in \{\text{True}, \text{False}\}, \quad u_1, U_2 \in U \quad \text{(attached at arrival)}, \\
&\text{attachedAToBd'}(u'_1, u'_2) \in \{\text{True}, \text{False}\}, \quad u'_1, u'_2 \in U \quad \text{(attached at departure')}.
\end{aligned}$$

Here, attachedAToBa is *True* if and only if the B-side of unit u_1 is attached to the A-side of unit u_2 in one of the arrival compositions. The definition for attachedAToBd' is similar. The latter is defined on departure trains, since the value for the arrival trains is not known in advance as it depends on the matching.

The reachability of tracks from the access points can also be determined beforehand:

$$\begin{aligned}
Ra_{tr} &= \{(t, S) | tr \text{ can reach track } t \text{ via side } S\}, \quad tr \in C \quad \text{(reachable arrival)}, \\
Rd'_{tr'} &= \{(t, S) | tr' \text{ can reach track } t \text{ via side } S\}, \quad tr' \in C' \quad \text{(reachable departure')},
\end{aligned}$$

and for relocations

$$Rr = \{(td, Sd, ta, Sa) | td \neq ta \text{ and track } td \text{ can be entered from side } Sd \text{ by leaving track } ta \text{ from side } Sa\} \quad \text{(allowed relocations)}.$$

B.4 Constraints

B.4.1 Consistency constraints for auxiliary variables

The following constraints have been imposed to guarantee consistency between the decision variables and the auxiliary variables on the train units. These auxiliary variables on the departure unit have the same meaning as the original decision variables defined on the arrival units. Therefore, their values need to be the same if the arriving and departing units are matched:

$$\begin{aligned}
u' = Mu &\implies Nd'_{u'} = Nd_u, \quad u \in U, u' \in U', \\
u' = Mu &\implies Sd'_{u'} = Sd_u, \quad u \in U, u' \in U', \\
u' = Mu &\implies td'_{u'} = td_u, \quad u \in U, u' \in U', \\
(u'_1 = Mu_1) \wedge (u'_2 = Mu_2) &\implies \text{attachedAToBd}(u_1 u_2) = \text{attachedAToBd}'(u'_1, u'_2), \\
&\quad u_1, u_2 \in U, u'_1, u'_2 \in U'.
\end{aligned}$$

Also, the values of the auxiliary train variables need to be the same as those of the variables belonging to the units that are contained in the train:

$$\begin{aligned}
ta_{tr} &= ta_u, \quad u \in U, Ca_u = tr, \\
td'_{tr'} &= td'_{u'} \quad u' \in U', Cd'_{u'} = tr' \\
Sa_{tr} &= Sa_u, \quad u \in U, Ca_u = tr, \\
Sd'_{tr'} &= Sd_{u'} \quad u' \in U', Cd'_{u'} = tr'.
\end{aligned}$$

B.4.2 Matching constraint

The following global constraint enforces that an arriving unit is matched to a unique departing units.

$$\text{allDifferent}([M_u]_{u \in U}) \quad \text{(matching is a bijection)}.$$

This constraint makes the matching from U to U' a bijection, since both sets are of the same size.

B.4.3 Order of arrivals and departures

The following constraints guarantee that the order of events is in accordance with the fixed arrival and departure times of trains:

$$\begin{aligned}
Na_{u_1} < Na_{u_2} &\quad \forall u_1, u_2 \in U, Ca_{u_1} < Ca_{u_2}, \\
Nd'_{u'_1} < Nd'_{u'_2} &\quad \forall u'_1, u'_2 \in U', Cd'_{u'_1} < Cd'_{u'_2}, \\
Na_{u_1} < Nd'_{u'_2} &\quad \forall u_1 \in U, u'_2 \in U', Ta_{u_1} < Td'_{u_2}, \\
Nd'_{u'_1} < Na_{u_2} &\quad \forall u_1 \in U', u_2 \in U, Td'_{u'_1} < Ta_{u_2}.
\end{aligned}$$

The order between mutual arrivals (departures) is based on the order of trains in an array that has been ordered by time, where trains with equal arrival (departure) time have been ordered randomly. A time constraint would not work with arrival compositions that have equal arrival times, because then the order variables could mix up the units of the different compositions.

B.4.4 Combined units move together

The above constraints guarantee that the order values for units in different arrivals and departures are correctly ordered. For the order of units in the same composition, the following constraints are introduced:

$$\begin{aligned} Na_{u_2} &= Na_{u_1} + Sa_{u_1} \quad \forall u_1, u_2 \in U, \text{attachedAToBa}(u_1, u_2) \text{ is True}, \\ Nd'_{u'_2} &= Nd'_{u'_1} - Sa'_{u_1} \quad \forall u'_1, u'_2 \in U', \text{attachedAToBd'}(u'_1, u'_2) \text{ is True}. \end{aligned}$$

B.4.5 Order uniquess

The following constraint guarantees that all order values are different for arrivals and departures:

$$\text{allDifferent}([Na_u]_{\forall u \in U} \cup [Nd_u]_{\forall u \in U}).$$

B.4.6 Accessibility constraints

Each arriving and departing train have been assigned access points of the yard as part of the input. The following constraints impose that a train composition can only be stabled at a track that can be reached from its access point. It uses the possiblity of the most CP solvers to explicity specify the set of allowed assignments:

$$\begin{aligned} (ta_{tr}, Sa_{tr}) &\in Ra_{tr}, \quad tr \in C, \\ (td'_{tr'}, Sd'_{tr'}) &\in Rd'_{tr'} \quad tr' \in C'. \end{aligned}$$

B.4.7 Order of relocation departure and arrival when no relocation occurs

Each train unit is allowed to move once to a different track, allowing the arrival stabling track from a train unit to be different from its departure stabling track. Of course, if the arrival and departure tracks are the same, no movement is to be made. In this case, the values for the arrival side and order for the relocation are enforced to be equal to the values for the values of the arrival side and the order. Similary, for the departure. This results in the following constraints:

$$\begin{aligned} ta_u = td_u &\implies Sar_u = Sa_u, \\ ta_u = td_u &\implies Sdr_u = Sd_u, \\ ta_u = td_u &\implies Nar_u = Na_u, \\ ta_u = td_u &\implies Ndr_u = Nd_u. \end{aligned}$$

B.4.8 Accessibility constraints for relocation

Not every stabling track might be connected to any other stabling track. The next set of constraints imposes that every unit only moves between tracks that are connected:

$$(td_u, sdr_u, ta_u, sar_i) \in Rr \cup \{t, x, t, y | x, y \in \{-1, 1\}, t \in \text{tracks}\} \quad \forall u \in U.$$

The set R takes care off the allowed assignments when the tracks tdr_u and tar_u are different. The other set in the union is added to deal with the case that $td_u = ta_u$, and hence no relocation occurs. In that case, no further resctrictions on the sides need to be imposed, because sar_u and sdr_u are equal to sa_u and sd_u, respectively, and those have already been constrained.

B.4.9 Order uniqueness constraints for relocation

The following constraints make sure that the order values for proper relocations do not coincide with order values of other order values.

$$\begin{aligned}
&\text{allDifferent}(\{Nar_u | u \in U\} \cup \{Nd_u \in U\}),\\
&\text{allDifferent}(\{Ndr_u | u \in U\} \cup \{Na_u \in U\}),\\
&\text{allDifferent}(\{Nar_u | u \in U\} \cup \{Ndr_u \in U\}),\\
&\qquad Nar_{u_1} \neq Na_{u_2} \quad \forall u_1, u_2 \in U, u_1 \neq u_2,\\
&\qquad Ndr_{u_1} \neq Nd_{u_2} \quad \forall u_1, u_2 \in U, u_1 \neq u_2.
\end{aligned}$$

Together with the allDifferent-constraints for arrivals and departures, these constraints guarantee that the order values of all Na, Nd, Nar and Ndr are unique, unless no relocation occurs for a unit u and $Nar_u = Na_u$ and $Ndr_u = Nd_u$.

B.4.10 Relocate only when unit is present

The following constraints impose that a relocation of a unit can only happen when the unit is present.

$$\begin{aligned}
ta_u \neq td_u &\implies Na_u < Ndr_u, \quad u \in U,\\
ta_u \neq td_u &\implies Ndr_u < Nar_u, \quad u \in U,\\
ta_u \neq td_u &\implies Nar_u < Nd_u, \quad u \in U.
\end{aligned}$$

B.4.11 Relocate attached units simulateneously

The order values have now been specified at the unit level. However, if a unit is attached to the same unit (in the same order) both at arrival and departure, then the units should move together during relocations:

$$\begin{aligned}
&\text{attachedAToBd}(u_1, u_2) \implies Nar_{u_2} = Nar_{u_1} + sar_{u_1}, \quad \forall u_1, u_2 \in U \text{ and attachedAtoBa}(u_1, u2) \text{ is True},\\
&\text{attachedAToBd}(u_1, u_2) \implies Ndr_{u_2} = Ndr_{u_1} - sdr_{u_1}, \quad \forall u_1, u_2 \in U \text{ and attachedAtoBa}(u_1, u2) \text{ is True}.
\end{aligned}$$

B.4.12 No arrivals or departures during relocation

It is enforced that no other events can happen at a yard when attached units are relocating. The following constraint impose that no arrival or departure of units can occur during a relocation:

$$\begin{aligned}
&(Ndr_{u_1} < Na_{u_2}) \wedge (Nar_{u_1} > Na_{u_2}) \text{ is False}, \quad u_1, u_2 \in U,\\
&(Ndr_{u_1} < Nd_{u_2}) \wedge (Nar_{u_1} > Nd_{u_2}) \text{ is False}, \quad u_1, u_2 \in U.
\end{aligned}$$

B.4.13 No other relocations during relocation

It remains to impose that no other unit can relocate between the departure for relocation and the arrival after relocation of a combined set of units. This can be accomplished by imposing that, if train units relocate simultaneosuly, they have to be part of the same arrival and departure

composition and that their order of attachment is kept intact:

$$\begin{aligned}
Ndr_{u_1} < Nar_{u_2} < Nar_{u_1} &\implies (Ca_{u_1} = Ca_{u_2}) \wedge (Cd_{u_1} = Cd_{u_2}),\\
Ndr_{u_1} < Nar_{u_2} < Nar_{u_1} &\implies \text{attachedAToBa}(u_1, u_2) = \\
&\qquad \text{attachedAToBd}(u_1, u_2), \quad u_1, u_2 \in U,\\
Ndr_{u_1} < Ndr_{u_2} < Nar_{u_1} &\implies (Ca_{u_1} = Ca_{u_2}) \wedge (Cd_{u_1} = Cd_{u_2}),\\
Ndr_{u_1} < Ndr_{u_2} < Nar_{u_1} &\implies \text{attachedAToBa}(u_1, u_2) = \\
&\qquad \text{attachedAToBd}(u_1, u_2), \quad u_1, u_2 \in U.
\end{aligned}$$

B.4.14 Lock-ins

The primary objective of the model is to find a stabling plan that contains no lock-ins. In order to find an expression for the number of lock-ins, let us first consider just two units u_1 and u_2 and determine a boolean expression that tells whether one of the unit is blocking the other upon departure. In the way the order variables are formulated, it is irrelevant if u_1 and u_2 are part of the same train or not.

Assume for now that u_1 departs before u_2 and both units are parked on the same track at some certain moment. Otherwise, there is no lock-in. If u_2 arrives after u_1, then there can only be a lock-in upon departure of u_1 if u_2 enters the track at the departure side of u_1. If u_1 arrives later, then u_1 is not blocked only when it departs in the same direction. This result in the following boolean expression:

$$((Na_{u_2} > Na_{u_1}) \wedge (sd_{u_1} = sa_{u_2})) \vee ((Na_{u_2} < Na_{u_1}) \wedge (sd_{u_1} \neq sa_{u_1})).$$

for a lock-in between two units if u_1 departs before u_2. A similar expression can be formulated if u_2 departs before u_1. It is imposed that on none of the tracks any two train units have a lock-in.

B.4.15 Track capacity

Finally, it is required that the total length of all the trains parked on a track, never exceeds the track length. This has been implemented by a built-in cumulative constraint, that basically imposes for each track t:

$$\sum_{u \in U} (([Na_u \leq ti < Ndr_u] \cdot [ta_u = t]) \vee ([Nar_u \leq ti < Nd_u] \cdot [td_u = t])) \cdot l_u \leq l_t,$$

for all *time points* ti, that are chosen such that there is a t_i between every two events.

B.5 Objectives

The objective consists of a weighted combination of

1. the number of splits,
2. the number of combines,
3. the number of relocations.

The number of relocations can be counted by counting the number of units that get parked on two different tracks.

A split of combine between two units is determined by the following boolean expressions:

$$\begin{aligned}
&\text{attachtedAToBd}(u_1, u_2) \wedge \neg\text{attachedAToBa}(u_1, u_2), \quad \text{(combine)}\\
&\text{attachedAToBa}(u_1, u_2) \wedge \neg\text{attachtedAToBd}(u_1, u_2) \quad \text{(split)}.
\end{aligned}$$

C Mathematical description of the stabling algorithm circuit constraint model

The mathematical description of the circuit constraint model can be read here.

C.1 Notation and assumptions

The following notational conventions are used throughout the description of the model:

- The set of arriving train units is denoted by U and the set of departuring train units by U'. Assumed is that $|U| = |U'|$.
- The set of tracks is denoted by T.
- It is assumed that each track has an A-side and a B-side and that one always moves from the A-side of one track to the B-side of another, and never from A to A or B to B. Thus far, this has shown to be a valid assumption.
- The train units in a train composition are always listed from A to B.
- It is assumed that any split of a train composition happens as soon as the train arrives at its arrival stabling track, and any combine happens just before the combined train composition departs from its departure stabling track. Consequently, the train composition of a relocation is always known, because a relocation always happens after the split and before the combine, if any.
- It is assumed that all train arrival and departure times are unique.

C.2 Variables

The variables are similar to the original stabling algorithm, only this time a train unit has variables for each track in the yard. That is, for each track and train unit there is a variable for the arrival side, departure side, time of arrival at the track and time of departure from the track. Furthermore, there are variables that determine which tracks a unit visits and in what order. Finally, a matching variable is needed to matching the arriving unit to one of the departing units.

For every train unit a undirected graph is constructed with a node for every stabling track. In addition, a fictive track is used, that represents all gateways. Between each pair of nodes there are arcs that represent movements between the tracks. More specifically, an arc $t_{utst's'}$ between two nodes t and t represents a movement from track t to track t' that exits track t at side s and enters track t' from side s'.

C.2.1 Unit track variables

The following decision variables are defined for each arrival unit and track.

1. $Sa_{ut} \in \{-1, 1\}$ Side of arrival of u at track t ,
2. $Ta_{ut} \in \mathbb{N}$ Time of arrival of u at track t,
3. $Sd_{ut} \in \{-1, 1\}$ Side of departure of u at track t,
4. $Td_{ut} \in \mathbb{N}$ Time of departure of u at track t,
5. $X_{ut} \in \{0, 1\}$ Whether u visits track t.

C.2.2 Unit variables

The folowing decsion variables are defined for each arrival unit.

1. $M_u \in [0, \text{number of units} - 1]$ The index of the matched departure unit of u, domain restricted to allowed matches.

C.2.3 Auxiliary unit variables

The variables in this section are auxiliary variables to simplify expressions in the model.

1. $Td_u \in \mathbb{N}$ The time of departure of the matched unit of u,
2. $Pd_u \in \mathbb{N}$ The position in the departure composition of the matched unit of u,
3. $Mb_{uu'} \in \{0, 1\}$ Whether unit u and u' are matched,
4. $\text{attached}_{u_1 u_2} \in \{0, 1\}$ Whether arrival units u_1 and u_2 are attached A to B at departure.

C.2.4 Train variables

The following variables are introduced for trains.

1. $A_{cts} \in \{0, 1\}$ Whether arriving composition c arrives at track t and side s,
2. $D_{cts} \in \{0, 1\}$ Whether departing composition c departs from track t and side s.

C.2.5 Arcs

The graph is modeled for each unit, existing of the following arcs which the unit can take.

1. $aa_{uts} \in \{0, 1\}$ Whether the arrival arc of unit u to track t and side s is used,
2. $da_{uts} \in \{0, 1\}$ Whether the departure arc of unit u from track t and side s is used,
3. $ta_{utst's'} \in \{0, 1\}$ Whether unit u uses the transfer arc to move from track t and side s to track t' and side s'.

C.3 Constraints

The model contains the following constraints.

C.3.1 Matching constraint

$$\text{allDifferent}([M_u]_{u \in U}).$$

This constraint enforces that each unit is matched to an unique departure unit. This is done using the global constraint $allDifferent$, which enforces that all variables in the given set are unique, forcing each unit to have a correct matching. Some arrival units might only be allowed to match to specific departure units, this is enforced by restricting the domains of the mathing variables M_u to the allowed values.

C.3.2 Circuit constraint

$$\text{Circuit}([aa_{uts}]_{t\in T,s\in\{-1,1\}} \cup [da_{uts}]_{t\in T,s\in\{-1,1\}} \cup [ta_{utst's'}]_{t\in T,t'\in T,s\in\{-1,1\},s'\in\{-1,1\}}), \quad \forall u \in U.$$

These constraints enforce that each unit takes a correct path through the graph, starting and ending at the fictive track, using Circuit constraints. A Circuit constraint is a global constraint in the OR-Tools CP-SAT solver, which enforces that a circuit is taken through the graph of at least 2 nodes. The graph is given by the set of arcs.

C.3.3 Arrival / departure arc happens at arrival / departure time

$$aa_{uts} \implies Ta_{ut} = \text{arrivalTime}(u) + (2s-1) * \text{posInTrain}(u), \quad \forall u \in U, t \in T, s \in \{-1,1\}, \quad (1)$$

$$aa_{uts} \implies Sa_{ut} = s, \quad \forall u \in U, t \in T, s \in \{-1,1\}, \quad (2)$$

$$aa_{uts} \implies X_{ut} = 1, \quad \forall u \in U, t \in T, s \in \{-1,1\}, \quad (3)$$

$$da_{uts} \implies Td_{ut} = Td_u + (2s-1) \cdot Pd_u, \quad \forall u \in U, t \in T, s \in \{-1,1\}, \quad (4)$$

$$da_{uts} \implies Sd_{ut} = s, \quad \forall u \in U, t \in T, s \in \{-1,1\}. \quad (5)$$

Constraints (1) enforce that if arrival arc aa_{uts} is used, the arrival time of unit u at track t is set taking into account side s of the arrival. Here arrivalTime(u) denotes the time of arrival of the train containing u, and posInTrain(u) denotes the index of u in its containing train. The position in the train is taken into account to make sure the units enter the track in the correct order. Constraints (2) enforce that if arrival arc aa_{uts} is used the arrival side of unit u at track t is equal to s. Constraints (3) enforce that if an arrival arc to track t is used by u, track t is also used by u. Constraints (4) enforce that if unit u uses a departure arc from track t, the departure time of unit u at track t is set based on the departure time of the matched unit, position in the train and side of departure. Constraints (5) enforce that the departure side from track t is set to s if departure arc da_{uts} is used.

C.3.4 Accessiblilty constraints at arrival and departure

$$\begin{aligned} \text{containingTrainCantAccess}(u,t,s) &\implies aa_{uts} = 0, \quad \forall u \in U, t \in T, s \in \{-1,1\}, \\ \text{containingTrainCantAccess}(u',t,s) \wedge Mb_{uu'} &\implies da_{uts} = 0, \quad \forall u \in U, u' \in U', t \in T, s \in \{-1,1\}. \end{aligned}$$

These constraints ensure that arrival and departure arcs are only used if the train in which unit u (or the matched unit u' for the departure) is contained can access track t from side s.

C.3.5 Track capacity

The constraint

$$\sum_{u\in U} [Ta_{ut} \le ti < Td_{ut}] \cdot l_u \cdot X_{ut} \le l_t, \quad \forall t \in T, \forall ti \in \text{timepoints}(t)$$

enforces that the sum of the lengths of units parked at a track t do not exceed the length of the track l_t at any point in time. Here timepoints(t) is a function that returns all unique timepoints of track t at which the capacity should be constrained. This is handled by a built-in cummulative constraint.

C.3.6 Transfer arcs consistency

$$ta_{utst's'} \implies Ta_{ut'} = Td_{ut} + d + \text{extra}(u, s, s'), \quad \forall t \in T, t' \in T, s \in \{-1, 1\}, s' \in \{-1, 1\}, \tag{6}$$

$$ta_{utst's'} \implies Sa_{ut'} = s', \quad \forall t \in T, t' \in T, s \in \{-1, 1\}, s' \in \{-1, 1\}, \tag{7}$$

$$ta_{utst's'} \implies Sd_{ut} = s, \quad \forall t \in T, t' \in T, s \in \{-1, 1\}, s' \in \{-1, 1\}, \tag{8}$$

$$ta_{utst's'} \implies X_{ut'} = 1, \quad \forall t \in T, t' \in T, s \in \{-1, 1\}, s' \in \{-1, 1\}. \tag{9}$$

Constraints (6) ensure that when a unit u transfers between tracks t and t', the arrival and departure times of these tracks are set accordingly. The variable d denotes the duration of the movement, which currently is a constant. The $\text{extra}(u, s, s')$ function makes sure the unit order for combined movements is adhered to. If two units are moved with a reversal on another track (which may not be explicitly modeled), the first unit to leave the origin should be the last unit to arrive at the destination. Constraints (7) and (8) enforce that arrival and departure sides are set correctly based on the chosen transfer arc. Constraints (9) enforce that track t is used by u if u transfers to track t.

C.3.7 Unique arrival and departure times per track

$$\text{allDifferent}([Ta_{ut}, Td_{ut}]_{\forall u \in U}), \quad \forall t \in T.$$

This ensures that all arrival and departure times on a track are different. This is required to properly determine the order of units on a track, and therefore to determine lock-ins.

C.3.8 Combined units move together at arrival and departure

$$\begin{aligned} aa_{uts} &= A_{cts}, \quad \forall u \in U, c = \text{containingTrain}(u), t \in T, s \in \{-1, 1\}, \\ Mb_{uu'} &\implies da_{uts} = D_{cts}, \quad \forall u \in U, u' \in U', c = \text{containingTrain}(u'), t \in T, s \in \{-1, 1\}. \end{aligned}$$

These ensure that each unit uses the same arrival and departure arcs as the train composition it is contained in.

C.3.9 Combined units move together during relocations

$$\text{alwaysAttached}(u_1, u_2) = \text{attachedAtArrival}(u_1, u_2) \wedge \text{attached}_{u_1 u_2}$$

$$\text{alwaysAttached}(u_1, u_2) \implies X_{u_1 t} = X_{u_2 t}, \quad \forall u_1 \in U, u_2 \in U, t \in T, \tag{10}$$

$$\text{alwaysAttached}(u_1, u_2) \wedge X_{u_1 t} \implies Ta_{u_1 t} = Ta_{u_2 t} - Sa_{u_1 t}, \quad \forall u_1 \in U, u_2 \in U, t \in T, \tag{11}$$

$$\text{alwaysAttached}(u_1, u_2) \wedge X_{u_1 t} \implies Td_{u_1 t} = Td_{u_2 t} + Sd_{u_1 t}, \quad \forall u_1 \in U, u_2 \in U, t \in T, \tag{12}$$

$$\text{alwaysAttached}(u_1, u_2) \wedge X_{u_1 t} \implies Sa_{u_1 t} = Sa_{u_2 t}, \quad \forall u_1 \in U, u_2 \in U, t \in T, \tag{13}$$

$$\text{alwaysAttached}(u_1, u_2) \wedge X_{u_1 t} \implies Sd_{u_1 t} = Sd_{u_2 t}, \quad \forall u_1 \in U, u_2 \in U, t \in T. \tag{14}$$

When two units are attached both at arrival and departure it is enforced that they never split. Constraints (10) enforce that when two units are attached both at arrival and departure they use the same tracks. Constraints (11), (12), (13), and (14) ensure that the attached units arrive and depart from tracks at the same time and in the correct order, such that they do not split.

C.3.10 Constraints on auxiliary variables

$$Td_u = \text{Element}([\text{departureTime}(u')]_{u' \in U'}, M_u), \quad \forall u \in U, \tag{15}$$

$$Pd_u = \text{Element}([\text{posInTrain}(u')]_{u' \in U'}, M_u), \quad \forall u \in U, \tag{16}$$

$$Mb_{uu'} = \text{Equals}(M_u, indexOf(u')), \quad \forall u \in U, u' \in U', \tag{17}$$

$$attached_{u_1 u_2} = \text{Element2D}([\text{attachedAtDeparture}(u'_1, u'_2)]_{u'_1 \in U', u'_2 \in U'}, M_{u_1}, M_{u_2}), \quad \forall u_1 \in U, u_2 \in U. \tag{18}$$

Constraints (15) enforce that the departure time of unit u is set correctly based on the departure time of the matched unit. This is done using an Element constraint. This is a global constraint supported by the solver, which constraints the result to the value in an array at a certain index. Thus, in this case Td_u is constrained to the value in $[\text{departureTime}(u')]_{u' \in U'}$ at index M_u.

Constraints (16) ensure that the position in the departure composition of unit u is set based on the position of the matched unit. Constraints (17) converts the index based matching to a boolean matching. Constraints (18) ensure that two units are attached at departure iff their matched units are attached at departure.

C.4 Objectives

The model is solved in two phases. This seems to result in much better performance than solving the model in it's entirety once. This is especially the case for the time required to find the first solution for the problem.

C.4.1 Phase 1 objective

In the first phase, all hard constraints are present with a simple objective of minimizing the total number of transfer arcs taken (19), and thus minimizing the number of movements.

$$R = \sum_{\forall t \in T, t' \in T, s \in \{-1,1\}, s' \in \{-1,1\}, u \in U} ta_{utst's'} \tag{19}$$

$$\text{Minimize} \quad w_r \cdot R \tag{20}$$

C.4.2 Phase 2 objective

The results of the first phase are used as a warm start for the second phase. In that phase the full objective function is used, which also includes the number of splits (21), combines (22), and dislocated units (i.e. lock-ins) (23) in the objective.

Function (21) sums the number of pairs of units which are attached at arrival, but not at departure, and therefore split. Function (22) sums the number of pairs of units which are not attached at arrival, but are at departure, and therefore combine. Function (23) sums the number of pairs of units which are dislocated for each track. This is done, in similar fashion to the previous implementation, by case distinction enumerating all ways a pair can be dislocated, taking into account their arrival and departure time and side.

$$S = \sum_{u_1 \in U, u_2 \in U} \neg\text{attached}_{u_1 u_2} \wedge \text{attachedAtArrival}(u_1, u_2) \tag{21}$$

$$C = \sum_{u_1 \in U, u_2 \in U} \text{attached}_{u_1 u_2} \wedge \neg \text{attachedAtArrival}(u_1, u_2) \tag{22}$$

$$D = \sum_{u_1 \in U, u_2 \in U, t \in T} \text{dislocatedOnTrack}(u_1, u_2, t) \tag{23}$$

$$\text{Minimze} \quad w_r \cdot R + w_s \cdot S + w_c \cdot C + w_d \cdot D \tag{24}$$