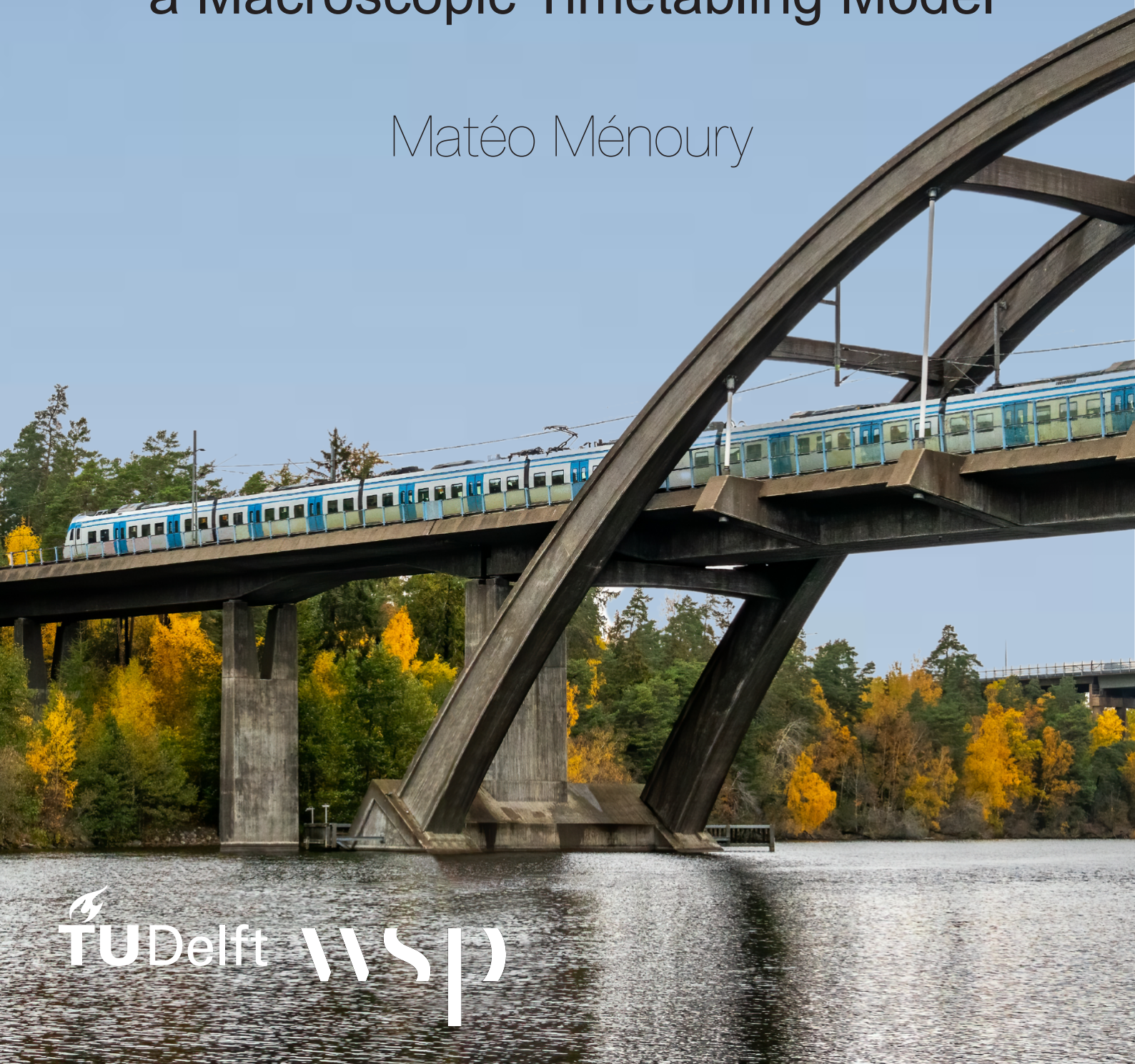
Optimizing Railway Infrastructure to Meet Future Demand:

a Macroscopic Timetabling Model

Matéo Ménoury



TU Delft wsp

Optimizing Railway Infrastructure to Meet Future Demand:

a Macroscopic Timetabling Model

by

Matéo Ménoury

To obtain the degree of Master of Science
at the Delft University of Technology.

To be defended publicly on Friday October 4, 2024 at 14:00.

Project duration:	March 2024 – September 2024	
Student:	M.E.N. Ménoury	5839505
Thesis committee:	Prof. Dr. R.M.P. Goverde	CEG, chairman
	Dr. J.A. Annema	TPM, supervisor
	Dr. P.S.A. Stokkink	TPM, supervisor
	Ir. K. Bediru Seid	WSP, supervisor

TU Delft WSP

Preface

This report represents the culmination of my master's thesis, marking the end of my master's program in Transport, Infrastructure, and Logistics at the Delft University of Technology. The project has been carried out in the format of a 6-month full-time internship at WSP in Stockholm, within the railway department. The research's primary goal is to assist planners in optimizing the construction of new tracks to meet the increasing long-term demand. This has been done by developing an innovative and flexible macroscopic timetabling model inspired by previous works found in the existing literature.

From a personal point of view, this research has been motivated by a strong interest in railway operations and planning. This field captivates my attentiveness for the role it plays in shaping sustainable and efficient transportation systems. I approached this project with the intention of gaining experience and knowledge in this field in which I am willing to start my post-graduate life. Applying the knowledge and methodology acquired during my studies has been stimulating and enriching.

I express my sincere gratitude to my TU Delft supervisors Prof. Dr. R.M.P. Goverde, Dr. J.A. Annema, and Dr. P.S.A. Stokkink for their guidance throughout this research. Further, this project could have not been conducted without the support of my company supervisor, Ir. K. Bediru Seid. His recommendations and feedback were particularly appreciated and helpful. Having both academic and professional perspectives permits me to understand the needs and requirements of the field. My appreciation also goes out to every WSP employee who participated in my meetings or interviews and offered me an external point of view. I would like to thank WSP and the railway department for their trust and warm welcome, working at WSP has always been a pleasure. Lastly, I am thankful to my friends and family for their insightful feedback, support, and encouragement all along the research.

This research has presented numerous obstacles. The initial challenge has been to define and narrow down the topic. By going back and forth between my different supervisors, I had to make sure it satisfied both the academic and company requirements. The formulation of the mathematical model and its implementation have also been quite challenging. Every possible scenario had to be taken into account and well-defined to make sure that the model was as flexible and realistic as possible. I have learned to always take a step back and reflect on the existing literature, the methodology I apply, and the results I obtain.

I am very grateful to have had the opportunity to conduct this research and for the amazing people I have met during my stay in Stockholm. I hope you find this work both engaging and insightful. Enjoy your reading.

Matéo Ménoury

Stockholm, August 2024

Executive summary

Demand for both passenger and freight railway services is increasing in Sweden. However, railway infrastructure is limited and many lines are still constituted of single tracks. Therefore, many projects are conducted across the country to upgrade the existing infrastructure from single to double tracks or from double to four tracks. These modifications require high investments but are constrained by limited financial resources. Therefore, a long-term investigation and strategic line planning are required to maximize the satisfaction of the future demand, while minimizing the construction costs. Usually, the line planning process focuses on identifying the bottleneck portions of a line and upgrading their capacity by constructing additional tracks. In the short term, timetabling simulation usually adapts the demand to the infrastructure by canceling trains if the infrastructure cannot satisfy the demand. However, in long-term planning and timetabling, it is reasonable to investigate all possible infrastructure upgrades, and infrastructure should therefore be considered as flexible.

This research aims to develop a model optimizing the construction of new tracks on a given railway network, to ensure the feasibility of the long-term demand while minimizing the construction costs. Along the research, the demand is limited by a list of trains and their characteristics: the passenger origin-destination demand is not considered. A solution is feasible if every train request can be scheduled according to the traffic regulations. The optimal solution minimizes the construction costs and maximizes the benefits of the upgrade on the traffic. To evaluate the benefits of a given improvement on the traffic, the operation costs are introduced. The total travel times of a train running on the network are converted into monetary value. The smaller the travel time and the operation costs, the more efficient the traffic. Therefore, the model aims to minimize both operation and construction costs.

The developed model gives a conflict-free timetable and any infrastructure upgrades required to ensure the feasibility of future demand. The model is designed at the macroscopic level: the infrastructure is only depicted with nodes and edges and the signaling characteristics such as the different block sections among the tracks are omitted. The model applies the Periodic Event Scheduling Problem (PESP) to railway operations, adds an objective function on top of it, and extends it by introducing possible infrastructure upgrades. The model is formulated as a Mixed Integer Linear Programming (MILP) problem.

A timetable is conflict-free if every train can run according to its planned schedule without meeting another train running on the same track. A conflict can happen between two trains running in the same or the opposite direction. These conflicts must be detected and restricted according to the number of tracks on the infrastructure. There cannot be any conflicts on a single-track portion of a line. Train running in opposite directions can meet on a double or four-track portion of the line because in that case, each track is reserved for one direction. Additionally, a four-track portion allows a train to overtake another slower one running in the same direction. Finally, to avoid any conflict, the model considers that, at any time, there cannot be more trains occupying a station than the number of tracks that same station has. The conflicts are detected by applying the flexible overtaking constraints developed by Zhang and Nie (2016), but this approach is extended by restricting the different types of conflict according to the number of tracks on each portion of the line.

The model is implemented in Python and solved with Gurobi. The model is applied to a small portion of the Mälarbanan. This line is located northwest of Stockholm and has a strategic position. Several studies suggest that the high-speed trains connecting Stockholm and Oslo should run on this line, instead of running south of Stockholm. However, an important portion of the line is still constituted of single tracks. This portion of the line is tested with the designed model to assess whether it could accommodate the demand. Three different scenarios are formulated: 2022 demand, 2040 demand, and 2040 demand with additional high-speed trains.

Simulation of the 2022 demand shows that the optimal state of the infrastructure is the current one: there is no need to build additional tracks. Nevertheless, results show that infrastructure upgrades are required to meet long-term demand. In the scenario where no high-speed trains run on the line in 2040, only one small portion of the line should be upgraded to 2040. In the scenario where high-speed trains are relocated on the Mälarbanan, the majority of the line should be upgraded to double tracks. These results demonstrate the model's strengths: if current studies suggest upgrading the entire portion of the line to double tracks, the solution suggested by the model decreases by 67% the construction costs while increasing the operation costs by only 11%. The optimal solutions are simulated at the microscopic level within RailSys. At this level, all the network characteristics and signaling are known. These simulations confirm the validity of the model, as all generated timetables are conflict-free, except the section requiring double-track upgrades, which aligns with expectations.

While the model demonstrates promising results, further testing is required under more complex operating conditions such as interactions between the different train lines (transfers, turnarounds) or higher frequencies. The practicality of the solution proposed by the model could also be tested to check whether the suggested infrastructure improvements are technically feasible. Moreover, this model approximates the various operation and construction costs and does not fully capture the real-world complexity of railway operation and planning costs. Further development of the model could investigate the possibility of introducing flexible headway constraints that could vary according to the number of tracks of the different portions of the network. The potential for building additional meeting stations on a line could also be investigated and should improve the model's solutions by reducing the construction costs. Moreover, further research should investigate the robustness of the timetable produced by the model.

In conclusion, this research goes beyond the current line planning methods and does not simply suggest the construction of new tracks on the bottleneck portion of a line. Instead, the model explores all possible infrastructure improvements and minimizes the overall construction and operation costs. This research contributes to an extension of the current timetabling and line-planning approach by formulating a flexible optimization model. The model provides a solution to the challenging increasing railway demand.

Contents

1 Introduction

1.1. Background information

Rail travel is seen as a sustainable alternative for road and air travel on a domestic and European scale. Indeed, even when considering the long-term vehicle and infrastructure life cycle emissions, traveling with high-speed rail emits about 5.6 times less CO_2 than flying and 2.8 times less than driving a diesel private car (de Bortoli and Féraille (2024)). The demand for both passenger and freight rail is increasing in Europe, particularly in Sweden. The country has seen an increase of passenger-kilometers transported of 24% and an increase of tonnes of goods transported of 4% from 2012 to 2019 (Eurostat (2024a) and Eurostat (2024b)) and the railway sector is recovering from the COVID crisis. However, the increasing demand puts pressure on the limited existing rail infrastructure, which is often already used at full capacity. To tackle this issue, several projects have been launched across the country. However, such projects are very costly and require high investments on a long-term scale. For instance, the 12km line between Malmö and Lund was upgraded from two to four tracks in 2023 for a cost of about SEK 5.4 billion (€ 470 million) (Trafikverket (2024b)). Thus, satisfying the demand on a limited infrastructure and with a restricted budget is one of the main challenges faced by railway planners. For this reason, it is fundamental to rigorously plan, design, and evaluate lines and their consequences on capacity utilization.

1.2. Problem description

WSP is one of the world's leading consulting companies specializing in various fields, including Transport & Infrastructure. The Swedish branch of the company has broad experience in traffic, capacity studies, and line planning within the railway sector and works in close collaboration with the Transport Administration. They intensively use the microscopic simulation software RailSys, which is increasingly being used in Sweden. However, this microscopic tool presents many disadvantages. First, it requires many input parameters (rolling stock characteristics, detailed infrastructure characteristics, signaling, interlocking systems, etc.), and those data are not always available or accurate when the company investigates long-term projects. Thus, incomplete data can lead to unrealistic simulation outcomes. Therefore, this software is not adapted for long-term strategic planning: the user, willing to evaluate different network scenarios (such as building new tracks in a station) has to design the detailed level of infrastructure and signaling in RailSys. Secondly, it requires long computer running times on complex networks, and it is therefore time-consuming and not user-friendly for conducting capacity analysis on large networks.

For those reasons, the interest in macroscopic simulation models is growing in Sweden, and the Transport Agency conducts many projects aiming to modernize the interface between railway operators and the Transport Administration by developing new capacity allocation processes (Palmqvist et al. (2018)). Market-adapted Planning of Capacity (MPK) has therefore been implemented in 2022 and is used in tactical and operational planning, but not for long-term strategic planning (Trafikverket (n.d.-c)). In this context, WSP is also seeking to develop its own macroscopic timetabling model, with

which the consultants would be able to quickly assess large network infrastructure scenarios, to focus on long-term strategic planning (2050, 2060, ...).

1.3. Scope of study

The railway planning process, as described by Lusby et al. (2011) in Figure 1.1, follows three steps: strategic, tactical, and operational. The strategic level relates to the development of new infrastructures over several years. Based on the demand forecasting and analysis, a capacity analysis is conducted to find the bottleneck lines of the network and to identify the improvements that should be made to satisfy the long-term demand. The tactical level refers to the production of timetables, usually on a one-year scale. Finally, the operational level focuses on real-time management and aims to adapt unforeseen disturbances at the tactical level to real-time operations (Palmqvist et al. (2018)). These levels are interdependent, and a poorly designed upper level can have significant consequences on the lower levels.

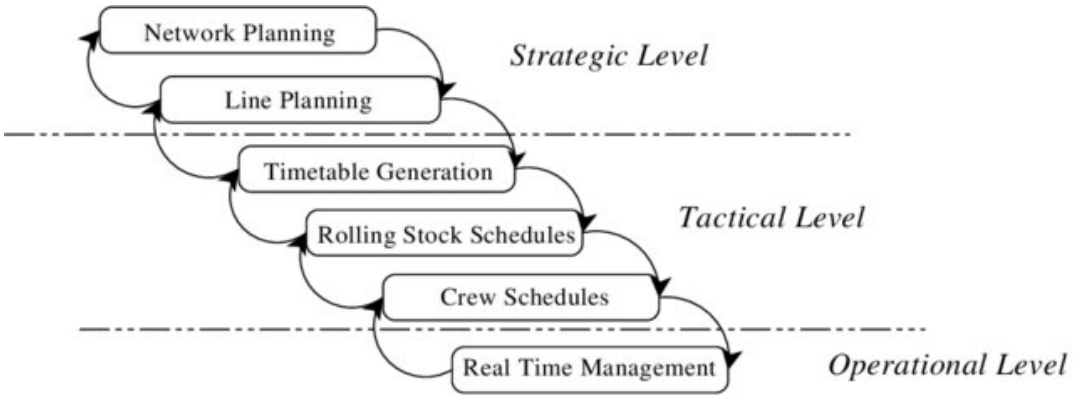


Figure 1.1: The railway planning process (Lusby et al. (2011))

This research is at the crossroads of strategic and tactical levels. A timetable simulation model is developed at the macroscopic level. Among the various performance indicators, the research considers the timetable feasibility. Indeed, the stability, robustness, resilience, infrastructure occupation, and energy efficiency of a timetable need further investigations at the microscopic level or real-time traffic information and are usually investigated at a later stage of the line planning process. The feasibility of the long-term demand is tested on the current infrastructure, and the construction of new tracks is proposed in case the current infrastructure does not fit the demand. Thus, this research focuses on the trade-off between construction costs and feasibility of the long-term demand. Along the research, the term ”demand” only refers to the number and characteristics of trains requesting access to a railway infrastructure. No passengers’ origin-destination matrices are investigated in this study. In the same way, when mentioning ”infrastructure improvement”, or ”infrastructure upgrade”, this research refers to the construction of new tracks on an already existing line or station. Hence, the possibility of improving a line by upgrading its signaling or the possibility of building a line between two unconnected stations is not considered.

1.4. Research questions

This research aims to develop a macroscopic timetabling model optimizing the construction of new tracks on a given railway network, to ensure the feasibility of the long-term demand. To do so, the infrastructure constraint is considered as flexible and can be relaxed to find the optimal trade-off between the construction costs required for the infrastructure upgrades and their benefits on the operation costs. This is achieved by answering the following main research question:

How can the construction costs of new tracks be minimized while ensuring the feasibility of the long-term demand?

The following sub-questions jointly answer the main research question:

1. *What are the requirements and constraints to ensure the feasibility of a given timetable?*
2. *How to evaluate the minimum number of tracks required on a line based on the number and types of conflicts detected in the timetable?*
3. *How to identify and implement necessary infrastructure upgrades?*

2

Literature review

2.1. Methodology

To formulate the research questions and define the scope of study, the research topic is narrowed down by reviewing the literature. The current state of the art in Sweden in matters of signalling, timetabling simulation, and line panning are analyzed and discussed, and knowledge gaps are identified. The literature is reviewed by following the methodology developed by Van Wee and Banister (2016): papers are searched using SCOPUS and ResearchGate, focusing on publications from 2015 onward, even though some articles published before this date are also reviewed. The snowballing methodology, especially backward snowballing is used. It consists of looking at the reference list of a relevant paper and tracking down those cited studies. Moreover, this research also references many reports from Trafikverket, the Swedish infrastructure manager (IM). Finally, concepts studied during the courses TIL4030-20 *Research and Design Methods*, CIEQ6233 *Railway Operations and Control* and CIEM6301 *Railway Traffic Management* are used.

2.2. Railway market and governance in Sweden

In Sweden, railway operations are categorized into five different services: high-speed trains, long-distance intercity trains, commuter trains, regional trains, and finally, freight trains (Trafikverket (2024a)). Freight services hold an important share of total railway operations. In 2023, freight operations constituted 22% of the total train-kilometers (sum of the distances traveled by all trains). When measured in millions of passenger-kilometers and tonne-kilometers, freight reaches about 62% of total railway operations (Analys (2024)). The government aims to increase the share of rail in total transport and therefore invests in many projects aiming to increase the reliability, capacity, and efficiency of rail transport. Even though both types of service share the same infrastructure and compete to get access, freight and passenger operations do not have the same constraints and requirements. Thus, the IM and railway planners must consider both services when designing new lines and timetables.

Following the EU policy, the Swedish railway passenger market has been open to competition since 2010. As of 2021, fifty railway undertakings (RU) operate on the tracks, making the country the most open railway market in Europe (Trafikverket (n.d.-a)). However, the Swedish State Railways (SJ) is still the most important RU in the country. The Swedish Transport Administration (Trafikverket) is the infrastructure manager. Once a year, the Transport Administration receives requests for the capacity for the next year and is responsible for scheduling the timetables. In case of conflicting requests from different RUs, a discussion is engaged. If the discussion turns out to be unsuccessful, the Transport Administration prioritizes the trains according to different criteria (Vigren (2017)). The Transport Administration charges the different RUs operating on their network. The costs are about 0.0152 SEK per gross tone kilometer added to approximately 3.33 SEK per train kilometer (Trafikverket (2023b)). Additional fees might apply in case of delays, train cancellation, or use of specific infrastructure such as bridges or tunnels. Moreover, based on the Swedish Government's transport policy goals, the Transport Administration is responsible for long-term planning. Every four years, they propose a National

Plan for Transport which draws the national strategy on maintenance, modernization, and construction planning for the upcoming 12 years (Trafikverket (2022)). Finally, the Swedish Transport Agency (Transportstyrelsen) is in charge of regulating, examining, and granting access to the Swedish infrastructure.

2.3. Railway traffic management

Railway operations must comply with the signalling principles, which rely on train separation. Tracks are divided into block sections, which can be occupied by one and one train only at the same time. In a fixed block operation, the block sections are limited by signals that provide movement authority to enter the block section. This movement authority must be given before the train has reached the braking distance in its approach to the section, and once the train ahead cleared the block. Thus, the blocking time refers to the total time the block section is allocated to a train movement, and therefore blocked for other trains. Once the train has cleared the section and all signals have been reset to normal position, the blocking time ends and the process repeats so that movement authority can be given to the next train to enter the section. Hansen and Pachl (2008) define the different components of the blocking time as follows:

- **Setup/clearing time:** time to clear the signal.
- **Sight and reaction time:** time for the driver to view the clear aspect at the signal that gives the approach indication to the main signal at the entrance of the block section.
- **Approach time:** time for the train to run between the approach signal and the first block signal.
- **Running time:** time for the train to run between the block signals.
- **Clearing time:** time to clear the block section and the overlap with the full length of the train.
- **Release time:** time to unlock the block system.

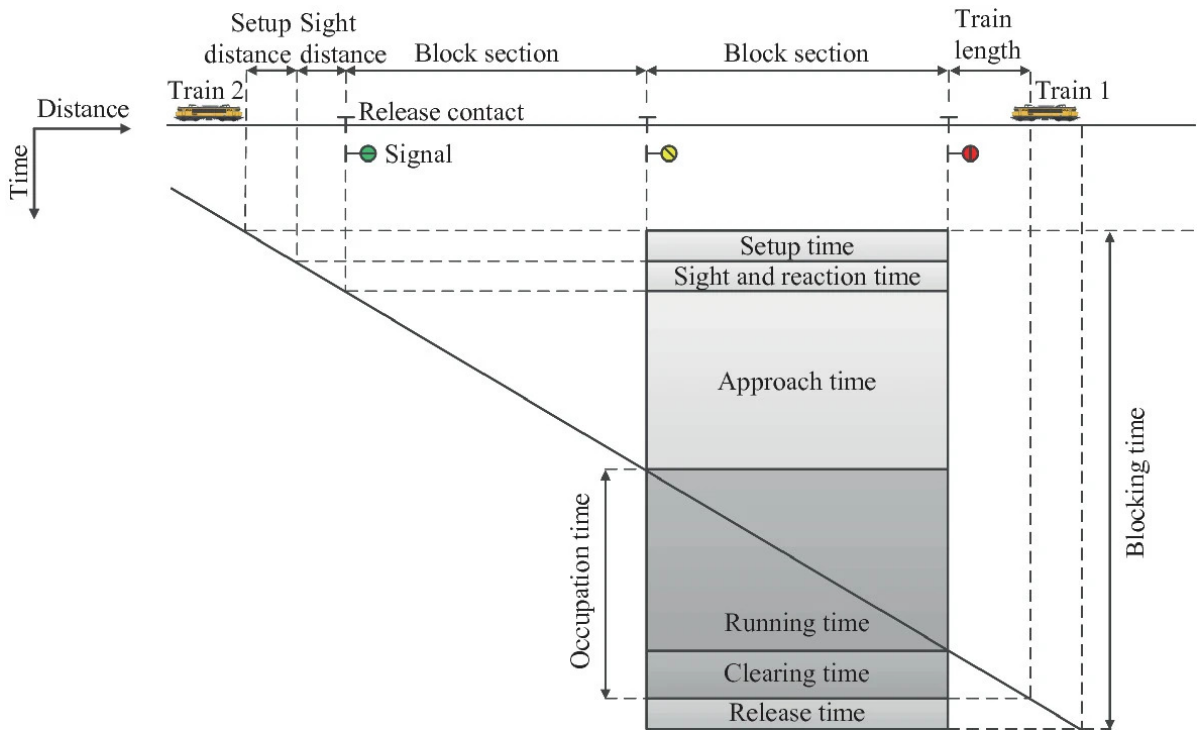


Figure 2.1: Blocking time components (Goverde and Scheepmaker (2023))

To control the train movements and ensure safety, instructions are given through signals, lights, and sign systems. Most of the Swedish railway network uses the ATC-2/ATC-S signalling system, which was introduced in the 1980s. This system is similar to ETCS Level 1: an onboard ATC (Automatic Train Control) system ensures that the train reduces its speed to the release speed before the target point after which the driver is responsible for braking the train to stop ahead of the signal. Furthermore, ATC-2 is generally incompatible with the ERTMS, although trains must be able to operate on both ERTMS-equipped infrastructure and existing ATC lines. Thus, a Specific Transmission Module (STM) has been developed to interface ATC-2 with ETCS. The STM unit reads data from the existing trackside equipment and converts it into a format that is readable by the new onboard system. This allows the ERTMS system to be introduced into the Swedish network in a phased manner, until its full implementation, which is expected by 2030 ERTMS (2021)). The ERTMS (European Rail Traffic Management System) is a signalling and speed control system being implemented by the European Union to create an efficient, safe, and interoperable railway system within European countries. It is composed of the European Train Control System (ETCS) and the Global System for Mobile Communications-Railway (GSM-R) (ERTMS (n.d.)).

2.4. Railway timetabling

2.4.1. Timetabling objectives and requirements

Hansen and Pachl (2008) describes the methods of railway timetabling and optimization. A timetable consists of basic train processes (running, dwelling, turning, etc.) and their interactions (passing, crossing, overtaking, etc.). A timetable aims to optimize the use of infrastructure by coordinating the train paths, to ensure the safety, predictability, and control of rail traffic, as well as to inform the passengers and schedule the rolling stocks and crew.

The type of operations has to be considered during the timetabling process. Indeed, freight and passenger operations do not have the same constraints and requirements. On one hand, passenger trains have to satisfy the demand and passengers' expectations: small travel and transfer times, high frequencies, but also easily memorable timetables. Thus, a preference is given to periodic timetables, i.e. timetables that repeat every given time, usually one hour. Moreover, priority is given to minimizing the travel time while ensuring stops are minor stations. Finally, the timetabling process has to consider peak and non-peak hours flows to ensure demand satisfaction. On the other hand, freight trains don't need to stop at minor stations, and minimizing their travel time is not a priority. Since periodic timetables require higher operation costs and are more difficult to plan in a competitive market where different RUs request access to the same infrastructure, the freight train timetables are aperiodic (each train is scheduled individually). Finally, they are usually scheduled during non-peak hours to not disturb passenger operations (Polinder et al. (2020)).

Solinen (2022) introduces the various traffic regulations currently in use in Sweden. The dwell times depend on the number of passengers getting in and out of the train, on the size of the train, and on the time of the day (peak or off-peak hours). Passenger trains over 300 meters or passenger trains with manual door closing must have a planned dwell time of at least two minutes. Passenger trains over 400 meters must have a planned dwell time of at least three minutes. The minimum running supplement, also called standard allowance in Sweden, is set to 8% of the minimal technical running time. It is currently being replaced on some lines by a kilometer-based supplement that depends on the type of train, added to a supplement depending on whether it is running on single-track or multi-tracks. For instance, the X2 passenger train has a supplement of 1 minute/100 km. Other passenger trains have a supplement of 40 seconds/100 km. Trains running on single track have a supplement of 3 minutes/100 km and trains running on multi-track have a supplement of 2 minutes/100 km. Additional running time supplements may be applied for specific infrastructure such as bridges or tunnels, or if the train is running on a track currently having work construction. Headway times depend on the infrastructure but are usually set to 3 minutes. Finally, the buffer time is recommended to be set to 60 seconds: two trains must be allowed to be up to 60 seconds late without disturbing the other when departing or arriving at a station.

2.4.2. Assessing timetable performances

From the passenger's point of view, a good timetable is a timetable with low running times, dwell times, transfer times, and high frequencies. However, IMs can have additional objectives. For example, Trafikverket aims for a more efficient and better timetabling process with a punctuality target of 95% of trains being less than 5 minutes late. To do so, additional performance indicators are needed. Goverde and Hansen (2013) give an overview of these performance indicators to evaluate a timetable's performance, as well as their interrelations. The infrastructure occupation is the share of time required to operate trains on a given railway infrastructure given a timetable pattern. It can be computed by the UIC 406 compression method provided by Landex et al. (2008). The timetable feasibility is defined as the ability of all trains to adhere to their scheduled train paths. It requires that all processes are realizable within their scheduled process times and that the scheduled train paths are conflict-free. Feasibility is achieved by following the green wave policy, as investigated by Corman et al. (2009). The green wave policy claims that if trains only face green signals during their run, their delays and energy consumption are reduced. However, it requires high coordination of train paths. The timetable stability is defined as the ability of the timetable to absorb initial delays (departure delays) and primary delays (variant in real-life operations that results in a longer process time than the scheduled time) so that delayed trains return to their scheduled train paths without rescheduling trains. The timetable robustness is defined as its ability to withstand design errors, parameter and operational variations. Hence, timetable robustness aims to minimize the occurrence of primary and secondary delays (delays propagation between trains). The timetable resilience is defined as its flexibility to prevent or reduce secondary delays using dispatching (re-timing, re-ordering, re-routing). Thus, it considers the interactions between the timetable and real-time traffic management. Finally, the energy efficiency of a timetable refers to the energy consumption of the trains, modeled based on the train's speed profiles. Usually, energy efficiency is seen as an optional performance and is investigated only when the first performances meet their requirements (Goverde and Scheepmaker (2023)). Keeping the same performance indicators, Warg (2016) assesses the quality of a railway timetable considering both capacity and socio-economic performances from a consumer point of view. However, this research is based on University of Birmingham (2013) which does not consider the timetable feasibility as a performance indicator. Instead, the timetable feasibility is considered as a fundamental condition: a timetable *has to* be feasible.

2.4.3. Microscopic, mesoscopic and macroscopic timetabling

The same railway corridor can be modeled at different levels: microscopic, mesoscopic, and macroscopic. Figure 2.2 shows those different levels. The macroscopic level considers the railway network at an abstract level, neglecting most of the infrastructure details. Tracks are depicted with edges and stations and junctions with nodes. The mesoscopic approach introduces a more detailed representation of nodes, the number of tracks per section, and an approximated model of signalling. Finally, the microscopic approach considers the railway network at a detailed level. Tracks are detailed with every signalling and interlocking system.

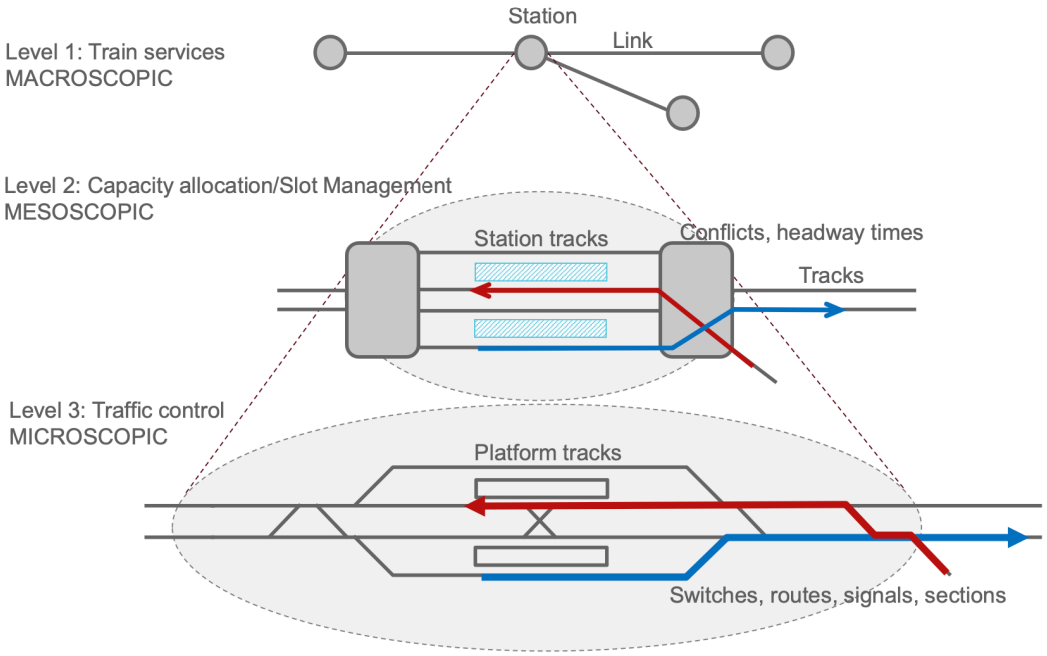


Figure 2.2: Different levels of infrastructure modeling (ProRail (2023))

Trains movement on a selected corridor are depicted on a time-distance diagram, as shown in Figure 2.3. A macroscopic diagram only represents the scheduled running times between different stations, whereas a microscopic diagram also depicts the blocking times of the different block sections.

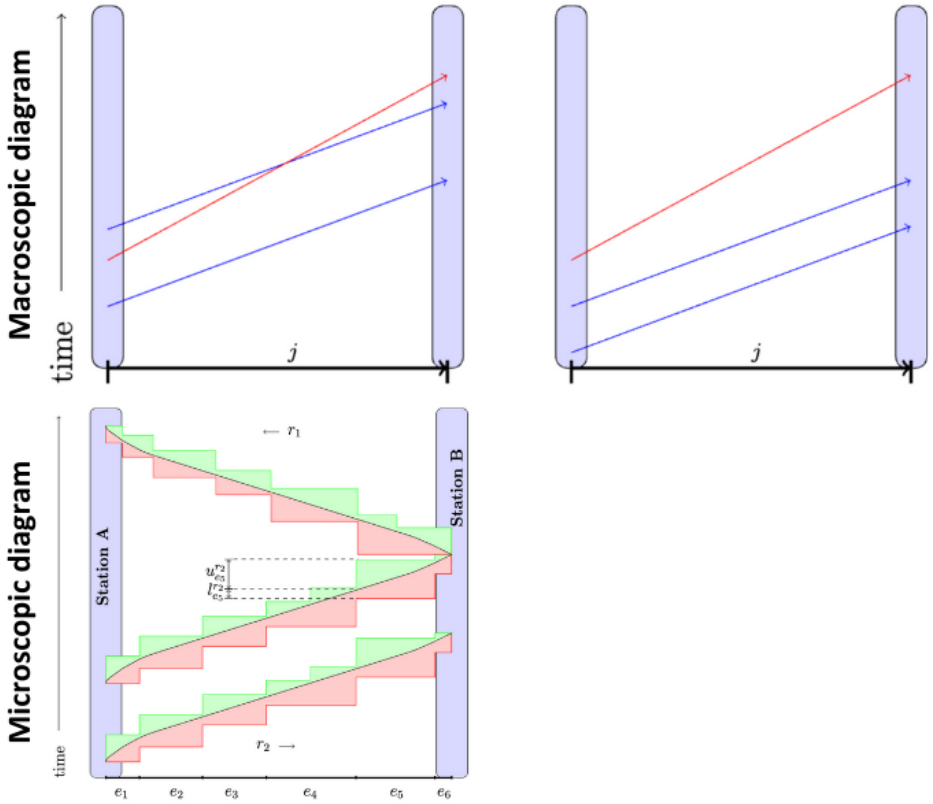


Figure 2.3: Time-distance diagrams

2.4.4. Macroscopic timetabling models

The Periodic Event Scheduling Problem (PESP) as introduced by Serafini and Ukovich (1989) constitutes a systematic approach used as the basis of most periodic activities scheduling problems. This model can be solved by means of a branch-and-bound algorithm. The PESP is based on a periodic event-activity directed graph $G = (E, A)$ in which the nodes $x \in E$ depict the arrival and departure events and the arcs $a = (x, y) \in A$ represent the activities between those events with interval constraints. Applied to railway operations, those activities can be running, dwelling, turnaround, passing-through, transfer, headways, or regularity. Figure 3.3 shows an example of an event-activity graph. One of the main strengths of this model is that the orders of the trains on the tracks do not need to be known *a priori*. However, this also makes the problem extremely complex in the case of a large network with many train requests. This model assumes that the lines and passenger connections are given, and the stations are considered with simple nodes. PESP is a feasibility problem: if a feasible solution does not exist, the user can relax one of the constraints and run the model again. However, it is possible to add an objective function (minimizing total running times for example). Then, the algorithm aims to find the optimal solution according to the objective function. If no feasible timetable is found, a possibility is to keep the timetable that satisfies the largest number of constraints. This method can easily be extended with different objective functions: minimization of passenger total travel time, maximization of robustness, etc.

The basis of the PESP model is the time window constraint, written as follows:

$$L_{x,y} \leq (v_y - v_x) \leq U_{x,y} \quad \forall a = (x, y) \in A$$

With A the set of all activities $a = (x, y)$ that have to be scheduled. $L_{x,y}$, and $U_{x,y}$ express the minimum and maximum duration of each activity. In a Π-periodic timetable, each activity has to be scheduled in such a way that it can be repeated. Thus, the equation becomes:

$$L_{x,y} \leq (v_y - v_x) \mod \Pi \leq U_{x,y} \quad \forall a = (x, y) \in A$$

Once linearized, this equation becomes:

$$L_{x,y} \leq (v_y - v_x) + p_{x,y} \cdot \Pi \leq U_{x,y} \quad \forall a = (x, y) \in A$$

With $p_{x,y}$ an integer decision variable that can be limited to a binary variable if $U_{x,y} < \Pi \quad \forall a = (x, y) \in A$.

The literature shows that the PESP approach is broadly used for railway timetabling problems. Odijk (1996) applies the PESP model to construct periodic railway timetables and develops a new algorithm using a constraint generation approach. However, it took more than 10 years to see the first application of the PESP to real-life railway operations. Indeed, decision support by operations research methods in railway companies was usually limited to operations planning. Liebchen and Möhring (2008) apply the PESP to the Berlin subway. The results have shown that the new timetables lead to shorter passenger waiting times and turnaround times, resulting in a reduction in the number of rolling stocks required. To increase their number of passengers and the robustness of their timetable, the Dutch Railways started to apply the PESP to model their timetables, resulting in an increase in the number of passengers, profits, reliability, and intermodality, as proven by Kroon et al. (2008).

The literature shows that the PESP approach has many extensions. Liebchen (2004) introduces symmetry to the PESP. It has been shown that symmetry leads to sub-optimality when minimizing passenger waiting times: the feasible solution is usually worse than the feasible solution without symmetry. However, introducing symmetry decreases the computer running times. To speed up the running time, Herrigel et al. (2018) formulates the PESP as a MILP (Mixed-Integer Linear Programming) with a sequence of smaller sub-problems that can be parameterized to reach a trade-off between simultaneous or sequential planning. This considerably accelerates solving times compared to PESP using the standard MILP formulation. Polinder et al. (2020) formulates the PESP as a MIQP (Mixed-Integer Quadratic Programming) for strategic Passenger-Oriented Timetabling to an approach aiming at decision-making in the strategic phase of planning, and to determine an outline of a timetable that is good from the passengers' perspective. Bešinović et al. (2016) suggests that macroscopic models cannot be used

as such and therefore need to be integrated with microscopic models to ensure feasibility on a microscopic level. Indeed, a timetable can be feasible on a macroscopic level but, to comply with the train separation principle, the feasibility of the designed timetable must be tested at the microscopic level.

2.5. Railway line planning

From the expected long-term passenger flows, line planners identify bottleneck sections of the corridor and investigate possible solutions. Such solutions could consist in increasing the rolling stock's capacity (more seats, more cars), increasing the capacity of a corridor by upgrading its signalling, or building additional tracks. The construction costs of a new track are difficult to estimate. The cost of a new line depends on the geography of the landscape it crosses and the need to build or not specific infrastructures such as bridges or tunnels. Trafikverket and Jacobs (2021) carries out an estimation of the overall costs of the New Main Lines railways. This project proposes the construction of new high-speed railways between Stockholm-Gothenburg and Stockholm-Malmö (Trafikverket (2023a)). The estimations are made by collecting, analyzing and comparing the construction costs of several European high-speed rail projects. A construction cost per kilometer benchmark is presented for high-speed rail, which excludes the planning and design costs. The report reveals that the average and median construction costs across the benchmark range are respectively SEK 415 and SEK 373 million per km. This study also estimates the costs of different infrastructures such as bridges, tunnels, or stations. The stations are categorized into three categories reflecting on the complexity of the station. The construction costs are estimated to SEK 206,500 per platform meter for minimal stations. These cost estimations are used for the rest of the research.

2.6. Conclusion: research gap

The literature review shows that in long-term strategic planning, the focus is usually given to upgrading the infrastructure by identifying the congested infrastructure using timetabling tools. At the short-term tactical level, timetabling tools are fundamental to optimize the use of existing infrastructure. However, the simulation models cannot always find a timetable satisfying every constraint (demand, infrastructure, norms), which requires relaxing some constraints. Usually, in case of insufficient capacity, the demand constraint is relaxed by canceling, re-ordering, re-routing trains, or reducing their frequency. This approach is suitable for short-term tactical and operational levels, where changes in the infrastructure are not possible. However, this is not appropriate for the long-term strategic phase, where the planners can modify the infrastructure and evaluate different scenarios. Thus, instead of only focusing on minimizing passenger travel time and maximizing demand satisfaction with strict respect for the infrastructure constraint, this research investigates the opposite approach. It examines the potential for adapting infrastructure to meet long-term demand, suggesting optimal infrastructure improvements if the predicted demand exceeds the current capacity of the infrastructure. Timetables are first simulated at the macroscopic level, before being tested at the microscopic level to detect the possible overlap of different blocking times. The application of the PESP to railway operations has already demonstrated its strength and now serves as a foundational model for macroscopic timetabling, integrating various constraints like train running times, dwell times, and minimum headway times. This model can be extended and formulated as a MILP, by including an objective function and additional constraints. If current MILP formulations of the PESP model can already include conflict detection to ensure a conflict-free timetable, no model including flexible numbers of tracks and a minimization of the construction costs has been formulated yet. This research gap is fundamental for our approach: since the model aims to investigate every possible infrastructure upgrade to minimize the construction costs, the infrastructure should be considered as flexible and the possible conflicts should be adapted to the number of tracks.

3

Model

3.1. Methodology

A model is designed to answer the research question. Firstly, the objectives and requirements of the model are clearly defined in compliance with the scope of study and research question previously introduced. The model aims to minimize the new infrastructure costs while ensuring the feasibility of the long-term demand. The requirements of the model are defined by reviewing the literature, brainstorming, and interviewing different project members. These requirements are regrouped in Table 3.1. A non-functional constraint or objective only affects the quality or usability of the model, but does not directly influence the core functional performance.

	Functional	**Non-functional**
Constraints	*Must do* Give a feasible timetable at macro level Adapted to the Swedish market Satisfy the demand Satisfy the traffic regulations	*Must be* Adapted to the Swedish regulations
Objectives	*Should do* Check if the demand can be satisfied If not, upgrade the infrastructure Minimize the construction costs Minimize the passenger times	*Should be* Fast to solve Flexible User-friendly

Table 3.1: Requirements of the model

Based on the constraints and objectives it has to follow, a PESP model is formulated as a MILP, by reviewing existing models from the literature and their limits. Each constraint of the PESP-MILP model is first written in a non-mathematical way, with a simple and clear formulation. Some of those constraints are divided into multiple sub-constraints if they are too broad, or grouped in a bigger constraint if they are too precise. Once the constraints of the PESP-MILP are formulated in a non-mathematical way, the list of sets, parameters, and decision variables needed for the mathematical formulation is defined. Each constraint is then written mathematically. They are first written as basic equations and then elaborated as more complex equations. The mathematical formulation has to ensure that the constraint is satisfied in every different scenario that might occur. Most of the time, a help variable is needed. A help variable is a decision variable that is used when the constraint cannot be written using only the main decision variables. Those decision variables are often binary variables depending on the main decision variables.

Before being able to implement the optimization model, processing functions have to be developed, to convert the raw input data into data usable by the optimization model. Moreover, a visualization

function is also developed so that the timetable given as the output of the PESP-MILP model can be visualized in a time-distance diagram so that the constraints satisfaction can be quickly checked.

The implementation of the mathematical formulation is a complex part. The PESP-MILP model is implemented with the Gurobi Optimizer, a solver broadly used by railway companies such as DB or SNCF (Gurobi (n.d.-a)). It permits solving complex optimization problems and quickly testing and modifying them (Gurobi (n.d.-b)). The verification aims to test the model to ensure that it works the way it is intended to. To do so, the constraints are implemented one by one, starting with the basic constraints. Every time a new constraint is implemented, the model is tested with different basic networks and for different train requests, for which the results are known in advance. If the output of the model is not what was expected, i.e. if some constraints or the objective function are not satisfied, then the code is modified until each constraint is respected. Logical reasoning, visualization of the inconsistent timetable, or expert consultation are different means to find the code error. When and only when every implemented constraint is perfectly satisfied, a new constraint can be implemented. Every constraint has to be written as a MILP, using tips such as the big-M or epsilon constraints (Williams (2013)).

The following example shows the process used to obtain the linear formulation.

$$x > k \implies b = 1$$
$$x \leq k \implies b = 0$$

Where
$x \in \mathbb{N}$ an integer decision variable
$k \in \mathbb{R}$ a constant
$b \in \{0, 1\}$ a binary decision variable indicating whether x is greater than or equal to k

The equations above are not linear and can therefore not be implemented as such. Thus, two new constraints and two new parameters are introduced as follows:

$$k \leq x - \epsilon + M(1 - b)$$
$$k \geq x - Mb$$

Where
$M \in \mathbb{R}$ a sufficiently large constant
$\epsilon \in \mathbb{R}^{+*}$ a sufficiently small positive value

If the values of M and ϵ are properly calibrated, then the model forces b to take either the value 0 or 1, depending on the condition on k that x has to satisfy.

if $b = 1$ then the equations become:

$$k \leq x - \epsilon$$
$$k \geq x - M$$

Since M is a very large value, the second constraint is always satisfied, which means that only the first constraint needs to be satisfied by the model

if $b = 0$ then the equations become:

$$k \leq x - \epsilon + M$$
$$k \geq x$$

Since M is a very large value, the first constraint is always satisfied, which means that only the second constraint needs to be satisfied by the model

Many hypotheses are made all along the design process, to simplify the model by omitting possible situations that are not relevant to the research. However, each assumption has to be justified and their expected consequence on the results carefully considered. The hypotheses made during this research are presented in Section 3.6 and discussed in Section 5.7.

3.2. Model architecture

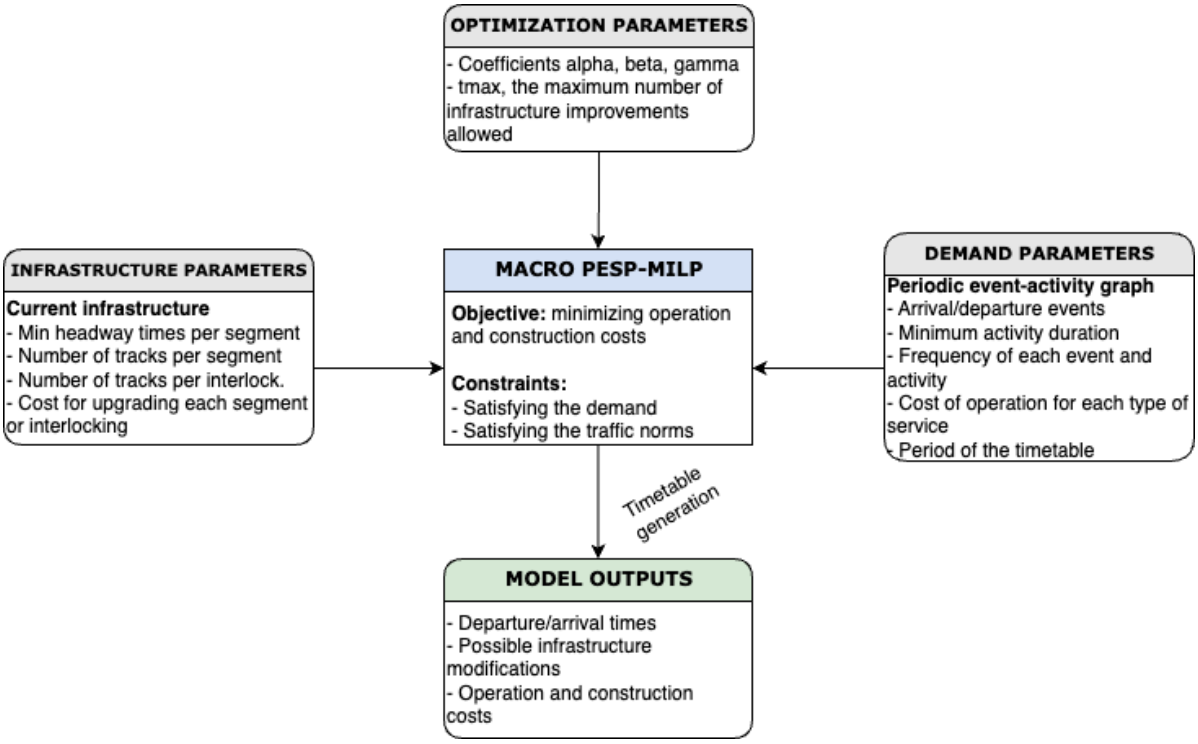


Figure 3.1: Architecture of the model

The developed model aims to schedule a list of train requests and their characteristics while minimizing the costs of the tracks required to meet the demand. Figure 3.1 presents the inputs and outputs of the model. The total number of tracks allowed to be constructed $t^{\max}$ is taken as a parameter and restricts the tracks to be constructed, but the model is free to propose or not any infrastructure improvements. An infrastructure upgrade is defined as the construction of *one* additional track at *one* interlocking or *one* segment. For example, the construction of two tracks in the same station or on the same segment is considered as two infrastructure improvements. Once the optimal solution (if it exists) is found, the macroscopic timetable and associated infrastructure improvements are given as outputs of the PESP-MILP model.

3.3. Model formulation

3.3.1. Graphs, sets and indices

Graphs, sets and indices		
$\mathcal{N} = (I, S)$	Un-directed network graph N with I set of nodes and S set of arcs	
I	Set of interlocking areas (station, junction)	$i \in I$
S	Set of segments between interlocking areas	$s \in S$
$\mathcal{G} = (E, A)$	Periodic event-activity graph G with E set of nodes and A set of arcs	
E	Set of events (arrival, departure, arr-pass, dep-pass)	$x \in E$
$A = \{(x, y) : x, y \in E, x \neq y\}$	Set of activities between events x and y (run, dwell, pass, transfer, headway, turnaround, reg)	$a = (x, y) \in A$
$A_i \subset A$	Set of activities taking place at interlocking i	$a = (x, y) \in A_i$
$A_s \subset A$	Set of activities taking place at segment s	$a = (x, y) \in A_s$
$C_s^k = \{\{a, a', ..., a^k\} : a', a', ..., a^k \in A_s$ and $a^i \neq a^j$ for $i \neq j\}$	Set of every possible combination of k activities taking place at segment s	$c = \{a, a', ..., a^k\} \in C_s^k$
$C_i^k = \{\{a, a', ..., a^k\} : a', a', ..., a^k \in A_i$ and $a^i \neq a^j$ for $i \neq j\}$	Set of every possible combination of k activities taking place at interlocking i	$c = \{a, a', ..., a^k\} \in C_i^k$

Table 3.2: Graphs, sets and indices of the mathematical model

Table 3.2 shows the two graphs, their sets, and indices used for the mathematical formulation. The first graph is the network graph, depicted at the macroscopic level as an un-directed network graph $\mathcal{N} = (I, S)$ with I set of nodes and S set of arcs. Each node i represents an interlocking area (station or junction), has a unique ID and a number of tracks. Each arc s represents a segment between two interlocking areas, and has a unique ID, a number of tracks, a length, and a minimum headway. Figure 3.2 gives an example of an un-directed network graph with four interlockings and three segments.

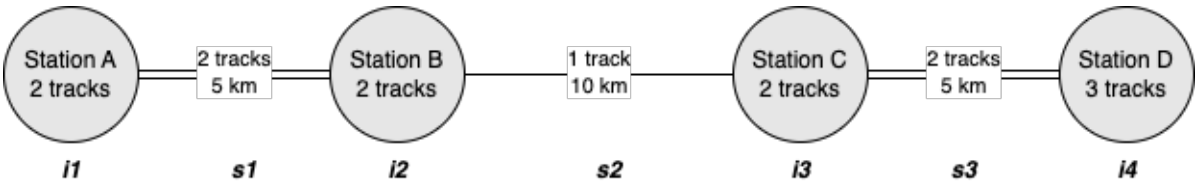


Figure 3.2: Un-directed railway network graph

The events and activities are represented as a second graph, called periodic-event activity graph $\mathcal{G} = (E, A)$, with E set of nodes and A set of arcs. Each node x represents an event and each arc $a = (x, y)$ represents an activity. The set of activities A is the intersection of all subsets representing each possible activity. In the same way, A is also the intersection of all subsets representing each possible location:

$$A = A_{\text{run}} \cap A_{\text{dwell}} \cap A_{\text{pass}} \cap A_{\text{reg}} \cap A_{\text{headway}} \cap A_{\text{turnaround}} \cap A_{\text{transfer}}$$

$$A = (\cap_{i \in I} A_i) \cap (\cap_{s \in S} A_s)$$

Figure 3.3 shows an example of an event-activity graph. Each arc has a periodic interval. For the headway, the arcs represent both possible orders. Indeed, before the optimization of the model, neither the time of each event nor the order of the train is known: one or the other train can be chosen to run first. The regularity arcs indicate that the pointed event has to happen an exact time after the origin event. The turnaround and transfer arcs can be interpreted in the same way: the pointed event has to be scheduled after the origin event with a minimum time difference. The number X indicated after the line name indicates the copy of the line. For instance, if the line HS1 runs three times an hour, then three train lines are created: HS1-1, HS1-2 and HS1-3.

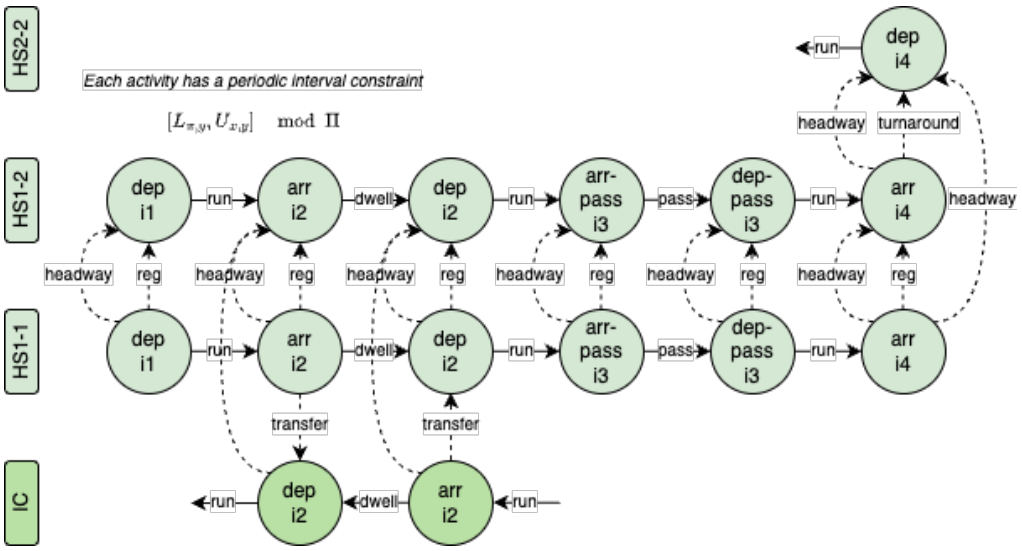


Figure 3.3: Periodic event-activity graph applied to railway operations

Unlike other PESP approaches, this model introduces a flexible conflict restriction. The conflicts first need to be detected. To do so, different sets C_i^k and C_s^k of all combinations of k activities taking place at interlocking i or segment s are introduced. A combination refers to a selection of activities a, from a larger set A_i or A_s, where the order of the items does not matter. Indeed, if activity a is conflicted with activity a', then activity a' is also conflicted with activity a: the combination (a, a') is equivalent to the combination (a', a). The combinations of only one activity are not considered since a conflict involves by definition at least two different activities. Figure 3.4 shows an example of a set of activities A_i and the corresponding set of combination C_i^2, C_i^3, C_i^4.

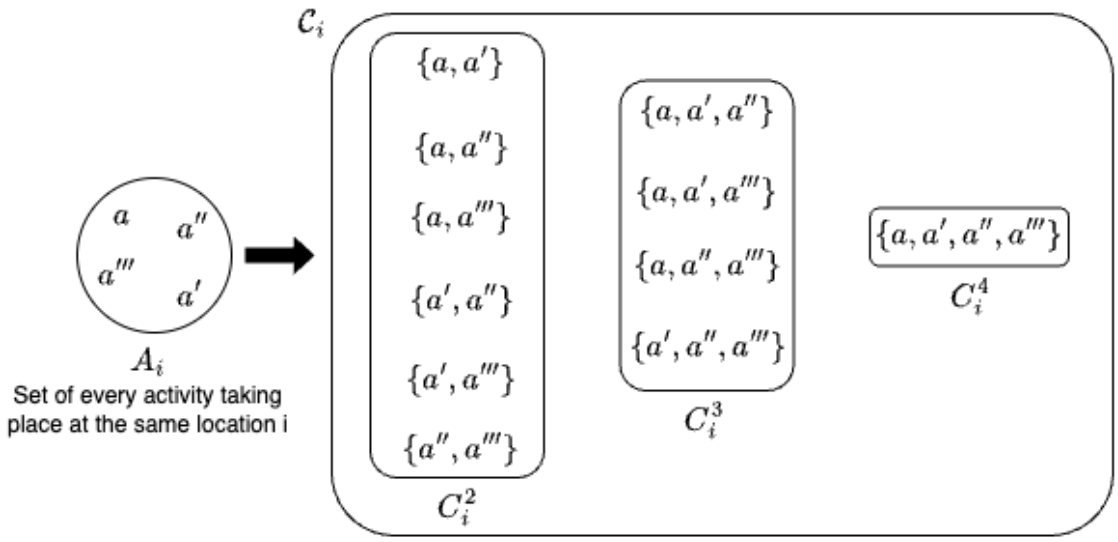


Figure 3.4: Set of activities and associated sets of combinations

Given a set of activities A_i such that $|A_i| = n$, the number of possible combinations of k activities (where $k \leq n$) is given by the following formula:

$$|C_i^k| = \binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{3.1}$$

In the example given in Figure 3.4, we have

$$|C_i^2| = \binom{4}{2} = \frac{4!}{2!(4-2)!} = 6$$

$$|C_i^3| = \binom{4}{3} = \frac{4!}{3!(4-3)!} = 4$$

$$|C_i^4| = \binom{4}{4} = \frac{4!}{4!(4-4)!} = 1$$

Figure 3.4 and Equation 3.1 demonstrate the computational burden of these different sets. For a set of only four activities taking place at one given location i, eleven possible combinations of activities are possible in total. On a large network with multiple interlockings and segments, this number is multiplied by the number of locations. Moreover, for every additional train that has to be scheduled, one new activity a is added and the size of A_i is increased by one. Therefore, $\binom{n}{k-1}$ new possible combinations of conflicts are added, according to Pascal's rule. Adding constraints for every subset of nodes is computationally quite demanding, but needed to properly detect and restrict every possible conflicts between different trains.

3.3.2. Parameters

The parameters taken as input of the model are grouped into different categories: the infrastructure parameters, the demand parameters, and the optimization parameters, as presented in Table 3.3. The infrastructure parameters describe the network and its characteristics before the optimization process. The demand parameters aim to outline the different train lines the timetable has to satisfy and their characteristics. The optimization parameters are the big-M constraint values, used for the linearization of the model, the different operation and construction costs expressed in Swedish Crown (SEK), and their coefficients. These values are fixed by the user and can be adjusted regarding the priority of the experiment. By adjusting the values of the coefficients, the users can choose to focus on minimizing

the operation or construction costs. By adjusting the parameters c_a, the user can prioritize the minimization of some activities (for example, it is more important to have small running times than small turning-around times).

	Infrastructure parameters	
t_s^{before}	Number of tracks of segment s	[unit]
t_i^{before}	Number of tracks of interlocking i	[unit]
$L_{x,y} > 0$	Minimum duration time of activity $a = (x, y)$	[s]
	Demand parameters	
Π	Periodicity of the timetable	[s]
$fre_{x,y}$	Frequency of the line in which activity $a = (x, y)$ is part of	[unit/period]
$dir_{x,y}$	Direction of the train if activity $a = (x, y)$ implies a train running on a segment or passing through an interlocking	[dimension less]
	Optimization parameters	
M	Sufficiently large value	[dimension less]
ϵ	Sufficiently small positive value	[dimension less]
α	Coefficient for operation costs	[dimension less]
β	Coefficient for segment construction costs	[dimension less]
γ	Coefficient for interlocking construction costs	[dimension less]
$t^{\max}$	Maximal number of new tracks allowed to be constructed on the entire network	[unit]
$c_{x,y}$	Cost for operating activity $a = (x, y)$, based on the type of activity and the weight of the line	[SEK/sec]
$c_s^{\text{new track}}$	Cost for building an additional track on segment s	[million SEK]
$c_i^{\text{new track}}$	Cost for building an additional track on interlocking i	[million SEK]

Table 3.3: Parameters of the mathematical model

3.3.3. Decision variables

Decision variables		
$v_x \in \{0, 1, \dots, \Pi - 1\}$	Time at which event x takes place	[s]
$d_{x,y} \in \{0, 1, \dots, \Pi\}$	Duration of activity $a = (x, y)$	[s]
$p_{x,y} \in \{0, 1\}$	Binary variable indicating whether activity $a = (x, y)$ is such that $v_y < v_x$	[dimension less]
$q_{c,i} \in \{0, 1\}$	Binary variable indicating whether each activity of the combination $c = (a, a', \dots)$ is conflicted with every other one at interlocking i	[dimension less]
$q_{c,s} \in \{0, 1\}$	Binary variable indicating whether each activity of the combination $c = (a, a', \dots)$ is conflicted with every other one at segment s	[dimension less]
$k_s^{\text{opp}} \in \{0, 1\}$	Binary variable indicating whether at least one conflict implying two trains running in opposite directions is detected at segment s	[dimension less]
$k_s^{\text{same}} \in \{0, 1\}$	Binary variable indicating whether at least one conflict implying two trains running in the same direction is detected at segment s	[dimension less]
$t_i^{\text{needed}} \in \mathbb{N}$	Minimal number of tracks need on interlocking i after optimization	[unit]
$t_s^{\text{needed}} \in \{1, 2, 4\}$	Minimal number of tracks needed on segment s after optimization	[unit]
$t_i^{\text{after}} \in \mathbb{N}$	Number of tracks of interlocking i after optimization	[unit]
$t_s^{\text{after}} \in \{1, 2, 4\}$	Number of tracks of segment s after optimization	[unit]

Table 3.4: Decision variables of the mathematical model

Table 3.4 presents the decision variables that the model must optimize. v_x and $d_{x,y}$ are the main decision variables, corresponding to the time at which each event x takes place and the duration of the activity $a = (x, y)$. $p_{x,y}$ is an help variable variable, as defined in Section 2.4.4. $q_{c,s}$ and $q_{c,i}$ are binary variables indicating whether each activity a from the combination $c = (a, a', \dots)$ is conflicted with every other ones. Based on these binary variables, the decision variables t_s^{needed} and t_i^{needed} represent the minimum number of tracks required to resolve detected conflicts, ensuring that the demand fit within the infrastructure without causing any collision. Finally, based on these decision variables and the initial state of the infrastructure, t_s^{after} and t_i^{after} return the number of tracks the network should have to satisfy the demand. The formulation of these variables and the associated calculations are explained in Section 3.3.5 and Figure 3.5 explains the use of each variable and their interactions. The interactions between the different variables are depicted with double arrows since the dependencies between the variables go both ways. The following Figure can be read both ways: the number of tracks of the network restricts the possible conflicts happening (from the top to the bottom) *and* the conflicts detected on the network define the required number of tracks on each portion of the network (from the bottom to the top).

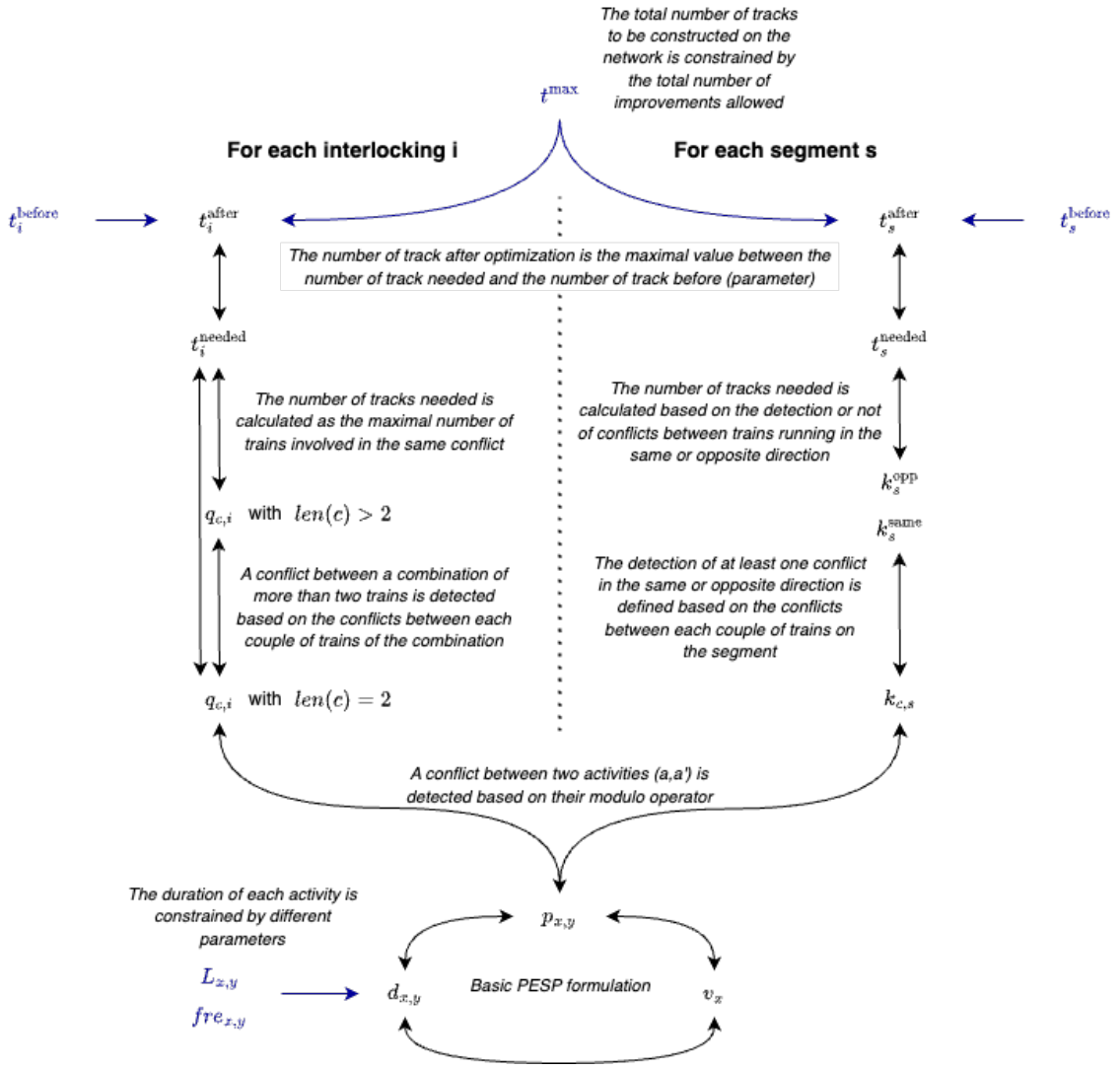


Figure 3.5: Interactions between the model's variables *(in black)* and different parameters *(in blue)*

3.3.4. Objective function

The objective function of the PESP-MILP model aims to minimize the track construction costs while maximizing their benefits on the traffic. It is expressed as follows:

$$\text{minimize} \quad \alpha \cdot c_{\text{operation}}^{\text{normalized}} + \beta \cdot c_{\text{segment}}^{\text{normalized}} + \gamma \cdot c_{\text{interlocking}}^{\text{normalized}} \tag{3.2}$$

With:

$$\alpha + \beta + \gamma = 1 \tag{3.3}$$

$$c^{\text{normalized}} = \frac{c - c^{\min}}{c^{\max} - c^{\min}} \tag{3.4}$$

The objective function 3.2 is written as a minimization of total costs by minimizing the sum of three different components: operation costs, construction costs of segments, and construction costs of interlockings. The construction cost of one track is approximately several billion SEK, while the operational cost for one timetable is around hundreds of thousands of SEK. The overall costs can therefore not be minimized as such, otherwise, the model would focus too much on reducing the construction costs.

Therefore, the costs are normalized using the min-max normalization 3.4. Additionally, the different coefficients α, β, and γ reflect on the priority to give to the optimization.

$$c_{\text{operation}} = \sum_{a=(x,y)\in A} c_{x,y} \cdot d_{x,y} \tag{3.5}$$

$$c^{\min}_{\text{operation}} = \sum_{a=(x,y)\in A} c_{x,y} \cdot L_{x,y} \tag{3.6}$$

$$c^{\max}_{\text{operation}} = \sum_{a=(x,y)\in A} c_{x,y} \cdot \Pi \tag{3.7}$$

The total operation costs are computed as the sum of every activity duration multiplied by its cost. The minimum possible operation costs correspond to the situation where every activity is scheduled accordingly to their minimum duration time $L_{x,y}$. The maximum possible operation costs correspond to the situation where every activity duration is maximal, i.e. if every activity duration equals Π.

$$c_{\text{interlocking}} = \sum_{i\in I} c_i^{\text{new track}} \cdot (t_i^{\text{after}} - t_i^{\text{before}}) \tag{3.8}$$

$$c^{\max}_{\text{interlocking}} = t^{\max} \cdot \max_{i\in I}(c_i^{\text{new track}}) \tag{3.9}$$

$$c^{\min}_{\text{interlocking}} = 0 \tag{3.10}$$

The total construction costs of interlocking are computed as the sum of all tracks that need to be built on every interlocking. The minimum possible cost is zero and corresponds to the situation where no interlocking is upgraded. The maximum possible cost corresponds to the situation where every new track $t^{\max}$ is constructed at the most expensive interlocking to upgrade.

$$c_{\text{segment}} = \sum_{s\in S} c_s^{\text{new track}} \cdot (t_s^{\text{after}} - t_s^{\text{before}}) \tag{3.11}$$

$$c^{\max}_{\text{segment}} = t^{\max} \cdot \max_{s\in S}(c_s^{\text{new track}}) \tag{3.12}$$

$$c^{\min}_{\text{segment}} = 0 \tag{3.13}$$

The total construction costs of segments are computed as the sum of all tracks that need to be built on every segment. The minimum possible cost is zero and corresponds to the situation where no segment is upgraded. The maximum possible cost corresponds to the situation where every new track $t^{\max}$ is constructed at the most expensive segment to upgrade. This maximal value is approximated because, in reality, the segment can only have 1, 2 or 4 tracks. Thus, if $t^{\max}$ is larger than the number of tracks that can be constructed on a segment, the maximum possible segment construction cost is slightly overestimated.

3.3.5. Constraints

The objective function 3.2 is subject to the following constraints.

$$d_{x,y} = (v_y - v_x) + p_{x,y} \cdot \Pi \qquad \forall a = (x,y) \in A \tag{3.14}$$

Constraint 3.14 defines the duration of activity $a = (x, y)$ as the time difference between the end time v_y and the start time v_x. If the activity is spread over two periods, i.e. if $p_{x,y} = 1$, then Π is added to the time difference $(v_y - v_x)$ to ensure that the duration is always a positive value, as presented in Figure 3.6.

$$p_{x,y} = \begin{cases} 1 & \text{if } v_y < v_x \\ 0 & \text{otherwise} \end{cases} \qquad \forall a = (x, y) \in A \tag{3.15}$$

Constraints 3.15 defines the value of the modulo operator $p_{x,y}$. In this model, it is restricted to a binary variable since the upper bound of every activity is lower than the period Π. It takes the value 1 if event y is scheduled before event x, and 0 otherwise. In other words, the modulo operator indicates whether the activity is spread over two periods, i.e. it indicates whether the activity points at an event in the same or next period than the origin event.

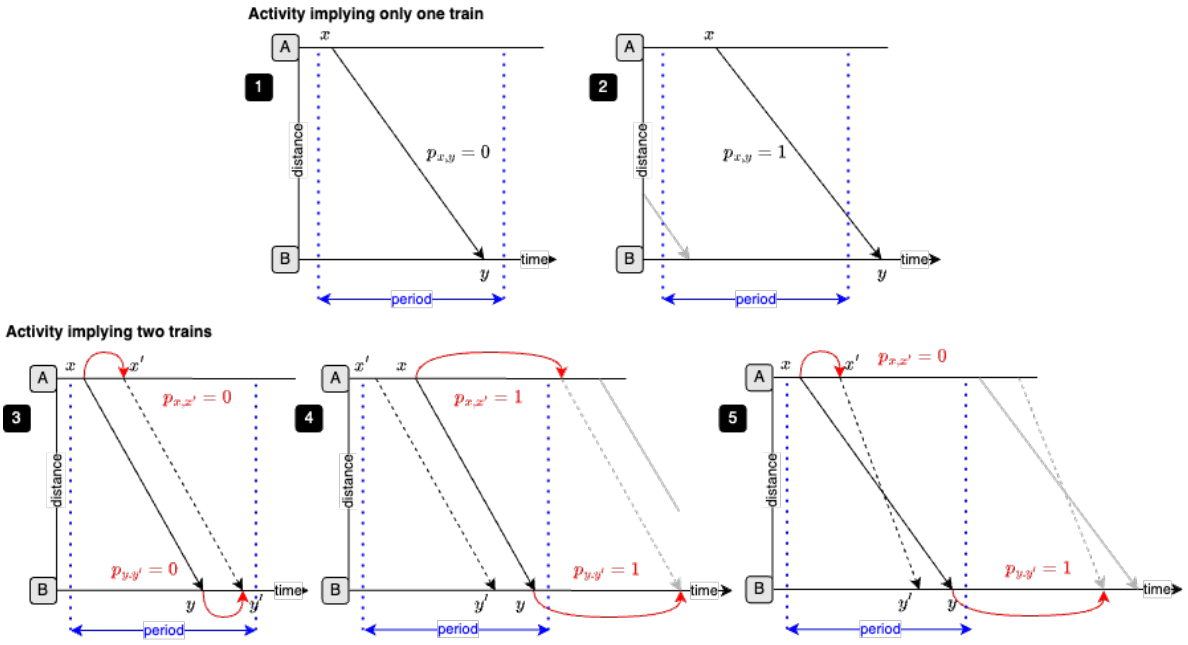


Figure 3.6: Definition of the modulo operator $p_{x,y}$

$$L_{x,y} \leq d_{x,y} \leq \Pi \qquad \forall a = (x, y) \in A_{\text{run}} \cup A_{\text{dwell}} \cup A_{\text{pass}} \cup A_{\text{turnaround}} \cup A_{\text{transfer}} \tag{3.16}$$

$$L_{x,y} \leq d_{x,y} \leq \Pi - L_{x,y} \qquad \forall a = (x, y) \in A_{\text{headway}} \tag{3.17}$$

$$d_{x,y} = \frac{\Pi}{fre_{x,y}}, \text{ with } \frac{\Pi}{fre_{x,y}} \in \mathbb{N}^* \qquad \forall a = (x, y) \in A_{\text{reg}} \tag{3.18}$$

Constraints 3.16 ensure that the duration of all running, dwelling, passing-through, transferring and turn-around activities is bunded by their minimum duration time $L_{x,y}$ and the period Π, because these activities do not have specific upper bounds needed. Constraint 3.17 ensure that the headway activities a are also bounded with a maximal duration. In this macroscopic model, the headway times are assumed to be only depending on the segment, and not on the direction, which gives $L_{x,y} = L_{y,x} \quad \forall a = (x, y) \in A_{\text{headway}}$. Finally, constraint 3.18 ensures that the duration of the regularity activity $a = (x, y)$ is fixed to the period of the corresponding line. This ensures that all events on the line occur at consistent, evenly-spaced intervals.

$$\sum_{i \in I} (t_i^{\text{after}} - t_i^{\text{before}}) + \sum_{s \in S} (t_s^{\text{after}} - t_s^{\text{before}}) \leq t^{\max} \tag{3.19}$$

Constraint 3.19 restricts the number of new tracks to be constructed on the entire network by ensuring that there are no more upgrades than $t^{\max}$. At each location, the number of new tracks is calculated as the difference between the number of tracks after and the number of tracks before the optimization process.

$$t_s^{\text{after}} = \max\{t_s^{\text{before}}, t_s^{\text{needed}}\} \qquad \forall s \in S \tag{3.20}$$

Constraint 3.20 ensures that if the initial number of tracks at a segment is lower than the minimal required number of tracks, the infrastructure is updated.

$$t_s^{\text{needed}} = \begin{cases} 1 & \text{if } k_s^{\text{opp}} = 0 \text{ and } k_s^{\text{same}} = 0 \\ 2 & \text{if } k_s^{\text{opp}} = 1 \text{ and } k_s^{\text{same}} = 0 \\ 4 & \text{if } k_s^{\text{same}} = 1 \end{cases} \qquad \forall s \in S \tag{3.21}$$

Constraint 3.21 defines the minimum number of tracks needed at segment s to ensure that the timetable is conflict-free, based on the number of conflicts detected. Each segment can have one, two, or four tracks, as presented in Figure 3.7. On a single-track corridor, trains can run in both directions. Therefore, a segment needs at minimum one track if no trains are running in opposite directions at the same moment and no trains are overtaking each other. On a double-track corridor, each track is reserved for one direction. Therefore, a segment needs at minimum two tracks if at least two trains are running in opposite directions but no train is overtaking another one. Finally, on a four-track corridor, we consider that two tracks are used for each direction, which allows overtaking. A segment needs at minimum four tracks if at least one train is overtaking another one running in the same direction.

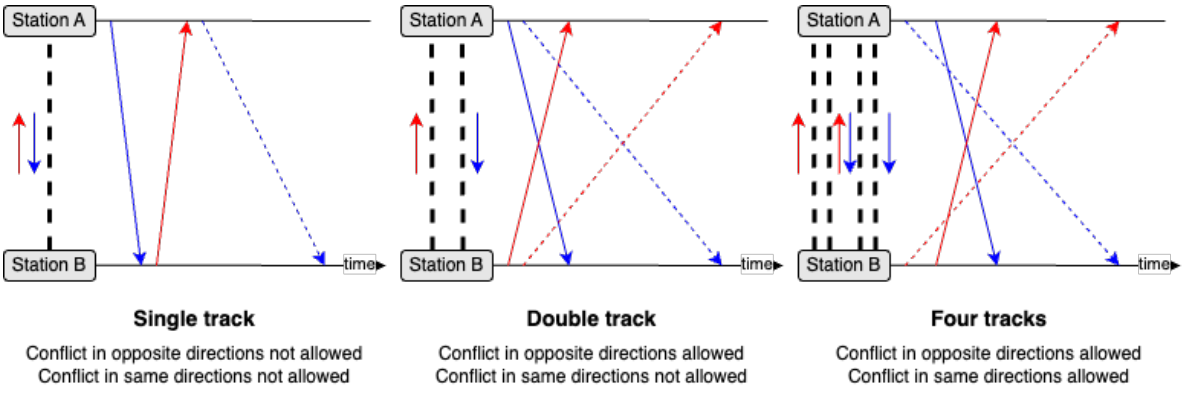


Figure 3.7: Relation between the type of conflicts allowed and the number of tracks needed per segment

$$k_s^{\text{opp}} = \begin{cases} 1 & \text{if } \sum_{\substack{c=(a,a') \in c_s^2 \\ dir_a \neq dir_{a'}}} k_{c,s} \geq 1 \\ 0 & \text{otherwise} \end{cases} \qquad \forall s \in S \tag{3.22}$$

$$k_s^{\text{same}} = \begin{cases} 1 & \text{if } \sum_{\substack{c=(a,a') \in c_s^2 \\ dir_a = dir_{a'}}} k_{c,s} \geq 1 \\ 0 & \text{otherwise} \end{cases} \qquad \forall s \in S \tag{3.23}$$

Constraints 3.22 and 3.23 define the help binary variables k which take the value 1 if there is respectively at least one overtaking (i.e. conflict between two trains running in the same direction) and at least one collision (i.e. conflict between two trains running in opposite directions) on the segment s.

$$q_{c,s} = \begin{cases} 1 & \text{if } (p_{x,y} + p_{x',y'} \\ & +p_{x,x'} + p_{y,y'}) \equiv 0 \pmod 2 \\ 0 & \text{otherwise} \end{cases} \qquad \forall c = \{(x,y),(x',y')\} \in C_s^2, \forall s \in S \tag{3.24}$$

Constraints 3.24 defines the binary variable q which takes the value 1 if the activities $a = (x,y)$ and $a' = (x',y')$ from the combination $c = (a,a')$ are conflicted at interlocking s. Zhang and Nie (2016) proposes linear constraints to avoid or allow overtaking based on the PESP problem. The research describes all possible scenarios and analyses the behavior of the different modulo operators $p_{x,y}$. When two trains $a = (x,y)$ and $a' = (x',y')$ run on the same segment, they are linked with two headway activities at their origin and destination. Thus, four modulo operators can be obtained: $p_{x,y}$, $p_{x',y'}$, $p_{x,x'}$ and $p_{y,y'}$. The authors show that the sum of those modulo operators equals zero, two, or four when overtaking is prevented, and equals to one or three otherwise. In other words, if the sum is even, no conflict is detected and if the sum is odd, a conflict is detected. Figures 3.9 and 3.8 depict all possible overtaking scenarios, and the associated Table 3.5 details the values of p in each scenario. However, the conflict detection developed for our research presents two differences from the one proposed by Zhang and Nie (2016). Firstly, our research also detects the collision, i.e. the conflict between two trains running in opposite directions. To do so, the model simply inverses the direction of the train and applies the same constraint previously introduced. Secondly, in our model, the conflicts are only detected and counted per segment, and the model is free to create or not conflict. Only the total number of new tracks is restricted, as shown in Figure 3.5.

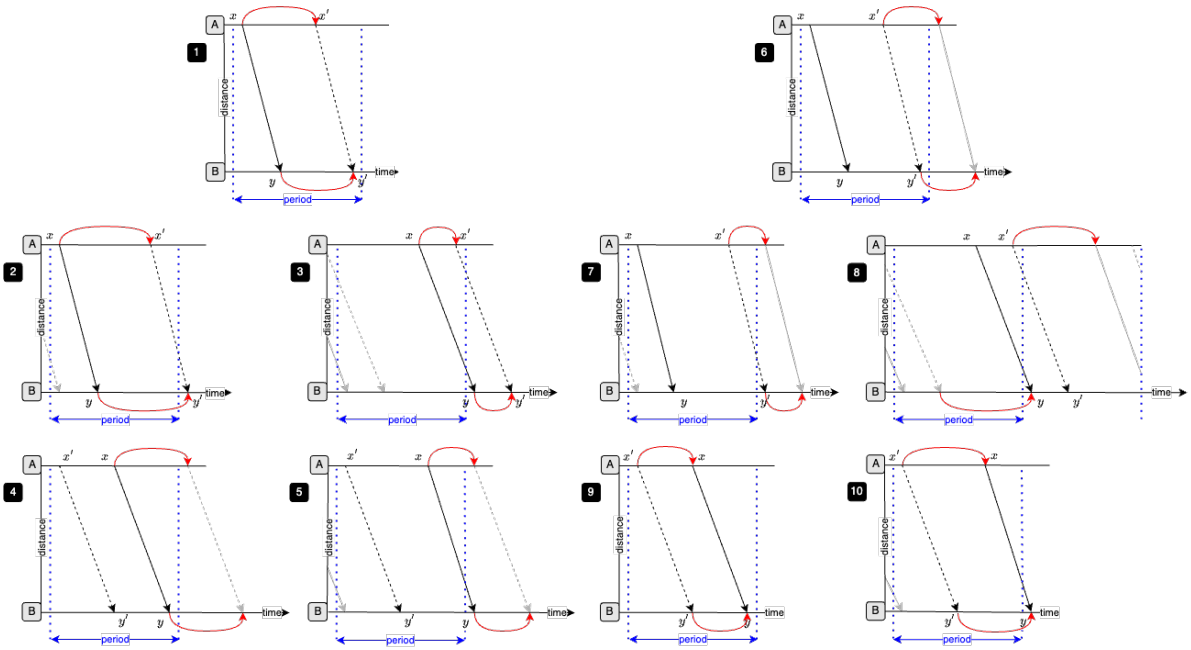


Figure 3.8: All possible situations where an overtaking is avoided, based on Zhang and Nie (2016)

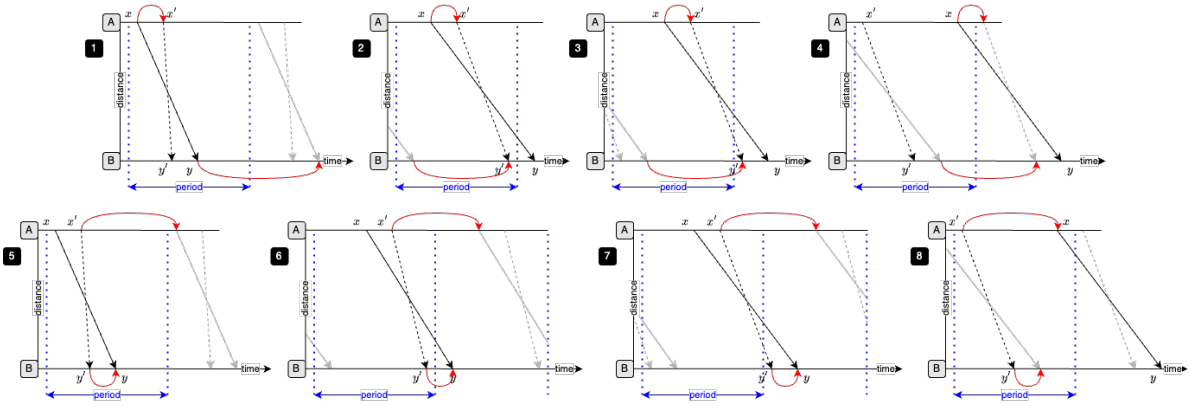


Figure 3.9: All possible situations where an overtaking is allowed, based on Zhang and Nie (2016)

	Direction of headways	Case	$p_{x,y}$	$p_{x',y'}$	$p_{x,x'}$	$p_{y,y'}$	Sum
Overtaking is prevented, see Figure 3.8	Forward direction	1	0	0	0	0	0
		2	0	1	0	1	2
		3	1	1	0	0	2
		4	0	0	1	1	2
		5	1	0	1	0	2
	Backward direction	6	0	0	1	1	2
		7	0	1	1	0	2
		8	1	1	1	1	4
		9	0	0	0	0	0
		10	1	0	0	1	2
Overtaking is allowed, see Figure 3.9	Forward direction	1	0	0	0	1	1
		2	1	0	0	0	1
		3	1	1	0	1	3
		4	1	0	1	1	3
	Backward direction	5	0	0	1	0	1
		6	1	0	1	1	3
		7	1	1	1	0	3
		8	1	0	0	0	1

Table 3.5: Relation between the sum of the modulo operators p and the detection of conflicts, based on Zhang and Nie (2016)

$$k_{c,s} = 0 \qquad \forall c = (a, a', a'') \in C_s^3 \text{ such that } dir_a = dir_{a'} = dir_{a''}, \forall s \in S \tag{3.25}$$

Constraint 3.25 ensures that on each segment, there is no combination of three different activities a, a' and a'' having the same direction such that each of them has a conflict with the two others. This constraint is needed in this research, since the maximal number of tracks per segment is limited to four tracks. Therefore it is not possible to have three trains running in the same direction overtaking each other, this would require more than two tracks reserved for each direction. However, one train is still allowed to overtake many slower trains on the same segment, and one train can be overtaken by many faster trains.

$$t_i^{\text{after}} = \max\{t_i^{\text{before}}, t_i^{\text{needed}}\} \qquad \forall i \in I \tag{3.26}$$

Constraint 3.26 ensures that if the initial number of tracks at an interlocking is lower than the minimal required number of tracks, the infrastructure is updated.

$$t_i^{\text{needed}} = \max_{c \in C_i} (q_{c,i} \cdot \text{len}(c)) \qquad \forall i \in I \tag{3.27}$$

Constraint 3.27 defines the minimum number of tracks needed at interlocking i to ensure that the timetable is conflict free, based on the number of conflicts detected. Each interlocking can have as many tracks as needed, computed as the maximal number of trains running, passing, dwelling or turning around at an interlocking at the same time. To do so, each possible combination of activities taking place at interlocking i is checked. Among the combinations for which $q_{c,i} = 1$, the combination with the larger number of activities is set to be the minimum number of tracks needed. As a reminder, $q_{c,i}$ takes the value 1 if and only if each activity of c is conflicted with every other one. Figure 3.10 shows some (non-exhaustive) examples of conflicts between four activities (a, a', a'', a'''). The number of tracks needed for each case is calculated in Table 3.6. The first case of this Table shows a situation where $t^{\textbf{needed}} = 1$, the number of track is here underestimated since at least one train is always needed

in a segment to allow trains to run on it. However, in that case, constraint 3.26 will always result in $t^{\text{after}} = t^{\text{before}}$.

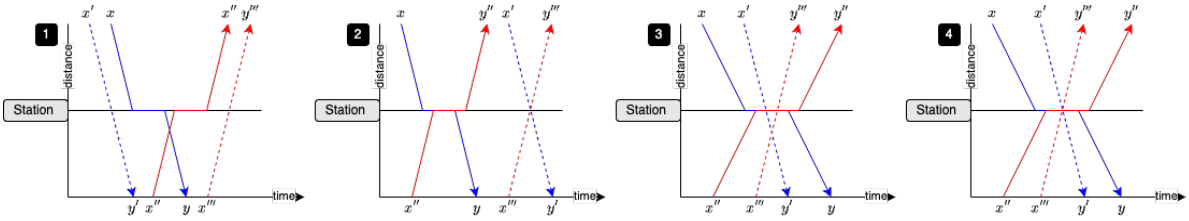


Figure 3.10: Relation between the number of conflicts detected and the number of tracks needed per interlocking

Case	Conflict between 2 trains $q_{c,i} \forall c \in C_i^2$	Conflict between + 2 trains $q_{c,i} \forall c \in C_i^k$ with $k > 2$	Number of tracks needed t^{needed}
1	$q_{a,a'} = 0$ $q_{a,a''} = 0$ $q_{a,a'''} = 0$ $q_{a',a''} = 0$ $q_{a',a'''} = 0$ $q_{a'',a'''} = 0$	$q_{a,a',a''} = 0$ $q_{a,a',a'''} = 0$ $q_{a,a'',a'''} = 0$ $q_{a',a'',a'''} = 0$ $q_{a,a',a'',a'''} = 0$	0
2	$q_{a,a'} = 0$ $q_{a,a''} = 1$ $q_{a,a'''} = 0$ $q_{a',a''} = 0$ $q_{a',a'''} = 1$ $q_{a'',a'''} = 0$	$q_{a,a',a''} = 0$ $q_{a,a',a'''} = 0$ $q_{a,a'',a'''} = 0$ $q_{a',a'',a'''} = 0$ $q_{a,a',a'',a'''} = 0$	2
3	$q_{a,a'} = 1$ $q_{a,a''} = 1$ $q_{a,a'''} = 1$ $q_{a',a''} = 1$ $q_{a',a'''} = 0$ $q_{a'',a'''} = 1$	$q_{a,a',a''} = 1$ $q_{a,a',a'''} = 0$ $q_{a,a'',a'''} = 1$ $q_{a',a'',a'''} = 0$ $q_{a,a',a'',a'''} = 0$	3
4	$q_{a,a'} = 1$ $q_{a,a''} = 1$ $q_{a,a'''} = 1$ $q_{a',a''} = 1$ $q_{a',a'''} = 1$ $q_{a'',a'''} = 1$	$q_{a,a',a''} = 1$ $q_{a,a',a'''} = 1$ $q_{a,a'',a'''} = 1$ $q_{a',a'',a'''} = 1$ $q_{a,a',a'',a'''} = 1$	4

Table 3.6: Calculation of t^{needed} in the different cases presented in Figure 3.10

$$q_{c,i} = \begin{cases} 1 & \text{if } (p_{x,y} + p_{x',y'} \\ & +p_{x,x'} + p_{y,y'}) \equiv 0 \pmod 2 \\ 0 & \text{otherwise} \end{cases} \qquad \forall c = \{(x,y),(x',y')\} \in C_i^2, \forall i \in I \tag{3.28}$$

Constraints 3.28 defines the binary help variable q which takes the value 1 if the activities a and a' are conflicted at interlocking i. This is done following the same methodology as the one used for defining constraint 3.24.

$$q_{c,i} = \begin{cases} 1 & \text{if } \sum_{c^*=(a,a')\in C_i^2 \text{ such that } a,a'\in c} q_{c^*,i} = \frac{k(k-1)}{2} \\ 0 & \text{otherwise} \end{cases} \qquad \forall c \in C_i^k \text{ with } k > 2, \forall i \in I \tag{3.29}$$

Constraint 3.29 defines the binary variable $q_{c,i}$. It takes the value 1 if every activity of the combination c is conflicted with every other. To do so, every possible pair of conflicts among the combination c is checked. If the sum of their binary variables $q_{c^*,i}$, computed in constraint 3.28 equals the number of possible pairs, then every activity is conflicted with each other. The number of possible pairs among a combination c of k activities equals $\binom{k}{2} = \frac{k!}{2!(k-2)!} = \frac{k(k-1)}{2}$, as showed in equation 3.1.

3.3.6. Linearization and implementation

The model previously formulated is not linear. Thus, constraints 3.15, 3.20, 3.21, 3.22, 3.23, 3.24, 3.26, 3.27, 3.28 and 3.29 need to be linearized. The linear mathematical formulation for the model is presented as follows. The non linear formulations are written on the left and their associated linear formulation are written on the right. The linear model is implemented in Gurobi and the code can be found in Appendix B.1.2.

Constraint 3.15

$$p_{x,y} = \begin{cases} 1 & \text{if } v_y < v_x \\ 0 & \text{otherwise} \end{cases} \qquad \forall a = (x,y) \in A$$

$$v_y - v_x \leq -\epsilon + M \cdot (1 - p_{x,y})$$

Constraint 3.22

$$k_s^{\text{opp}} = \begin{cases} 1 & \text{if } \sum_{\substack{c=(a,a')\in c_s^2 \\ dir_a \neq dir_{a'}}} k_{c,s} \geq 1 \\ 0 & \text{otherwise} \end{cases} \qquad \forall s \in S$$

$$\sum_{\substack{c=(a,a')\in c_s^2 \\ dir_a \neq dir_{a'}}} k_{c,s} \leq M \cdot k_s^{\text{opp}}$$

$$\sum_{\substack{c=(a,a')\in c_s^2 \\ dir_a \neq dir_{a'}}} k_{c,s} \geq k_s^{\text{opp}}$$

Constraint 3.23

$$k_s^{\text{same}} = \begin{cases} 1 & \text{if } \sum_{\substack{c=(a,a')\in c_s^2 \\ dir_a = dir_{a'}}} k_{c,s} \geq 1 \\ 0 & \text{otherwise} \end{cases} \qquad \forall s \in S$$

$$\sum_{\substack{c=(a,a')\in c_s^2 \\ dir_a = dir_{a'}}} k_{c,s} \leq M \cdot k_s^{\text{same}}$$

$$\sum_{\substack{c=(a,a')\in c_s^2 \\ dir_a = dir_{a'}}} k_{c,s} \geq k_s^{\text{same}}$$

Constraint 3.29

$$q_{c,i} = \begin{cases} 1 & \text{if } \sum_{\substack{c^*=(a,a')\in C_i^2 \\ a,a'\in c}} q_{c^*,i} = \frac{k(k-1)}{2} \\ 0 & \text{otherwise} \end{cases} \qquad \forall c \in C_i^k \text{ with } k > 2, \forall i \in I$$

$$\sum_{\substack{c^*=(a,a')\in C_i^2 \\ a,a'\in c}} q_{c^*,i} \leq \frac{k(k-1)}{2} + \epsilon + M \cdot q_{c,i}$$

$$\sum_{\substack{c^*=(a,a')\in C_i^2 \\ a,a'\in c}} q_{c^*,i} \geq \frac{k(k-1)}{2} - M \cdot (1 - q_{c,i})$$

The linear formulations of 3.15, 3.22, 3.23 and 3.29 use the big-M methodology developed in Section 3.1.

Constraint 3.20	
$t_s^{\text{after}} = \max(t_s^{\text{before}}, t_s^{\text{needed}})$ $\forall s \in S$	$t_s^{\text{after}} \geq t_s^{\text{before}}$ $t_s^{\text{after}} \geq t_s^{\text{needed}}$
Constraint 3.26	
$t_i^{\text{after}} = \max(t_i^{\text{before}}, t_i^{\text{needed}})$ $\forall i \in I$	$t_i^{\text{after}} \geq t_i^{\text{before}}$ $t_i^{\text{after}} \geq t_i^{\text{needed}}$

The linearization of the two constraints 3.20 and 3.26 is possible as such because, even though t_s^{after} and t_i^{after} are not directly minimized, the construction costs are, and therefore the optimal solution will always be the one with the lowest number of tracks to be constructed.

Constraint 3.21	
$t_s^{\text{needed}} = \begin{cases} 1 & \text{if } k_s^{\text{opp}} = 0 \text{ and } k_s^{\text{same}} = 0 \\ 2 & \text{if } k_s^{\text{opp}} = 1 \text{ and } k_s^{\text{same}} = 0 \\ 4 & \text{if } k_s^{\text{same}} = 1 \end{cases}$ $\forall s \in S$	$z_1 \leq k_s^{\text{opp}}$ $z_1 \leq k_s^{\text{same}}$ $z_1 \geq k_s^{\text{opp}} + k_s^{\text{same}} - 1$ $z_2 = 1 - k_s^{\text{same}} - k_s^{\text{opp}} + z_1$ $z_3 = k_s^{\text{opp}} - z_1$ $t_s^{\text{needed}} = z_2 \cdot 1 + z_3 \cdot 2 + k_s^{\text{same}} \cdot 4$ With z_1, z_2, z_3 binary variables

This linearization is not straightforward and needs further explanations. Firstly, the formulation on the left can be written as follows:

$$t_s^{\text{needed}} = z_2 \cdot 1 + z_3 \cdot 2 + k_s^{\text{same}} \cdot 4 \tag{3.30}$$

With

$$z_2 = (1 - k_s^{\text{opp}}) \cdot (1 - k_s^{\text{same}}) \tag{3.31}$$
$$z_3 = (k_s^{\text{opp}}) \cdot (1 - k_s^{\text{same}}) \tag{3.32}$$

By definition, the condition $z_2 + z_3 + k_s^{\text{same}} = 1$ is always satisfied and is therefore not needed. By developing the two equations 3.31 and 3.32, we obtain the two following notations:

$$z_2 = 1 + (k_s^{\text{opp}} \cdot k_s^{\text{same}}) - k_s^{\text{opp}} - k_s^{\text{same}} \tag{3.33}$$
$$z_3 = k_s^{\text{opp}} - (k_s^{\text{opp}} \cdot k_s^{\text{same}}) \tag{3.34}$$

The formulations 3.33 and 3.34 are not linear. Thus, a new binary variable $z_1 = k_s^{\text{opp}} \cdot k_s^{\text{same}}$ is introduced as follows:

$$z_1 \leq k_s^{\text{opp}} \tag{3.35}$$
$$z_1 \leq k_s^{\text{same}} \tag{3.36}$$
$$z_1 \geq k_s^{\text{opp}} + k_s^{\text{same}} - 1 \tag{3.37}$$

Constraints 3.35, 3.36, and 3.37 ensure that $z_3 = 1$ if and only if $k_s^{\text{opp}} = 1$ and $k_s^{\text{same}} = 1$. Finally, we obtain the six linear constraints presented on the right.

Constraint 3.24

$$q_{c,s} = \begin{cases} 1 & \text{if } (p_{x,y} + p_{x',y'} + p_{x,x'} + p_{y,y'}) \\ & \equiv 0 \pmod 2 \\ 0 & \text{otherwise} \end{cases}$$
$$\forall((x,y),(x',y')) \in C_s^2, \forall s \in S$$

$$k_{a,a',s,d} = (p_{x,y} + p_{x',y'} + p_{x,x'} + p_{y,y'}) - 2 \cdot z_4$$
With z_4 an integer variable

Constraint 3.28

$$q_{c,i} = \begin{cases} 1 & \text{if } (p_{x,y} + p_{x',y'} + p_{x,x'} + p_{y,y'}) \\ & \equiv 0 \pmod 2 \\ 0 & \text{otherwise} \end{cases}$$
$$\forall((x,y),(x',y')) \in C_i^2, \forall i \in I$$

$$q_{c,i} = (p_{x,y} + p_{x',y'} + p_{x,x'} + p_{y,y'}) - 2 \cdot z_5$$
With z_5 an integer variable

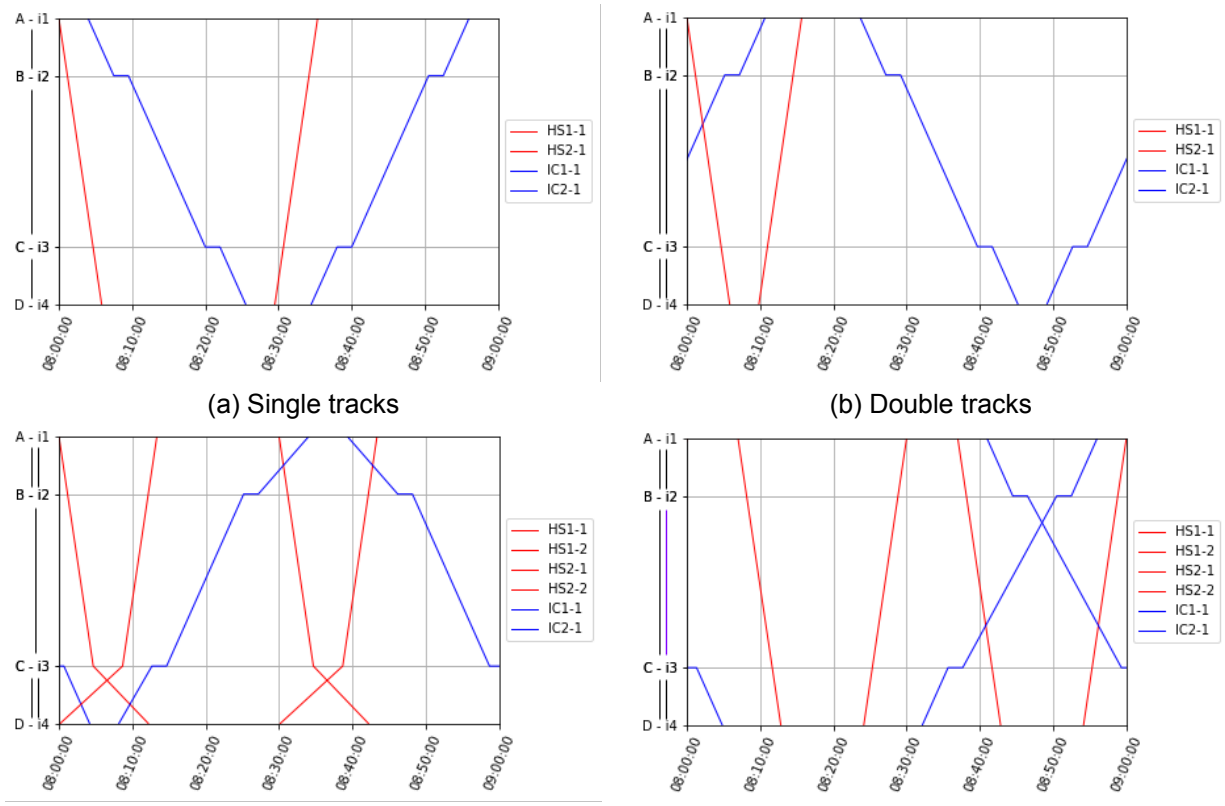
Constraints 3.24 and 3.28 are linearized using integer variables. The binary variables $q_{c,i}$ and $q_{c,s}$. The only way these variables can be equal to 0 is if the sum $p_{x,y} + p_{x',y'} + p_{x,x'} + p_{y,y'}$ is even. In the same way, these variables can only be equal to 1 is if the sum $p_{x,y} + p_{x',y'} + p_{x,x'} + p_{y,y'}$ is odd.

Constraint 3.27

$$t_i^{\text{needed}} = \max_{c \in \mathcal{C}_i}(q_{c,i} \cdot \text{len}(c))$$
$$\forall i \in I$$

$$t_i^{\text{needed}} \geq q_{c,i} \cdot \text{len}(c) \quad \forall c \in \mathcal{C}_i$$

3.4. Demonstration of the model on simple examples



(a) Single tracks

(b) Double tracks

(c) Bottleneck without possible upgrade

(d) Bottleneck with possible upgrades

Table 3.7: Model's application to four simple examples.

To demonstrate the model and illustrate the way it intends to work, it is applied to four simple examples presented in Table 3.7: (a) single track corridor, (b) double tracks corridor, (c) and (d) bottleneck corridor (double - single - double tracks) with high demand. In example (c), $t^{\max} = 0$ which forces the model to fit every train request without suggesting any infrastructure upgrade. In example (d), $t^{\max} = 1$ which allows the model to suggest an infrastructure upgrade.

These four examples show that the conflict restriction works as intended: there is no conflict on a single track. On a double track, the trains can meet each others. When the infrastructure is strained by high demand, the model can reduce the trains' speed and force opposite trains to meet on the double-track portions of the corridor, to ensure a conflict free timetable (c). If allowed to, the model can also suggest infrastructure improvement (d). Here, the bottleneck portion is upgraded to double tracks and the timetable therefore present conflicts between train running in opposite direction.

3.5. Data processing

As it has been shown in the previous Section 3.3.2, the PESP-MILP model takes many input parameters and sets of data. However, the data usually available are not usable as such by the PESP-MILP model and therefore need to be processed. This section outlines the various processes involved in transforming raw data into output data. Figure 3.11 presents the flow of data between the different functions of the model.

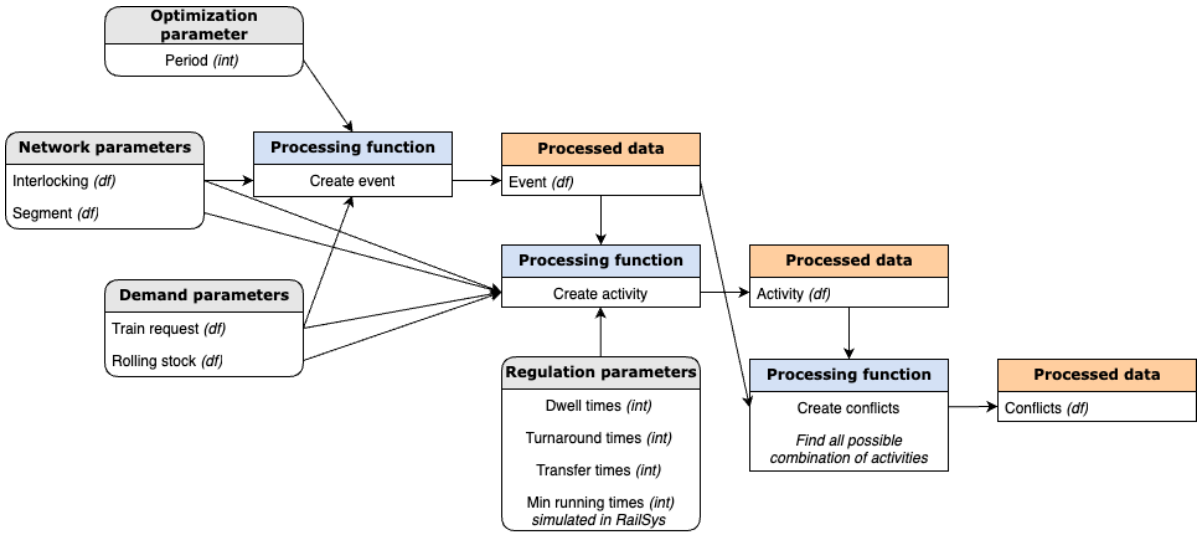


Figure 3.11: Flow of data between the different processing functions involved in the model

3.5.1. Raw data

Firstly, the network parameters are collected and stored in two different DataFrames. The first DataFrame `interlocking` contains all the interlockings of the network and their characteristics (unique identifier, code name, number of tracks, average cost to built a new track). The second DataFrame `segment` contains all the segments connecting two different interlockings along the network and their characteristics (unique identifier, the two connected interlocking, length, number of tracks, maximal speed). Tables 4.4 and 4.5 present the DataFrames `interlocking` and `segment` used for the case study.

The first DataFrame `train_request` contains every line requested to be scheduled by the PESP model. Each line has a unique identifier, a service, a path (list of every interlocking covered by the line), a list of stop stations (list of every station served by the line), a frequency, a rolling stock, cost of operation, and a list of interactions. Each line can interact with other lines. An interaction can be either a turnaround or a transfer. A turnaround implies that the two lines uses the same rolling stock. Thus, the interacted line has to depart *after* the considered origin line at its first station. In the same way, a

transfer implies that the passenger should be able to transfer from the origin line to the interacted line at a given station. Thus, the train of second line has to depart from the station *after* the arrival of the train of the first line.

3.5.2. Processed data

The raw data cannot be used as such by the PESP-MILP model. As indicated in Figure 3.11, the input data are first processed to create an event-activity graph. Firstly, the DataFrame `event` gives all the events the timetable has to satisfy, generated by the function `create_event`. The steps below summarize the logic and process implemented in the code, with the full version provided in Appendix B.1.1.

INPUT: `train_request` *(DataFrame)*, `interlocking` *(DataFrame)*, `period` *(integer)*.

1. **Initialization:**
   - Create an empty DataFrame `event` with columns `['eventID', 'lineID', 'event_type', 'event_location', 'event_frequency']`.
2. **Fill the empty DataFrame `event` by appending rows one by one:**
   - **For** each row of `train_request`:
     - The list `path` stores every interlocking in which the train goes (dwell or pass). The list `stop_stations` stores every station in which the train dwells.
     - Create multiple copies of each line depending on the `line_frequency` and calculate `event_frequency`
     - **For** each copy of the line:
       - `lineID` is modified by appending the copy number.
       - **For** each `interlocking` along the `path`:
         - **If** `interlocking` is the first in the `path`: an event of type `'dep'` is created for the first departure.
         - **If** `interlocking` is the last in the `path`: an event of type `'arr'` is created for the last arrival.
         - **Else** `interlocking` is an intermediate: two events are created: one for arrival (`'arr'` if `interlocking` is in `stop_stations` or `'arr-pass'` otherwise) and one for departure (`'dep'` if `interlocking` is in `stop_stations` or `'dep-pass'` otherwise).
       - Each event is stored in the `event` DataFrame with the relevant details.
3. **Finalization:**
   - The `'eventID'` column is set as the index of the `event` DataFrame to uniquely identify each event.

OUTPUT: `event` *(DataFrame)*.

Secondly, the DataFrame `activity` gives all the activities between each event. Each activity is identified by a unique identifiers and has different characteristics: a type (running, dwelling, transfer, headway, turnaround, passing through, regularity), a location (segment or interlocking in which the activity takes place), a minimum duration (the min duration of the activity) and a cost. The cost of the activity $a = (x, y)$ is set as the cost of operation of the line weighted by the type of activity. This allows to choose to minimize specific types of activity. In this research, the focus is given to minimizing the travel times, i.e. the running, dwell and passing-through times.

$$c_{x,y} = c_l \cdot c_{typ_{x,y}}$$

With:

$$c_{typ_{x,y}} = \begin{cases} 1 & \text{if } typ_{x,y} = \text{"run"} \\ 1 & \text{if } typ_{x,y} = \text{"dwell"} \\ 1 & \text{if } typ_{x,y} = \text{"pass"} \\ 0 & \text{if } typ_{x,y} = \text{"headway"} \\ 0 & \text{if } typ_{x,y} = \text{"turnaround"} \\ 0 & \text{if } typ_{x,y} = \text{"transfer"} \\ 0 & \text{if } typ_{x,y} = \text{"regularity"} \end{cases}$$

The function `create_activity` generates the DataFrame `activity`. The steps below summarize the logic and process implemented in the code, with the full version provided in Appendix B.1.1

INPUT: `train_request` *(DataFrame)*, `interlocking` *(DataFrame)*, `segment` *(DataFrame)*, `event` *(DataFrame)*, `rolling_stock` *(DataFrame)*, `regulation` *(DataFrame)*.

1. **Initialization:**
   - Create an empty DataFrame `activity` with columns `['activityID', 'from_eventID', 'to_eventID', 'activity_type', 'activity_location', 'min_activity_duration', "activity_cost', 'train_direction']`.
2. **Create run, dwell, and pass activities:**
   - **For** each row of `train_request`:
     - **For** each pair of consecutive events within the `lineID`
       - Check the direction of the train by comparing the firt and last interlocking
       - **If** the train moves forward, the direction is set to 1.
       - **If** the train moves backward, the direction is set to -1.
       - Classify the activities
       - **If** the events are `arr-pass` followed by `dep-pass` activity is `pass` with the location set to the `from_interlockingID`.
       - **If** the events are `arr` followed by `dep` activity is `dwell` with the location set to the `from_interlockingID`.
       - **Else** the activity is `run` with the location determined based on the segment that connects the `from_interlockingID` and `to_interlockingID`. The minimum duration is calculated.
       - Add activity to `activity` with unique ID and corresponding information.
3. **Create transfer and turnaround activities:**
   - **For** each interaction: calculates the corresponding event pairs between the interacting lines.
     - **If** the interaction is `turnaround`, its duration and cost are determined based on the regulation.
     - **If** the interaction is `transfer`, its duration and cost are determined based on the regulation.
     - Add activity to `activity` with unique ID and corresponding information.
4. **Create headway activities**
   - Identify pairs of arrival events occurring at the same location but from different lines.
   - **For** each pair:
     - Add activity to `activity` with unique ID and corresponding information.
5. **Create regularity activities**
   - **For** each copy of the same event
     - **If** if the two events are consecutive events
       - Add activity to `activity` with unique ID and corresponding information.
6. **Finalization:**
   - The `'activityID'` columns is set as the index of `activity` to uniquely identify each activity.

OUTPUT: `activity` *(DataFrame)*

The `create_conflict` function generates conflicts between train activities based on the location

and types of activities. The function returns four DataFrames, `K`, `K_multiple`, `Q`, and `Q_multiple`, representing different types of conflicts and their combinations. The steps below summarize the logic and process implemented in the code, with the full version provided in Appendix B.1.1.

INPUT: `activity` *(DataFrame)*, `event` *(DataFrame)*.

1. **Initialization:**
   - Create an empty DataFrame `K` with columns `['conflictID', 'pair', 'd', 'conflict_location']`
   - Create an empty DataFrame `K_multiple` with columns `['combinationID', 'combination', 'conflict_location']`
2. **Fill `Q` with pair of activities:**
   - Filter `activity` to only keep rows where `activity_type` is `'dwell'` or `'pass'` or `'run'`. Group `activity` by `activity_location`.
   - **For** each location:
     - Generates all possible pairs of activities using `itertools.combinations`.
3. **Fill `Q_multiple` with combination of more than 2 activities**
   - Filter `activity` to only keep rows where `activity_type` is `'dwell'` or `'pass'` or `'dwell'`. Group `activity` by `activity_location`.
   - **For** each location:
     - Generates all possible combination of at least 3 activities using `itertools.combinations`
     - Generates all possible pair of activities among the combination using `itertools.combinations`
     - **If** every pair of activity has been created and is present in `K`, the combination is added to `Q_multiple` with the corresponding combination of activity, combination of conflict, `d` and `conflict_location`.
4. **Finalization:**
   - The `'conflictID'` columns are set as the indexes of the four DataFrame to uniquely identify each combination.

OUTPUT: `K` *(DataFrame)*, `K_multiple` *(DataFrame)*.

3.5.3. Output data

The proceed data are then taken as input parameters of the PESP-MILP optimization function, presented in Appendix B.1.2. The PESP model gives back the optimized time v_x at which each event x takes place. Then, as presented in Figure 3.12, the obtained timetable is visualised as the form of a time-distance diagram, thanks to a function that has been developed for the this study. The function code is presented in Appendix B.1.3. This allows the user to quickly analyse the output of the model and to detect eventual errors. This visualisation is also flexible since the user can choose the time window to be plotted and to customize the tick spacing.

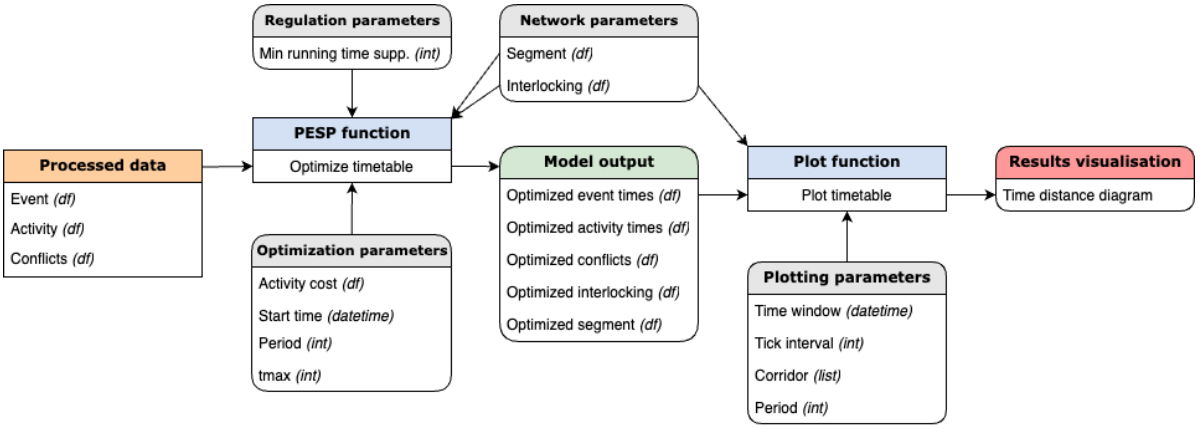


Figure 3.12: Flow of data between the optimization and visualization functions

3.6. Hypotheses

The model focuses on the macroscopic level and therefore uses a simplified network and data. Thus, several hypotheses have been made. These hypotheses and their consequences on the results are discussed in Section 5.7.

Firstly, any train can dwell on any track of any station. The number of tracks in a station is considered equal to the number of platforms, which is not always the case. Some tracks do not have any platform and are only used to allow a train to pass the station without having to stop. Moreover, the length of the trains and the platforms are not considered, but in real life, some trains may be too long to be able to dwell on a small platform. This might lead to an underestimation of the number of tracks needed per interlocking. However, the consequences of this hypothesis are limited if the user carefully chooses the trains that are running in their experiments, and makes sure no train is requested to stop in stations with too small platforms.

Secondly, a line can only have one, two or four tracks. This hypothesis results in an over-restriction of the possible conflicts per track but reflects the state of the infrastructure in Sweden, where lines are usually single, double or four tracks (only two lines have four tracks). Additional overtaking tracks (i.e. meeting stations) can be found, but they should be considered as interlocking in this model.

4

Case study

4.1. Context

The Mälarbanan (Figure 4.1) is one of the busiest railways in Sweden, connecting Stockholm to Öbebro via Västerås. The railway runs for 200 km, going through the north of the Mälaren lake. The corridor is, as the large majority of the Swedish network, managed by the Swedish Transport Administration (Trafikverket (2023b)). The line is fully electrified and has been upgraded in 2005 to be able to run high-speed trains with a maximal speed up to 200km/h. Nowadays, the line is mainly used for passenger trains (intercity, regional, and commuter trains) operated by Statens Järnvägar (SJ, the state-owned operator) and Mälardalstrafik (a regional operator). However, freight companies also operate on the line (Trafikverket (n.d.-b)). The line mainly consists of double tracks, but a part of the line still consists of single tracks, challenging the increasing demand.

Figure 4.1: The Mälarbanan *(in green)* and a part of the Swedish network *(in black)* (Trafikverket (n.d.-b))

The Mälarbanan is strategic for the long-term planning of the Swedish network. Several studies have recently been conducted to assess potential upgrades, such as the Oslo-Sthlm 2.55 project which aims to shorten the travel time between Oslo and Stockholm. Currently, trains connecting the two Scandinavian capitals pass through the south of the Mälaren lake: they do not run on the Mälarbanan. However, Berner et al. (2021) shows that a deviation of the trains to the Mälarbanan, supported by the construction of new tracks and the upgrade of the existing lines would reduce the travel time to 2 hours

55 minutes instead of the 5 hours and 8 minutes currently operated by the SJ 3000 high-speed train. Figure 4.2 depicts the current route of the train connection Oslo and Stockholm, and the construction planned by the Oslo-Sthlm 2.55 project. Thus, the Mälarbanan has a strategic interest and this case study is set within the context of the project of upgrading the capacity of the line. However, as of 2024, the Oslo-Sthlm 2.55 project is only at the study stage and no project has been officially approved by the Swedish Transport Administration.

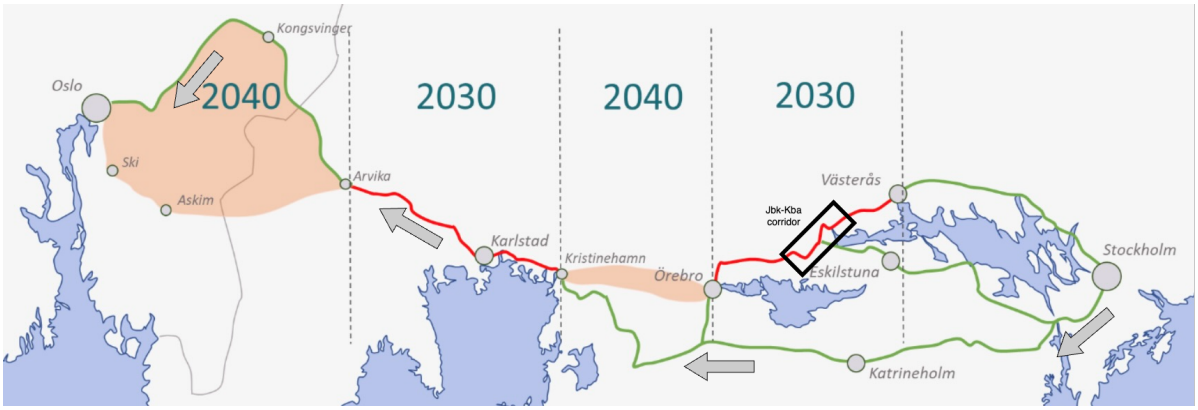


Figure 4.2: Project Oslo-Sthlm 2.55. *Green lines:* existing lines. *Red lines:* Existing lines to be upgraded to double tracks. *Orange area:* New portion of the line to be constructed. *Grey arrows:* Current path of the trains connecting Stockholm and Oslo. *Black rectangle:* Corridor between Jädersbruk and Kolbäck investigated in this case study. ("Varför är Oslo-Sthlm viktigt?" (n.d.))

The model is applied to a portion of the Mälarbanan, between Jädersbruk and Kolbäck (see black rectangle above). Figure 4.3 shows the microscopic infrastructure of the line modeled in RailSys. The line mainly consists of single tracks and the Oslo-Sthlm 2.55 project proposes to upgrade the line to double tracks. The line consists of three passenger stations (Arboga, Köping, and Kolbäck) and three meeting stations (Jädersbruk, Valskog, and Munktorp). Meeting stations are stations without platforms that are only used to allow trains to stop and overtake each other, but also to connect two converging or diverging lines. Therefore there is no possible entering or exit of passengers at those stations. The stations are officially abbreviated by the Transport Administration as follows: Jädersbruk (Jbk), Arboga (Arb), Valskog (Vsg), Köping (Kp), Munktorp (Morp) and Kolbäck (Kbä).

4.2. Scenarios

This case study aims to check whether the current infrastructure of the Jkb-Kbä corridor could absorb the long-term demand, and if not, to find the optimal infrastructure improvements. To do so, different scenarios are formulated. The traffic data of 2022 and traffic forecast data of 2040 are tested, to check whether the projected long-term demand could fit the current infrastructure. In addition, the 2040 scenario is also tested with additional high-speed trains, to evaluate the feasibility of the Oslo-Sthlm 2.55 project. All of those scenarios are applied to the current state of the infrastructure, as presented in Figure 4.3.

	Situation	**Traffic (per hour, per direction)**				**Coefficients**		**Buffer times**
		F	R	IC	HS	α	β γ	(in s)
Scenario 1	2022	1	1	1	0	0.5	0.25	60
Scenario 2	2040	1	2	1	0	0.5	0.25	60
Scenario 3	2040 + HS trains	1	2	1	1	0.9	0.05	0

Table 4.1: Different scenarios of the case study. *F:* Freight, *R:* Regional, *IC:* Intercity, *HS:* High-speed

These three scenarios are applied with the same conditions, according to the Swedish regulations introduced in Section 2.4.1.

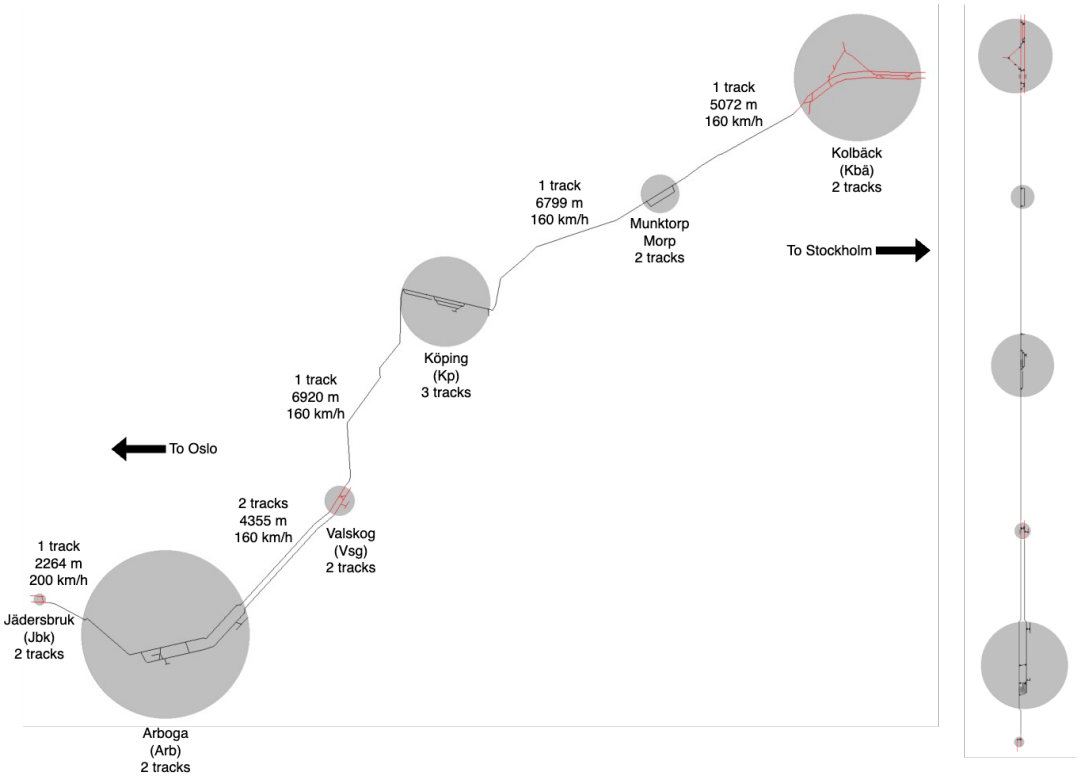


Figure 4.3: Case study line, modeled at the microscopic level

Parameter	Value
Running time supplement	8% of the minimum technical running time
Dwell times	2 min
Headway times	3 min
Period Π	60 min

Table 4.2: Regulation parameters used for the case study

In every scenario, the coefficients β and γ are such that $\beta = \gamma = (1 - \alpha)/2$, so that the model equally minimizes the segment construction costs and the interlocking construction costs. However, the demand is high for the third scenario, resulting in long computer running times. To speed up the optimization process of this scenario, the coefficient α is then increased to 0.9. Therefore, the model focuses more on minimizing the operation costs, and the minimization of the construction costs is relaxed, compared to scenarios 1 and 2 where $\alpha = 0.5$. The buffer times are set to 1 minute and are added to the headway constraints, resulting in headway times equal to 4 minutes (3 minutes headways + 1 minute buffer), making the problem more complex to solve. These buffer times are reduced to 0 for the third scenario, to relax the headway constraints and speed up the optimization process. This simplification will likely result in an underestimation of the construction costs and is discussed in Section 5.1.

4.3. Data collection

The train requests presented in Table 4.1 are extracted from the 2022 railway traffic data and noise forecast 2040 (Trafikverket (2021)). This database centralizes the number of trains and their type running in 2022 and expected to run in 2040 on every corridor of the Swedish network. Appendix C presents the data of the Jkb-Kbä corridor, the values are rounded since the model is restricted to integer frequencies. The high-speed trains running between Oslo-Stockholm are extracted from the Arvika-Charlottenberg corridor, located close to the Norwegian border and currently used by the fast train connecting Stockholm and Oslo. These data are used for the third scenario, assuming that the diminution of the travel

times due to the new path of the HS trains will not have any effect on the expected demand. No possible transfers have been added to simplify the optimization process and reduce the computer running times. Since the corridor is only a small portion of the line, trains do not turn over at any of the investigated stations. Regional trains dwell in Kolbäck and Arboga, intercity trains dwell in Köping and Arboga, and freight and high-speed trains are not required to dwell at any station. However, since the operation costs of freight trains are lower (see Table 4.3), the simulation model might prioritize other trains and therefore schedule dwell times to let passenger trains overtake freight trains.

Type of service	Operation costs (SEK/s)
Freight trains	4.31
Regional trains	4.53
Intercity trains	9.17
High-speed trains	13.98

Table 4.3: Operation costs of the different trains (Trafikverket (2024a))

The infrastructure data modeled in RailSys have been simplified to be used at the macroscopic level. The number of tracks per station is easily observable in RailSys, but not every track is taken into account at the macroscopic level. For example, the additional tracks used for turning over at stations Arb, Vsg, and Kp are not kept. Indeed, the macroscopic model only considers tracks on which the train can run and dwell. Moreover, the convergent track of station Kbä is also not taken into account since it is part of another corridor. Finally, the minimum technical running times of the different trains requested to run in the different scenarios of the case study are simulated in RailSys. The `segment` and `interlocking` DataFrames taken as inputs of the model are presented in Tables 4.4 and 4.5.

Interlocking	Station code	Tracks	Cost (M SEK)
i1	Jbk	2	25
i2	Arb	2	51
i3	Vsg	2	25
i4	Kp	3	64
i5	Morp	2	25
i6	Kbå	2	30

Table 4.4: Interlockings of Jkb-Kbä corridor

Segment	From	To	Length	Tracks	Min running time (s)				Cost
			(m)		F	R	IC	HS	(M SEK)
s1	i1	i2	2264	1	124	106	75	78	844
s2	i2	i3	4355	2	392	227	207	212	1624
s3	i3	i4	6920	1	408	274	276	258	2581
s4	i4	i5	6799	1	408	258	267	255	2536
s5	i5	i6	5072	1	307	172	175	179	1892

Table 4.5: Segments of Jkb-Kbä corridor

Following the discussion in Section 2.5, the construction costs of one track for each segment or interlocking are calculated based on the cost per unit of distance, under the assumption that the construction costs are proportional to the length of the track that has to be built. The cost to build one track at a station is set to 0.2 million SEK/platform-meter, and the cost for building one track on a segment is set to 373 million SEK/track-km. This value is extracted from a study estimating the construction costs of a high speed track being built from scratch (Trafikverket and Jacobs (2021)). The case study investigated in this research only consists of conventional tracks. Thus the construction costs would likely be lower in real life. Nevertheless, this overestimation should not have any consequences on the optimal infrastructure modifications suggested by the model, since the costs are normalized using the

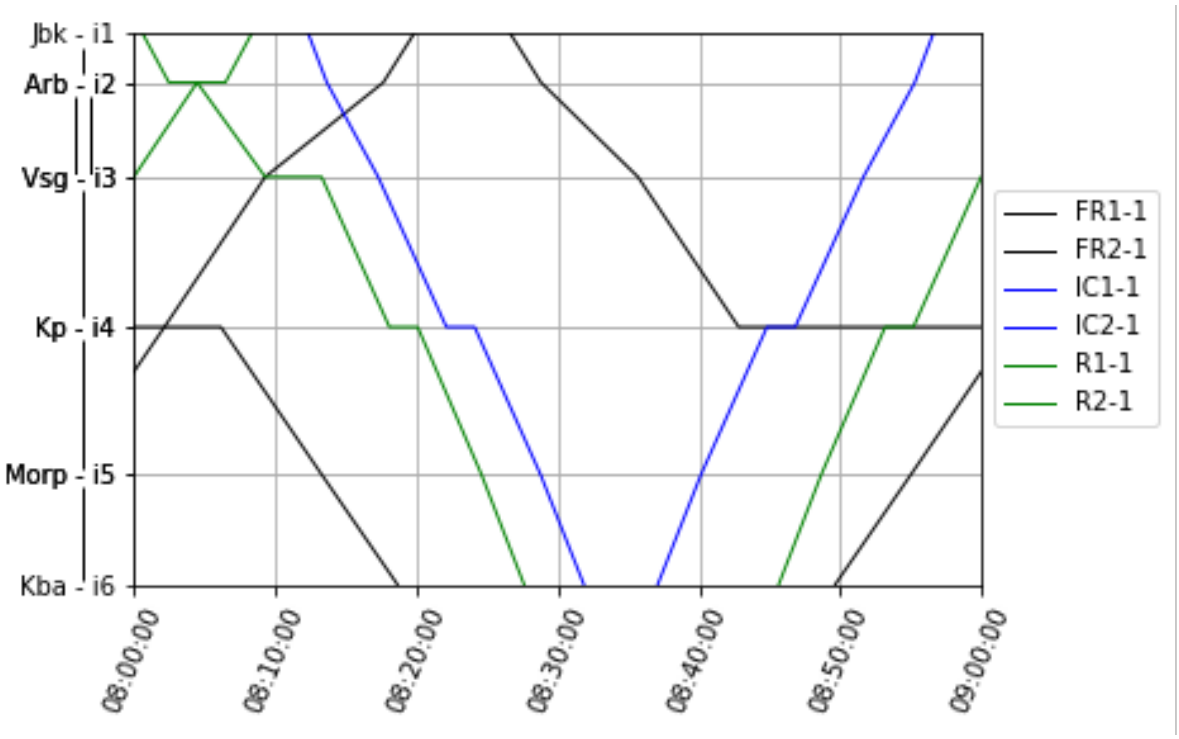
min-max normalization reported in Equation 3.4.

4.4. Results

4.4.1. Macroscopic timetables

The optimal timetables and infrastructure upgrades obtained when simulating the three scenarios, and their optimization statistics are given as follows. The initial number of tracks per segment is indicated on the distance axis of the diagrams, and the new tracks suggested to be built are depicted in purple. Appendix D.1, D.2 and D.3 present the DataFrames `event` given as output of scenarios 1, 2 and 3.

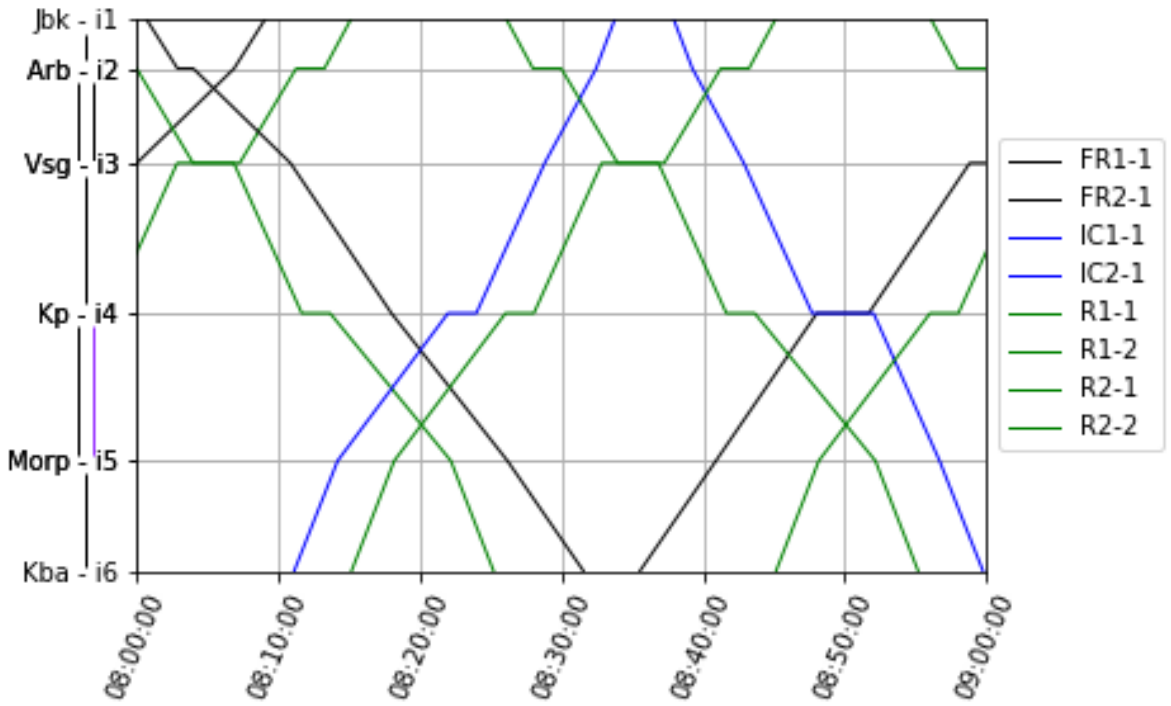
Figure 4.4: Optimal timetable of scenario 1



Infrastructure to upgrade:	None
Total operation costs:	54,712 SEK
Total construction costs:	0 SEK
Objective value:	0.150466
Number of variables:	953 (incl. 532 binary)
Number of constraints:	1691
Number of iterations:	82,969,255
CPU:	743 s
Optimally gap:	4.9%

Figure 4.4 shows that the optimal solution does not require any infrastructure upgrade, therefore the construction costs are null. Since the portion Arb-Vsg has double tracks, conflicts between two trains running in opposite directions are allowed. No other conflicts are detected anywhere else and the headway constraints are satisfied. The model forces one freight train to dwell for 20 minutes in station Kp.

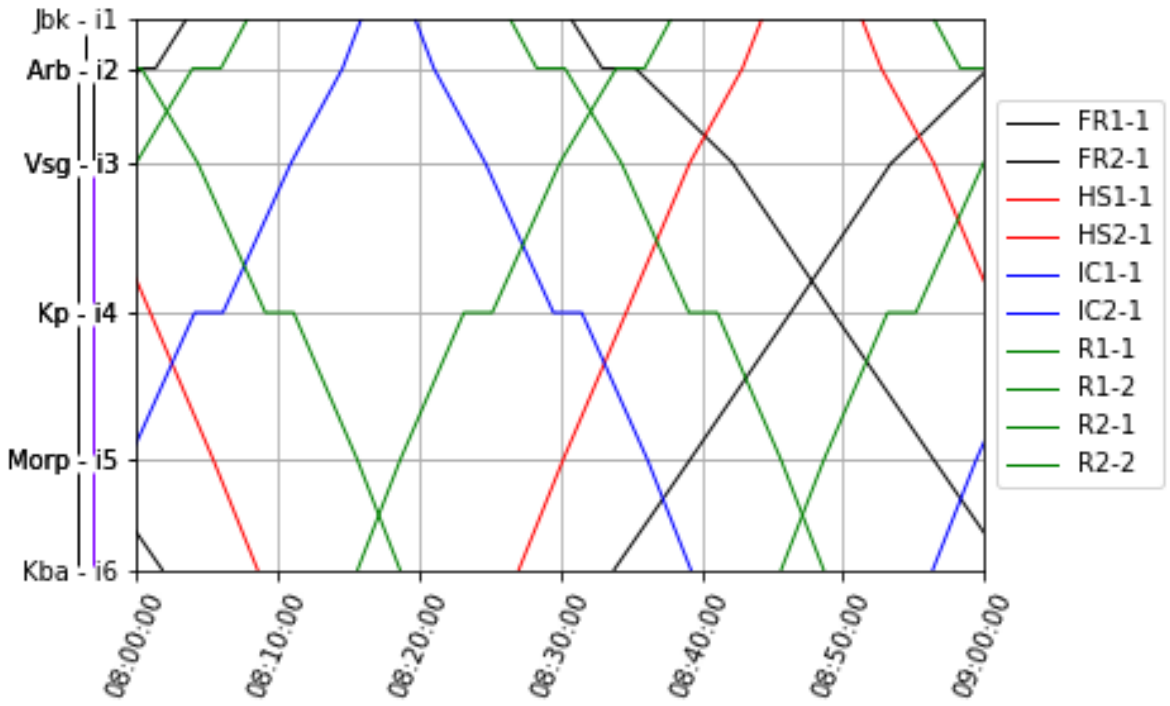
Figure 4.5: Optimal timetable of scenario 2



Infrastructure to upgrade:	Kp-Morp to double tracks
Total operation costs:	72,022 SEK
Total construction costs:	2,536 million SEK
Objective value:	0.242126
Number of variables:	1,770 (incl. 1,062 binary)
Number of constraints:	3,384
Number of iterations:	522,390,696
CPU:	5,810 s
Optimally gap:	4.9%

Figure 4.5 shows that the optimal solution requires an infrastructure upgrade. The model suggests upgrading the portion Kp-Morp to double tracks, which allows conflicts between two trains running in opposite directions. The proposed infrastructure upgrade results in a succession of portions with single and double tracks. The two freight trains have short additional stops in stations Arb and Kp. The regularity constraints between the different regional trains are satisfied, as well as the headways between the different trains.

Figure 4.6: Optimal timetable of scenario 3

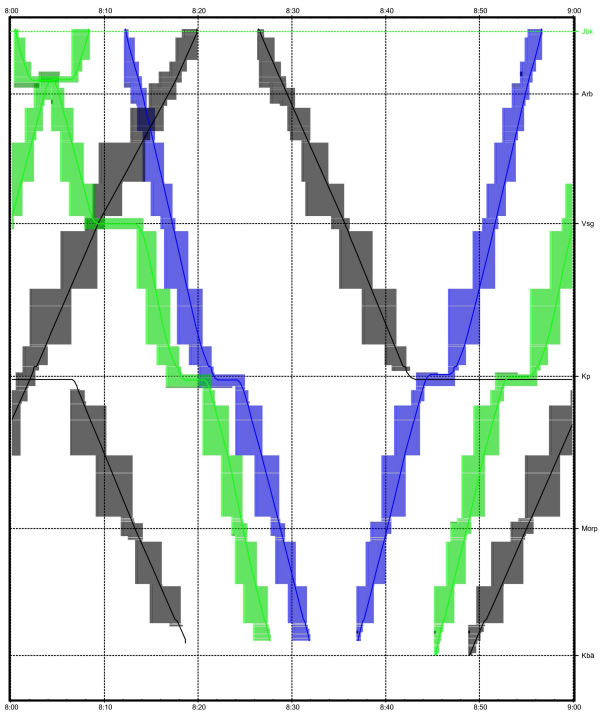


Infrastructure to upgrade:	Kp-Morp to double tracks
Total operation costs:	77,728 SEK
Total construction costs:	7,009 million SEK
Objective value:	0.089249
Number of variables:	4,129 (incl. 3,060 binary)
Number of constraints:	9,639
Number of iterations:	312,712,226
CPU:	3,598 s
Optimally gap:	4.9%

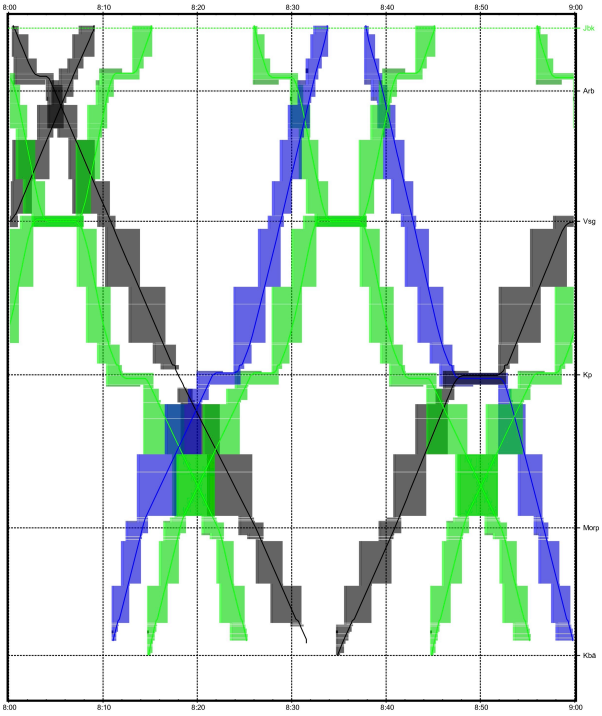
Figure 4.6 shows that the optimal solution requires upgrading four portions to double tracks: Vsg-Kp, Kp-Morp and Morp-Kba. Trains all run at their maximal possible speed and only the freight trains have additional stops at station Arb. The headway and regularity constraints are still satisfied.

4.4.2. Microscopic assessment of the results

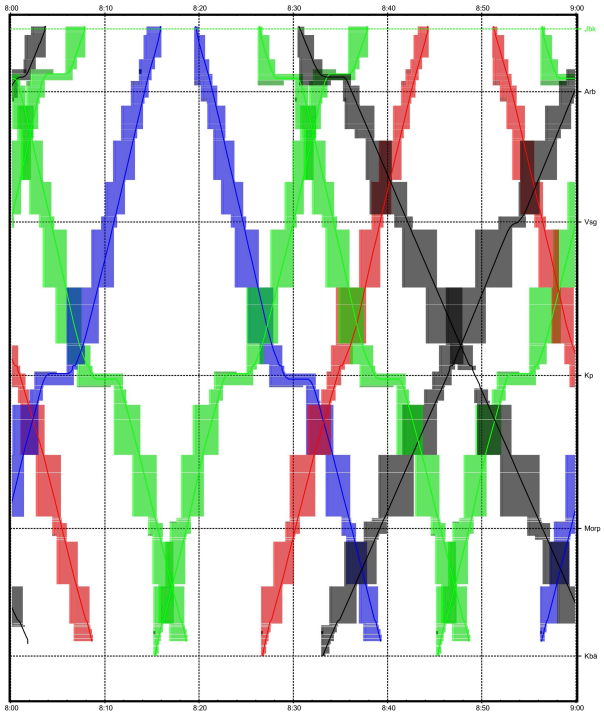
Previous results show conflict-free timetables at the macroscopic level. However, their feasibility at the microscopic level is not guaranteed since several hypotheses have been made and the detailed signalling of the infrastructure has not been considered. Thus, the produced timetables are simulated in RailSys to assess their feasibility at the microscopic level. Modeling new tracks is complex and time-consuming: it requires further investigations on the exact signalling and block sections the tracks should have. Therefore, the infrastructure modifications suggested by the macroscopic model have not been modeled in RailSys, and timetables are simulated on the initial state of the infrastructure (see Figure 4.3) and are presented in Table 4.6.



Scenario 1



Scenario 2



Scenario 3

Table 4.6: Microscopic simulation of the different timetables obtained at the macroscopic level

4.4.3. Sensitivity analysis

To assess the sensitivity of the optimal solution to the coefficient α, scenario 2 has been tested with different values of α, keeping all the other parameters identical. Figure 4.7 shows the various costs of the optimal solutions.

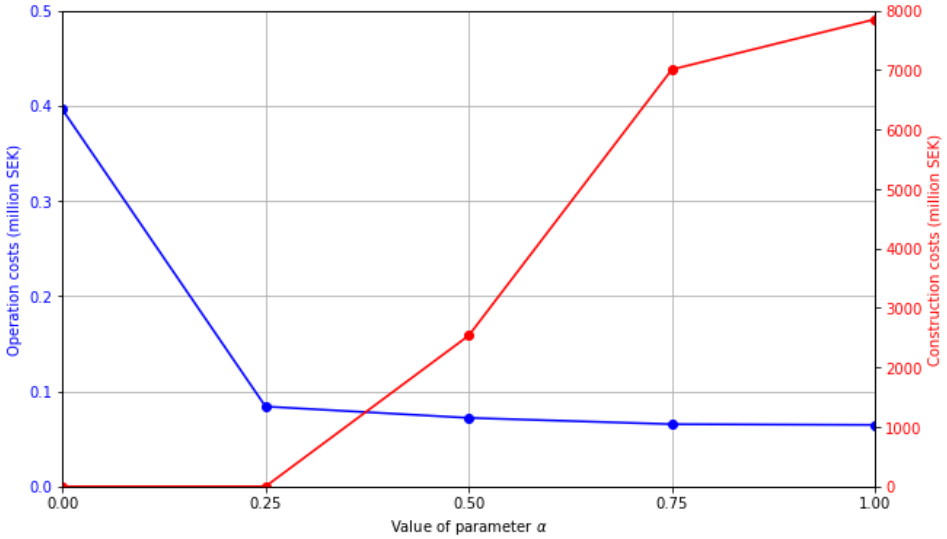


Figure 4.7: Costs of the optimal solution of scenario 2 under different values of α

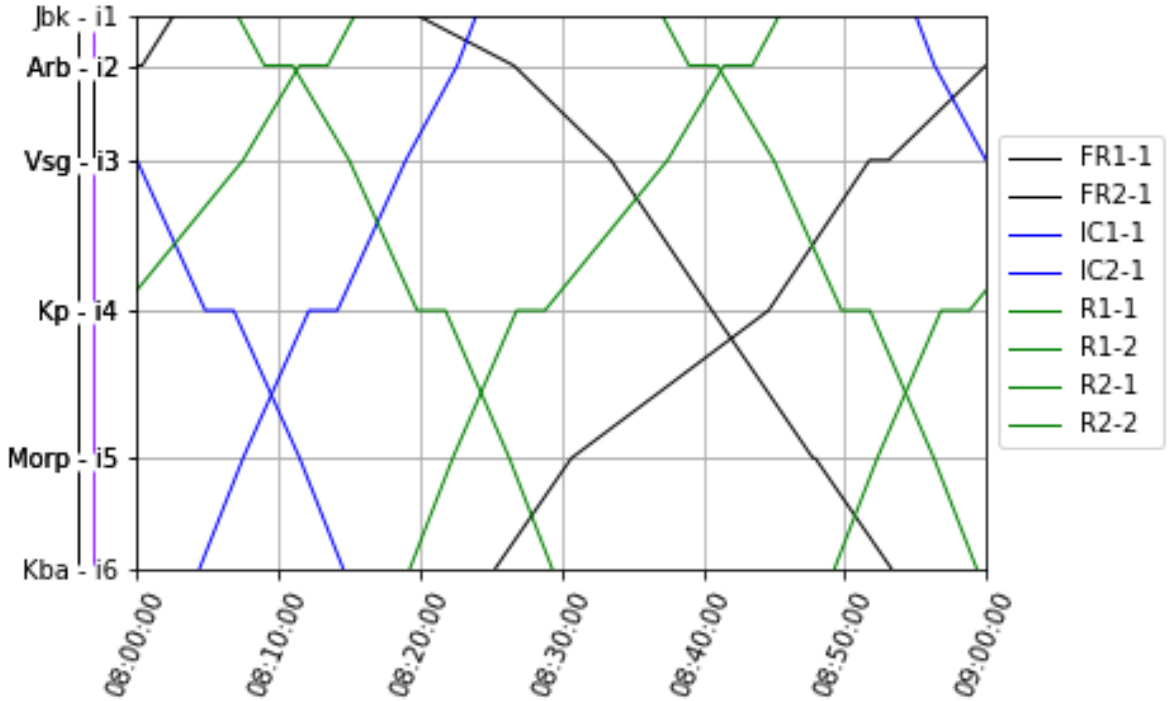
To assess the sensitivity of the computer running times to the value parameter α, the CPU of the optimal solutions presented in Figure 4.7 are listed in the table bellow.

α	0	0.25	5	0.75	1
CPU (s)	0.31	12971	5809	23157	1208

Table 4.7: Computer running times of scenario 2 under different values of α

The timetable of the extreme case where $\alpha = 1$ is presented in Figures 4.8.

Figure 4.8: Optimal timetable of scenario 2, with $\alpha = 1$



5 Discussion

5.1. Analysis of the case study's results

Results show that the model can handle the key constraints: the headway times, minimum running times, dwell times, and regularity constraints are all satisfied and no conflict is allowed where the infrastructure is only constituted of a single track. Since the buffer times have been introduced within the headway times, the model has to schedule a timetable with high headway times, and deleting the buffer times strongly decreases the CPU. No optimal solution presents high travel times (with trains running extremely slow or trains dwelling for a long time) or high infrastructure upgrades (none of the solutions suggest upgrading a portion to four tracks). This demonstrates that the model suggests realistic solutions.

The result of the first scenario shows that the current state of the infrastructure can absorb current demand. This result is expected since this scenario is the current real-life scenario, and obtaining an infeasible demand would have suggested that the model is over-constrained. The result of the second scenario shows that the model can suggest infrastructure upgrades, as intended. The current state of the infrastructure would not be able to meet the 2040 projected demand. Finally, the result of the third scenario shows that the suggestion of the Oslo-Sthlm 2.55 project of re-routing high-speed trains to the Mälarbanan line would require upgrading the Vsg-Kba portion to double tracks. This result aligns with several studies also proposing to construct a double track all along the Mälarbanan line (Sweco (2017)). However, studies estimate that the reduction of the travel time between Oslo and Stockholm would lead to an increase in the passenger demand of 116% and in the number of trains running of 70% (Jernbanedirektoratet and Trafikverket (2022)). The high-speed demand taken as input in scenario 3 does not consider the expected increased demand, and the demand is therefore underestimated.

The optimal solution can present a timetable with unsolicited stops, mostly concerning freight and regional trains. These two services have the lowest operation costs and therefore have low priority. This illustrates the trade-off the model has to optimize: upgrading the infrastructure to avoid these additional stops and to lower as much as possible operations costs is not necessarily optimal. The optimal timetable of the second scenario shows seven different conflicts taking place on the Kp-Morp portion of the line, which is upgraded to double tracks. This portion is the most expensive one to upgrade, but also the longest one and therefore the one having the most benefits on traffic. Therefore, if the model suggests an infrastructure upgrade, the optimal timetable takes as much as possible advantages of the new construction. The optimal infrastructure of the second scenario is a corridor of single - double - single - double - single tracks. Thus, this result also suggests that it is more efficient to spread the double tracks all along the corridor rather than having a long portion of double tracks. Indeed, this allows to have two distinct areas where opposite trains can meet each other.

None of the optimal solutions suggests the construction of new tracks at an interlocking, even though their construction costs are low. This could be explained in different ways. Firstly, every station in the case study already has two tracks, which allow trains to overtake and to meet each other. Thus, the

benefits of building a new track in one of these stations are quite low. Upgrading a segment from single to double has more benefits on traffic and leads to a stronger reduction of operation costs. Secondly, the costs are normalized using a min-max normalization, and the coefficients β and γ are equal: the model equally minimizes the construction costs of new tracks on the segments and the interlockings. The cost-benefit ratio therefore favors constructing a new track along one segment.

A periodic symmetric timetable is defined as a timetable where opposite trains always meet each other at times summing up to either 0 or the period Π (Liebchen (2004)). The formulation of the model does not require symmetric timetables, and none of the optimal timetables given as outputs of the case study are symmetric. This suggests that a symmetric timetable would not be optimal in the investigated cases. This result aligns with the findings of Liebchen (2004) showing that symmetry leads to sub-optimality when minimizing passenger travel times. However, the three timetables can show axes where the train's schedules mirror across these. For scenario 1, this axe takes place around 8:35. For scenario 2, two axes can be noticed around 8:20 and 8:35 Finally, scenario 3 shows two axes around 8:18 and 8:47. Results also give an idea on the way the model handles the heterogeneity of the demand. The third scenario has a heterogeneous demand: fast trains do not dwell at any station and slower trains are requested to dwell at two different stations. The optimal timetable groups the trains that do not stop together: in each direction, the freight and high-speed trains follow each other. These results align with the current knowledge of traffic optimization claiming that homogenization results in higher capacity utilisation.

5.2. Microscopic feasibility

The microscopic simulation of the timetables shows the different blocking times. Conflicts are detected between trains running in opposite directions. This aligns with our expectations since the infrastructure has not been upgraded in RailSys, and is therefore still composed of single tracks, even on the portions suggested to be upgraded to double tracks. Therefore, these conflicts can be omitted in this research. Timetables 1 and 3 show that the blocking times between two following trains do not overlap: the headway times are sufficient. However, scenario 2 presents, on the corridor Morp-Kp, many conflicts between 8:00 and 8:30, but it is not clear whether these conflicts take place between trains running in the same or opposite direction. Figure 5.1 gives a better view of these conflicts by zooming on the conflicted area and by dividing trains per direction. Blocking times of the regional and intercity trains overlap for a minute on two block sections of the corridor. The same goes for the regional and intercity trains whose blocking times overlap for a few seconds in one block section. This can be explained by the length of these two blocks which are longer than the other blocks of the corridor. The longer the block length, the longer the blocking time. Nevertheless, these three conflicts are minimal and could be easily resolved at the microscopic level by slightly re-scheduling one or the other train to an earlier or later departure, to ensure their blocking times do not overlap. At the macroscopic level, these conflicts could be avoided by reinforcing the headway constraint on the Kp-Morp portion of the corridor, by increasing its value to four minutes (instead of three on the other block sections).

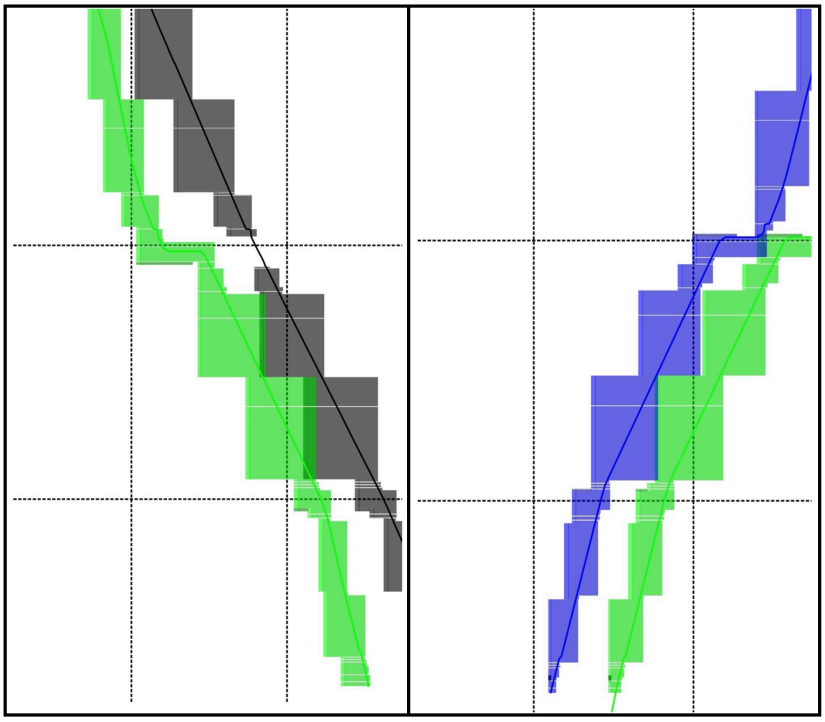


Figure 5.1: Zoom of the second scenario, between 8:00 and 8:00 on the Morp-Kp segment. *On the left:* trains running to Morp. *On the right:* trains running to Kp.

The third scenario has been simulated without any buffer times added to the headway constraints. However, the microscopic timetable does not present any conflict and the blocking times do not overlap (except where the model suggests infrastructure upgrades). However, the timetable is quite compressed, and it can be argued that including enforcing the headway times with the buffer times could have led to a different optimal solution suggesting more infrastructure upgrades, such as upgrading the portion Jbk-Arb to double tracks.

5.3. Sensitivity analysis

Parameter α reflects the priority given to minimizing the operation costs. The higher α is, the more the model focuses on reducing operation costs. The sensitivity analysis presented in Figure 4.7 shows that, as expected, the construction costs increase with α. With $\alpha = 0$, the model only minimizes the construction costs without considering the operation costs, thus the model tries as much as possible to fit the train request without changing the infrastructure. This results in high operation costs and an unrealistic timetable where trains can dwell for 1 hour at the same station. On the other hand, with $\alpha = 1$, the model only minimizes the operation costs without considering the construction costs, thus, the construction costs are high. This case is presented in Figure 4.8 and the model suggests upgrading every segment to double tracks. From the scenario where $\alpha = 1$ (the most expensive scenario) to the scenario where $\alpha = 0.5$, the construction costs are decreased by 67% while the operation costs only increase by 11%. Moreover, Figure 4.7 shows that from approximately $\alpha = 0.25$, the operation costs of the optimal solution are slowly diminishing, but quite stable. These results highlight the need to find the proper value of alpha to assess different scenarios. In other words, the parameter α reflects the time scale: if one wishes to investigate a long-term scenario with high demand, priority should be given to minimizing the operation costs. If one wishes to investigate short or middle-term scenarios with a reduced budget and time to build new tracks, the minimization of operation costs should be relaxed. These results demonstrate the flexibility and the strength of the model.

5.4. Computational efficiency

The first scenario is obtained in 12 minutes, while the second and the third are obtained in at least one hour. These CPU are high but align with other PESP models' running times found in the literature. The long-computer running and the large number of iterations reflect on the complexity of the problems. From scenario 1 to scenario 2, only the frequency of the two intercity trains is doubled. This results in an increase of 4% in the number of variables, of 255% in the number of constraints, of 529% in the number of iterations and in an increase of 682% of the CPU times. The computer running times increase exponentially with the complexity of the problem (number of variables and constraints), which depends on the number of trains to be scheduled and their interactions.

However, the third scenario has been obtained in shorter running times than the second one. This can be explained by the relaxation of the buffer times. Even though it does not reduce the number of variables or constraints, it makes it easier to find a feasible solution. Moreover, the parameter α changes between these two scenarios. Table 4.7 shows the different computer running times obtained for every simulation of the sensitivity analysis presented in 4.7. There is no linear relation between the CPU and α. Instead, the results suggest the presence of two local maximal values around $\alpha = 0.25$ and $\alpha = 0.75$, indicating that a tight trade-off increases the CPU. The optimal solution in the case where $\alpha = 0$ is fast to find because the model only minimizes the construction costs. Therefore the optimal solution is the one without any infrastructure upgrade (for the second scenario, at least) and the model does not need to minimize the operation costs.

5.5. Scalability and flexibility

The case study previously investigated considers a simple corridor only. However, a major strength of the PESP models applied to railways is that they can be used in nodded infrastructure networks without specific adjustments. In that case, the user has to specify the exact path of each line and choose the corridor that has to be plotted in the time-distance diagram. In the same way, it can be applied to more complex train requests: increased frequencies, and more dependencies between different lines with turnarounds or transfers. Nevertheless, the main limit to these extensions is the CPU. As discussed in Section 5.4, the larger the network and the more complex the demand, the more events and activities are created. Thus, complex networks might lead to long computer running time. This issue could be tackled by adjusting the different parameters such as α, β, and γ: if the demand is expected to be hardly feasible in the current state of the infrastructure, α should be increased to relax the construction cost minimization.

Moreover, the model is flexible: the infrastructure and the demand can be easily modified and do not require a lot of data. Users could also use that model to assess the effect of additional stations or lines that do not exist in the current state of the infrastructure. This can be extremely useful, especially in creating meeting stations: interlocking areas with additional tracks to allow trains to overtake each other, without passengers being able to enter or exit the train. The model can handle mixed traffic: slow and fast trains, and direct or stopping trains. Additionally, the model is not restricted to the Swedish market and it could be easily used on other railway networks with different regulations: running time supplements, headway times, or buffer times.

5.6. Innovations of the model

By extending the PESP model to a MILP formulation allowing infrastructure modification, this model goes beyond beyond the current line planning methods. The demand is not considered as flexible and cannot be relaxed by canceling trains. On the contrary, it modifies the infrastructure to ensure every train request can be scheduled without any conflict. The model does not simply suggest the construction of new tracks on the bottleneck portion of a line. Instead, the model explores all possible infrastructure improvements (station and segment) and minimizes the overall construction and operations costs. Thus, the optimal infrastructure improvement is not necessarily the cheapest one: it is the one with the best trade-off between its costs and its benefits on the traffic (on the operation costs).

The model extends the conflict detection approach developed by Zhang and Nie (2016) and instead of restricting or forbidding these conflicts, allows a limited number and type of conflicts on a given location, according to the number of tracks of this location. Therefore, the model does not only take a periodic event-activity graph as input: it also takes the sets of every possible combination of conflicts.

With this model, the assessment of different infrastructure scenarios is automatized and optimized, significantly enhancing the efficiency of planning processes. Line planners no longer need to manually evaluate each potential configuration. For instance, the simple Jbk-Kba corridor investigated in the case study can be upgraded in many ways. The segment Arb-Vsg can be upgraded to four tracks, and the remaining four portions can upgraded to double or four tracks. Thus, there are $2^1 \times 3^4 = 162$ possible arrangement of segment. If we limit the stations to a maximum of four tracks, station Kp has three tracks and can be upgraded to three or four tracks, the five other stations with two tracks can each be upgraded to 3, or 4 tracks. Thus, there are $2^1 \times 3^5 = 486$ possible configurations. By combining these two different results, the total number of configurations possible is 162 (railway corridor) $\times$ 486 (stations) $= 78,792$. The model ensures that all 78,792 potential infrastructure arrangements are assessed, facilitating thorough and efficient decision-making for infrastructure upgrades.

This innovative approach to the line planning process could be decisive in infrastructure investment decision-making, especially in a restricted budget context. Instead of allocating an entire budget to upgrading a full line, line planners could optimize the track construction and minimize the construction costs while maximizing the benefits on operation costs. Therefore, money could be saved and more lines could be upgraded. Applying this model to large networks, such as national networks to identify the lines that would require investment to meet long-term demand. For instance, it could be applied to the Swedish network (simplified to the main cities and lines) to help achieve rail traffic targets.

5.7. Limitations of the model

The case study and the different scenarios revealed that under specific situations, the model may be extremely long to solve. The activities between different train lines (transfer, headways, and regularity) make the model more complex, which could lead the model to suggest interlocking improvement, or segment improvement to four tracks.

The mathematical formulation of the model is based on several hypotheses. Firstly, platform routing at station is committed: every train can run on any platform of a station, without considering their length. In real life, some platforms are not long enough to allow some trains to dwell, and the interlockings need to be such that any train can access any platform, which is not always the case. In real life, trains turning around might use a specific track and dwell implying passenger movement can only be done if the tracks is equipped with platforms. Secondly, this model only investigates the possibility of building new tracks on existing lines. However, in line planning processes, a line can be upgraded without building a new track on its entire length: the signalling can be upgraded, usually resulting in higher speed and denser traffic. Additional tracks can be built on a small portion of the line only (meeting station), to allow trains to overtake or meet each other without upgrading an entire portion of the line to double tracks. Finally, when upgrading a segment from single to double tracks, only the benefits on conflict restriction are considered. However, because each track is reserved for one direction, the headway constraint between trains running in opposite directions should also be relaxed.

In this research, the construction costs for building a new track are limited to pure construction costs. Additional direct or indirect costs should also be considered. Direct costs could considered planning and designing costs, possible costs overruns, land acquisition costs or construction costs of new specific infrastructure such as bridges or tunnels. Nevertheless, since the model only consider upgrading line by adding one or more tracks next to a track already existing, those costs should be limited (for example, there is no need to buy new land). Secondly, the indirect effects, also called external effects, are the benefits related to the socio-economic impact of the new infrastructure on society. Since the focus of this research is given on the long term, these impacts might have consequences on the improvement of a line. For example, the long term modal shift and its consequences on carbon emission,

noise pollution or environmental impacts due to the perturbation of the local flora and fauna could have been investigated. However, to put things into perspective, given that our case study only considers a small section of railway, global socio-economic impact on the model shift is likely negligible. Moreover, indirect construction costs could also consider the duration of the work construction and the perturbation costs resulting from it. Indeed, work constructions on a line usually limit the use of the infrastructure and therefore increase the operations costs of the train running on it. Finally, additional maintenance costs of the new tracks should be taken into account. Operation costs only consider the running costs, without considering additional costs such as delays or train cancellation costs. However, these costs are quite high. Thus, further studies could also consider possible delays resulting from the produced timetable and their costs. However, this would require further investigation of the timetable at a microscopic level.

Finally, the research has been limited to train scheduling problem, but other aspects of the railways operations might have an effect on the results. For example, passenger flow impacts the operations costs: the more the train is full, the more benefits for the railway operator. The capacity of the different trains could also be considered.

5.8. Data collection of the cast study

The traffic data and traffic forecast made by the Transport Administration are quite limited. Indeed, only the number of trains running between 06:00-18:00, 18:00-22:00 and 22:00-06:00 are given, thus, this research considered the traffic to be evenly spread over the day. But in real life, the passenger demand is higher during morning and evening peak-hours, thus the number of train running these peak-hours is also higher. Since the model aims to check whether the current infrastructure could fit the long-term demand, the focus should be given in testing the highest demand possible, i.e. during peak-hours. Finally, the construction costs have been overestimated since they have been computed in the case of double-track new high speed line. However, the line of the investigated case study is a conventional line and is only an improvement: tracks are built next to already existing tracks and therefore do not require land acquisition or construction of new bridges or tunnels.

5.9. Future works

Future works could investigate the possibility of allowing the model to create additional meeting stations. A meeting station is a small area with additional tracks that allow trains to wait to allow other trains to overtake another one. This has the advantage of having similar benefits to upgrading the entire line but with a lower budget. However, this required symmetric timetables to ensure trains meet at the same area, usually in the middle of the segment.

Moreover, future improvements could relax headway constraints depending on the number of tracks on the segment. On a single track, headway constraints must concern both trains running in the same and opposite directions. However, if opposite trains are running on (at least) double tracks, then each track has a reserved direction and trains cannot meet each other. In that case, one train can leave a station at the same time as the other arrives at that station, because they do not run on the same tracks. Since the infrastructure can be updated, the headway constraints between trains running in opposite directions should be satisfied or not, depending on the number of tracks. This relaxation has not been investigated in this research and would require additional binary variables indicating whether or not the headway constraints have to be satisfied. This improvement could lead to longer CPU, but would lead to the possibility of allocating more trains on double tracks and could therefore increase the benefits of upgrading a single track to double tracks.

Finally, future works could also investigate the possibility of introducing symmetry in the formulation. Liebchen (2004) shows that introducing symmetry in the PESP formulation results in lower CPU but in poorer objective values. By reducing the CPU, the model could be implemented in larger networks.

6

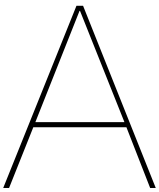
Conclusion

This research aimed to develop a macroscopic timetabling model optimizing the construction of new tracks on a given railway network to ensure the feasibility of the long-term demand. When investigating short-term scenarios, the demand is usually adapted to the infrastructure. If the demand exceeds the capacity of the infrastructure, it is reduced by canceling and re-routing conflicting trains. However, when investigating long-term scenarios, the problem should be considered from another perspective: the infrastructure should be modified to ensure it will meet predicted demand. To do so, this research applies the PESP model to railway activities and formulates it as a MILP. Formulations currently used in the field do not consider the possibility of having a flexible number of tracks on the different segments and stations of a corridor. This research therefore extends the conflicts restriction formulation proposed by Zhang and Nie (2016) by introducing flexible numbers of tracks and optimizing the cost-benefit trade-off of these upgrades by minimizing both the total construction costs and the operation costs of the timetable.

The model's strengths are demonstrated by investigating different demand scenarios on a portion of the Mälarbanan line, located northwest of Stockholm. When simulating the predicted 2040 demand, results show that upgrading only one small portion of the corridor to double tracks would ensure a feasible timetable, leading to 67% cut in construction costs while increasing the operation costs by only 11% (compared to the case where the entire corridor is upgraded to double tracks). Therefore, the model goes beyond the current line planning methods and does not simply suggest new tracks' construction on the bottleneck portion of a line. Instead, the model explores all possible infrastructure improvements and minimizes the overall construction and operations costs. When an infrastructure upgrade is suggested, the model maximizes the benefits of these modifications by presenting the greatest number of conflicts on the upgraded line portion. This extension of the PESP model offers a new approach to construction cost minimization in railway line planning.

While the model demonstrates promising results, further testing is required under more complex operating conditions such as interactions between the different train lines (transfers, turnarounds) or higher frequencies. The practicality of the solution proposed by the model could also be tested to check whether the suggested infrastructure improvements are technically feasible. Moreover, this model approximates the various operation and construction costs and does not fully capture the real-world complexity of railway operation and planning costs. Further development of the model could investigate the possibility of introducing flexible headway constraints according to the number of tracks of the different portions of the network. The potential for building additional meeting stations on a line could also be investigated and should improve the model's solutions by reducing the construction costs. Moreover, further research should investigate the robustness of the timetable produced by the model.

In conclusion, this research advances strategic and efficient line planning to meet the increasing demand for railway services. The model developed for this research offers an innovative approach to both conflict management and infrastructure improvements, while considering the economic viability of the solution proposed. It contributes to the ongoing efforts to move forward decarbonized mobility.



Research paper

Optimizing Railway Infrastructure to Meet Future Demand: a Macroscopic Timetabling Model

Matéo Ménoury
Supervised by R.M.P. Goverde, J.A. Annema, P.S.A. Stokkink and K. Bediru Seid

***Abstract*— This paper proposes an innovative macroscopic timetabling model aiming to optimize railway infrastructure to accommodate the future increasing demand. The model applies the PESP formulation to railway operations and formulates it as a MILP model. The number of tracks on each portion of the infrastructure is flexible, and so are the conflicts between different trains. The model minimizes the overall construction and operation costs while ensuring a conflict-free timetable for the long-term demand. A case study on the Swedish Mälarbanan line demonstrates the model's application and validates the microscopic feasibility of the produced timetables. Model shows promising results by suggesting low infrastructure upgrades while offering total travel times almost equivalent to the case where the entire line is upgraded.**

Keywords: Railway timetabling, Periodic Event Scheduling Problem (PESP), Mixed Integer Linear Programming (MILP), Line planning, Swedish railways.

I. Introduction

Demand for railway services is increasing in Sweden. However, the infrastructure is limited, and numerous lines are still constituted of single tracks only, limiting the capacity of the network. Concerns are rising on whether the current state of the infrastructure will be able to meet future demand. Long-term line planning is fundamental to investigating the bottleneck portions of a line that should be upgraded. However, upgrading an entire portion of the line requires high investments. Railway timetables organize different train movements to ensure safe and predictable traffic. Usually, timetables are simulated on a short-term scale and the demand is therefore adapted to ensure it does not exceed the current infrastructure's capacity. In long-term investigation, infrastructure should be adapted to meet the demand. Nevertheless, limited research has addressed timetabling models allowing infrastructure modifications, leaving a critical gap in our understanding of macroscopic timetabling. This research aims to develop a macroscopic timetabling model optimizing the construction of new tracks on a given railway network, to ensure the feasibility of the long-term demand. To do so, the number of tracks is considered flexible and can be increased to find the optimal trade-off between the construction costs required for the infrastructure upgrades and their benefits on the operation costs.

II. Current timetabling models

A. PESP applied to railway operations

The Periodic Event Scheduling Problem (PESP) [1] constitutes a systematic approach used as the basis of most periodic activities scheduling problems. The PESP is based on a periodic event-activity directed graph $G = (E, A)$ in which the nodes $x \in E$ depict the arrival and departure events and the arcs $a = (x, y) \in A$ represent the activities between those events with interval constraints. Those activities can be running, dwelling, turnaround, passing-through, transfer, headways, or regularity.

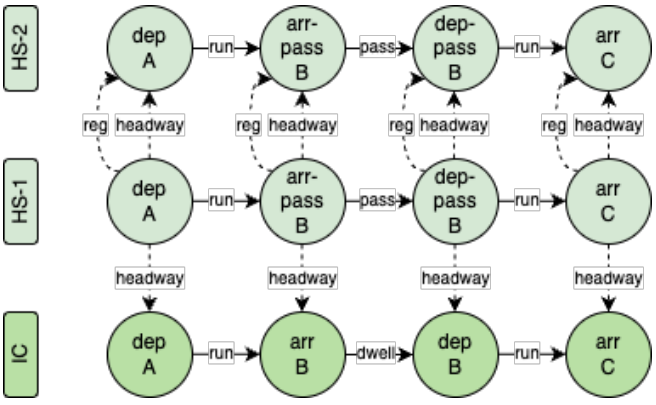


FIG 1. Event-activity graph. A, B and C are three different stations.

$$d_{x,y} = (v_y - v_x) + p_{x,y} \cdot \Pi \qquad \forall a = (x, y) \in A \qquad (1)$$

Constraint (1) defines the duration of activity $a = (x, y)$ as the time difference between the end time v_y and the start time v_x [sec]. If the activity is spread over two periods, i.e. if $p_{x,y} = 1$, then the period Π [sec] is

added to the time difference $(v_y - v_x)$ to ensure that the duration is always a positive value.

$$p_{x,y} = \begin{cases} 1 & \text{if } v_y < v_x \\ 0 & \text{otherwise} \end{cases} \qquad \forall a = (x,y) \in A \qquad (2)$$

Linear formulation of (2):

$$v_y - v_x \leq -\epsilon + M \cdot (1 - p_{x,y}) \qquad \forall a = (x,y) \in A \qquad (3)$$

With M a sufficiently large value

Constraints (2) defines the value of the binary variable $p_{x,y}$. In this model, it is restricted to a binary variable since the upper bound of every activity is lower than the period Π. It takes the value 1 if event y is scheduled before event x, and 0 otherwise. In other words, the modulo operator indicates whether the activity points at an event in the same or next period than the origin event.

$$L_{x,y} \leq d_{x,y} \leq \Pi \qquad \forall a = (x,y) \in A \backslash (A_{\text{headway}} \cup A_{\text{reg}}) \qquad (4)$$

$$L_{x,y} \leq d_{x,y} \leq \Pi - L_{x,y} \qquad \forall a = (x,y) \in A_{\text{headway}} \qquad (5)$$

$$d_{x,y} = \frac{\Pi}{fre_{x,y}} \qquad \forall a = (x,y) \in A_{\text{reg}} \qquad (6)$$

Constraints (4) and (5) ensure that the duration of all run, dwell, pass-through, transfer, turnaround, and headway activities a are bonded between a maximal and minimal duration $L_{x,y} > 0$ [sec]. Constraint (6) ensures that the duration of the regularity activity is fixed to the period of the corresponding line, based on its frequency $fre_{x,y}$ [unit/period]. This ensures that every event of the different copies of the same line occurs at consistent, evenly-spaced intervals.

The PESP is a feasibility problem: if a feasible solution does not exist, the user can then relax one of the constraints and run the model again. An objective function can be added on top of it. Then, the algorithm aims to find the optimal solution according to the objective function.

B. Flexible overtaking constraint

A feasible timetable must be conflict-free: two trains running on the same track can neither overtake each other nor meet each other. [2] extends the PESP problem and proposes linear constraints to avoid or allow overtaking. The research describes all possible scenarios and analyses the behavior of the different modulo operators $p_{x,y}$. When two activities $a = (x,y)$ and $a' = (x',y')$ imply two trains running on the same segment, their departure and arrival events are constrained by headway constraint (see FIG 1). Thus, four modulo operators can be obtained: $p_{x,y}$, $p_{x',y'}$, $p_{x,x'}$ and $p_{y,y'}$. The authors show that the sum of those modulo operators equals zero, two, or four when overtaking is prevented, and equals to one or three otherwise. In other words, if the sum is even, no conflict is detected and if the sum is odd, a conflict is detected. This paper extends this approach and formulates the following constraint that detects the presence or not of a conflict between two activities $a = (x,y)$ and $a' = (x',y')$ on a given track.

$$q_{(a,a')} = \begin{cases} 1 & \text{if } (p_{x,y} + p_{x',y'} \\ & \quad + p_{x,x'} + p_{y,y'}) \\ & \quad \equiv 0 \pmod 2 \\ 0 & \text{otherwise} \end{cases} \qquad \forall (a,a') \qquad (7)$$

Constraint (7) defines the binary variable q which takes the value 1 if the activities $a = (x,y)$ and $a' = (x',y')$ are conflicted, and zero otherwise. The conflict detection developed for our research presents two differences from the one proposed by [2]. Firstly, our research also detects the collision, i.e. the conflict between two trains running in opposite directions. To do so, the model simply inverses the direction of the train and applies the same constraint previously introduced. Secondly, in our model, the conflicts are only detected and counted per segment, and the model is free to create or not conflict.

III. Cost-minimization model

A. New sets and decision variables

Unlike other PESP approaches, this model introduces a flexible conflict restriction and infrastructure optimization. The network is depicted in a graph $H = (I,S)$ in which the nodes $i \in I$ are the interlockings (stations or meeting stations) and the arcs $s \in S$ represents the segments (lines) between the different interlockings. Each segment s and interlocking i has an initial number of tracks t_s^{before} and t_i^{before}. The location index $l \in (I \cup S)$ is also introduced to simplify the notation.

FIG 2: Macroscopic representation of a railway network.

B. Objective function

The objective function (8) is written as a minimization of total costs by minimizing the sum of three different components: operation costs, construction

costs of segments, and construction costs of interlockings. These different costs have different orders of magnitude. Thus, the sum cannot be minimized as such, otherwise, the model would focus too much on reducing the construction costs. The costs are normalized using the min-max normalization (9). Additionally, the different coefficients α, β, and γ such that $\alpha + \beta + \gamma = 1$ reflect on the priority to give to the optimization.

$$\min \quad \alpha \cdot c_{\text{operation}}^{\text{normalized}} + \beta \cdot c_{\text{segment}}^{\text{normalized}} + \gamma \cdot c_{\text{interlocking}}^{\text{normalized}} \tag{8}$$

With:

$$c^{\text{normalized}} = \frac{c - c^{\min}}{c^{\max} - c^{\min}} \tag{9}$$

$$c_{\text{operation}} = \sum_{a=(x,y)\in A} c_{x,y} \cdot d_{x,y} \tag{10}$$

$$c_{\text{interlocking}} = \sum_{i\in I} c_i^{\text{new track}} \cdot (t_i^{\text{after}} - t_i^{\text{before}}) \tag{11}$$

$$c_{\text{segment}} = \sum_{s\in S} c_s^{\text{new track}} \cdot (t_s^{\text{after}} - t_s^{\text{before}}) \tag{12}$$

The total operation costs (10) are computed as the sum of every activity duration $d_{x,y}$ [sec] multiplied by its cost $c_{x,y}$ [SEK/sec]. The minimum possible operation costs correspond to the situation where every activity is scheduled according to their minimum duration time $L_{x,y}$. The maximum possible operation costs correspond to the situation where every activity duration is maximal, i.e. if every activity duration equals Π.

The total construction costs of interlocking (11) and segments (12) are computed as the sum of the costs of every track that needs to be built on every interlocking or segment. $c_i^{\text{new track}}$ and $c_s^{\text{new track}}$ [SEK/track] are the costs for building one new track in a specific segment s or i. The difference (t_s^{after} - t_s^{before}) gives the number of tracks to be constructed at segment s. The same goes for interlocking i. The minimum possible construction costs are zero and corresponds to the situation where no interlocking is upgraded. The maximum possible construction costs correspond to the situation where every new track $t^{\max}$ is constructed at the most expensive interlocking to upgrade.

The objective function (8) is subject to the basic PESP constraints (1), (2), (4), (5), and (6) previously introduced, and to new constraints that are defined in the following sections.

C. Conflict detection

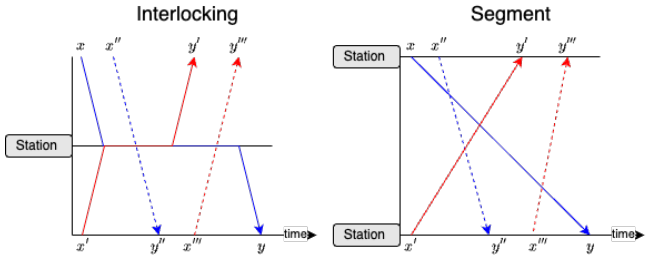
The conflicts must first be detected. To do so, different sets C_l^k of all combinations of k activities taking place at location l: these activities are susceptible to be conflicted. A combination refers to a selection of activities a where the order of the different activities does not matter: the combination (a, a') is equivalent to the combination (a', a) (see FIG 3). A conflict between two activities is detected using the formulation defined in constraint (7). Each set of two activities C_l^2 is associated with a binary variable $q_{c,l}$ indicating whether the two activities $a = (x, y)$ and $a' = (x', y')$ are conflicted.

$$q_{c,l} = \begin{cases} 1 & \text{if } (p_{x,y} + p_{x',y'} \\ & +p_{x,x'} + p_{y,y'}) \\ & \equiv 0 \pmod 2 \\ 0 & \text{otherwise} \end{cases} \quad \begin{matrix} \\ \forall c = (a, a') \\ \forall l \in (S \cup I) \end{matrix} \tag{13}$$

Linear formulation of (13):

$$\begin{matrix} q_{c,l} = (p_{x,y} + p_{x',y'} + \\ p_{x,x'} + p_{y,y'}) - 2 \cdot z_1 \\ \text{With } z_1 \text{ an integer variable} \end{matrix} \quad \begin{matrix} \forall c = (a, a') \\ \forall l \in (S \cup I) \\ \\ \end{matrix} \tag{14}$$

Nevertheless, detecting conflicts between two different trains is not enough. Indeed, depending on the number of tracks of a given location l, it can be acceptable to have more than two trains sharing the same infrastructure at the same time. Therefore, conflicts between more than two trains also have to be detected. To do so, each combination of more than two activities $c = (a, a', a'', ...)$ is associated with a binary variable $q_{c,l}$ indicating whether each activity a from the combination c is conflicted with every other one. The two examples above show the conflict detection among a set of activities A: one applied to an interlocking i (left diagram) and one applied to a segment s (right diagram).



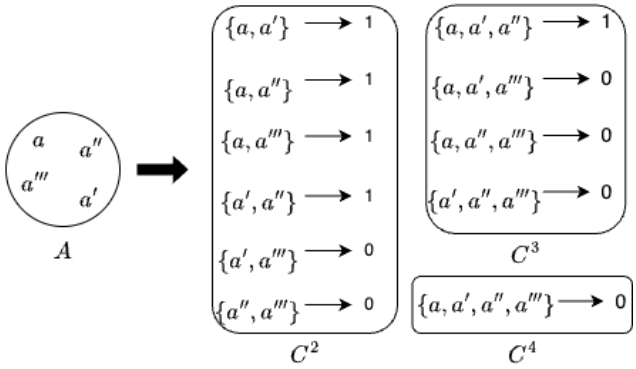


FIG 3: Possible sets of conflicts given a set of activities A. Each combination is associated with a binary variable.

The binary variables associated with the different sets C^3 and C^4 are calculated based the binary variables associated with the set C^2, as defined by constraint (13). In the example above, every activity a, a', and a'' is conflicted with the two others. Therefore, the some of the 3 binary variables $q_{(a,a')(a,a'')} + q_{(a,a')(a',a'')} + q_{(a,a'')(a',a'')} = 3$, thus, the binary variables $q_{(a,a',a'')}$ takes the value 1. The example is generalized with the formulation as follows.

$$q_{c,i} = \begin{cases} 1 \quad \text{if} \sum\limits_{\substack{c^* \in C_i^2 \\ c^* \in c}} q_{c^*,i} = \binom{k}{2} & \forall c \in C_i^k \\ 0 \quad \text{otherwise} & k > 2 \\ & \forall i \in I \end{cases} \tag{15}$$

Linear formulation of (15):

$$\begin{aligned} \sum_{\substack{c^*=(a,a') \in C_i^2 \\ a,a' \in c}} q_{c^*,i} &\leq \binom{k}{2} + \epsilon + M \cdot q_{c,i} & \forall c \in C_i^k \\ & & k > 2 \\ \sum_{\substack{c^*=(a,a') \in C_i^2 \\ a,a' \in c}} q_{c^*,i} &\geq \binom{k}{2} - M \cdot (1 - q_{c,i}) & \forall i \in I \end{aligned} \tag{16}$$

Constraint 15 defines the binary variable $q_{c,i}$. It takes the value 1 if every activity of the combination c is conflicted with every other. To do so, every possible pair of conflicts among the combination c is checked. If the sum of their binary variables $q_{c^*,i}$, computed in constraint 13 equals the number of possible pairs, then every activity is conflicted with each other. The number of possible pairs among a combination c of k activities equals $\binom{k}{2} = \frac{k!}{2!(k-2)!} = \frac{k(k-1)}{2}$.

D. Minimum number of tracks required

The conflicts and the number of tracks required on the network are interdependent. t_s^{needed} and t_i^{needed} are integer variables indicating the minimum number of tracks required to resolve detected conflicts, ensuring that the demand fit within the infrastructure without causing any collision. The formulation differs depending on the type of infrastructure: segment or interlocking.

On a segment s, the rule is: at any moment time, there cannot be conflicts on the same track. Therefore, constraint (17) defines the minimum number of tracks needed at segment s to ensure that the timetable is conflict-free, based on the number of conflicts detected. Each segment can have one, two, or four tracks. On a single-track corridor, trains can run in both directions. Therefore, a segment needs at minimum one track if no trains are running in opposite directions at the same moment and no trains are overtaking each other. On a double-track corridor, each track is reserved for one direction. Therefore, a segment needs at minimum two tracks if at least two trains are running in opposite directions but no train is overtaking another one. Finally, on a four-track corridor, we consider that two tracks are used for each direction, which allows overtaking. A segment needs at minimum four tracks if at least one train is overtaking another one running in the same direction. For example, the situation presented on the right diagram of Figure 3 requires four tracks because one train running in the blue direction overtakes another one running in the same direction.

$$t_s^{\text{needed}} = \begin{cases} 1 \quad \text{if } k_s^{\text{opp}} = 0 \text{ and } k_s^{\text{same}} = 0 \\ 2 \quad \text{if } k_s^{\text{opp}} = 1 \text{ and } k_s^{\text{same}} = 0 \\ 4 \quad \text{if } k_s^{\text{same}} = 1 \end{cases} \quad \forall s \in S \tag{17}$$

$$k_s^{\text{opp}} = \begin{cases} 1 \quad \text{if} \sum\limits_{\substack{c=(a,a') \in c_s^2 \\ dir_a \neq dir_{a'}}} k_{c,s} \geq 1 \\ 0 \quad \text{otherwise} \end{cases} \quad \forall s \in S \tag{18}$$

$$k_s^{\text{same}} = \begin{cases} 1 \quad \text{if} \sum\limits_{\substack{c=(a,a') \in c_s^2 \\ dir_a = dir_{a'}}} k_{c,s} \geq 1 \\ 0 \quad \text{otherwise} \end{cases} \quad \forall s \in S \tag{19}$$

Linear formulation of (17):

$$\begin{aligned} & z_2 \leq k_s^{\text{opp}} \\ & z_2 \leq k_s^{\text{same}} \\ & z_2 \geq k_s^{\text{opp}} + k_s^{\text{same}} - 1 \\ & z_3 = 1 - k_s^{\text{same}} - k_s^{\text{opp}} + z_2 \\ & z_4 = k_s^{\text{opp}} - z_2 \\ & t_s^{\text{needed}} = z_3 \cdot 1 + z_4 \cdot 2 + k_s^{\text{same}} \cdot 4 \\ & \text{With } z_2, z_3, z_4 \text{ binary variables} \end{aligned} \quad \forall s \in S \tag{20}$$

Constraints (18) and (19) define the help binary variables k which take the value 1 if there is respectively at least one overtaking (i.e. conflict between two

trains running in the same direction) and at least one collision (i.e. conflict between two trains running in opposite directions) on the segment s. The parameter $dir_a \in \{-1, 1\}$ identifies the direction of the train corresponding to the activity $a = (x, y)$.

$$k_{c,s} = 0 \qquad \begin{array}{r} \forall c = (a, a', a'') \in C_s^3 \\ \text{such that } dir_a = dir_{a'} = dir_{a''} \\ \forall s \in S \end{array} \qquad (21)$$

In any case - one, two, or four tracks - there cannot be three different trains running in the same direction overtaking each other (one slow train, one fast train, and one high-speed train for example). Indeed, that would require at least three tracks reserved for that direction. Therefore, constraint (21) ensures that on each segment, there is no combination of three different activities a, a' and a'' having the same direction such that each of them has a conflict with the two others. However, one train is still allowed to overtake multiple slower trains on the same segment, and one train can be overtaken by multiple faster trains.

On an interlocking i, the rule is: at any moment time, there cannot be more train than tracks. Here, the direction of the train does not matter and any train can dwell or pass on any track of a station.

$$t_i^{\text{needed}} = \max_{c \in C_i}(q_{c,i} \cdot len(c)) \qquad \forall i \in I \qquad (22)$$

Linear formulation of (22):

$$t_i^{\text{needed}} \geq q_{c,i} \cdot len(c) \qquad \forall c \in C_i, \forall i \in I \qquad (23)$$

Constraint (22) defines the minimum number of tracks needed at interlocking i to ensure that the timetable is conflict free, based on the number of conflicts detected. Each interlocking can have as many tracks as needed, computed as the maximal number of trains running, passing, dwelling or turning around at an interlocking at the same time. To do so, each possible combination of activities c taking place at interlocking i is checked. Among the combinations for which $q_{c,i} = 1$, the combination with the larger number of activities, given by its length $len(c)$, is set to be the minimum number of tracks needed. As a reminder, $q_{c,i}$ takes the value 1 if and only if each activity of c is conflicted with every other one. For example, the situation presented on the left diagram of Figure 3 requires three tracks on the station.

E. Optimal number of tracks

Based on these decision variables and the initial state of the infrastructure, t_s^{after} and t_i^{after} return the number of tracks the network should have to satisfy the demand.

$$t_l^{\text{after}} = \max\{t_l^{\text{before}}, t_l^{\text{needed}}\} \qquad \forall l \in (S \cup I) \qquad (24)$$

Linear formulation of (24):

$$\begin{array}{l} t_l^{\text{after}} \geq t_l^{\text{before}} \\ t_l^{\text{after}} \geq t_l^{\text{needed}} \end{array} \qquad \forall l \in (S \cup I) \qquad (25)$$

Constraint (24) ensures that if the initial number of tracks at a segment is lower than the minimal required number of tracks, the infrastructure is updated.

$$\sum_{l \in S \cup I} (t_l^{\text{after}} - t_l^{\text{before}}) \leq t^{\max} \qquad (26)$$

Constraint (26) restricts the number of new tracks to be constructed on the entire network using the integer parameter $t^{\max}$

IV. Results

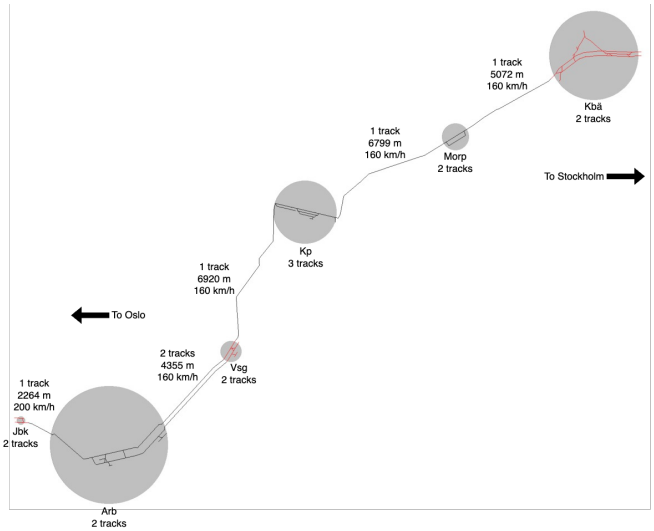


FIG 4: Jbk-Kbä corridor of the Mälarbanan.

The model is applied to a small portion of the Mälarbanan. This line is located northwest of Stockholm and demand is expected to increase. However, an important portion of the line is still constituted of single tracks. This portion of the line is tested with the designed model to assess whether it could accommodate the 2040 predicted demand. The minimum technical running times are directly simulated within RailSys.

TABLE I

VARIOUS INPUT DATA

PARAMETERS APPLIED [3]	
Min running time supp	8% of the min technical running time
Headway time	3 min
Buffer time	1 min, added to the headway times
Min dwell time	2 min
α	0.5
β	$\alpha/2$
γ	$\alpha/2$
DEMAND 2040 [4] AND OPERATION COSTS [5]	
Freight	1/h-direction
	no mandatory stop
	4.31 SEK/sec
Intercity	1/h-direction
	1 stop at Kp
	4.53 SEK/sec
Regional	2/h-direction
	2 stops at Kp and Vsg
	9.17 SEK/sec
CONSTRUCTION COSTS (IN MILLION SEK) [6]	
Cost per segment	373 / track km
Jbk-Arb	844
Arb-Vsg	1,624
Vsg-Kp	2,581
Kp-Morp	2,536
Morp-Kba	1,892
Cost per interlocking	0.2 / platform metre
Jbk	25
Arb	51
Vsg	25
Kp	64
Morp	25
Kba	30

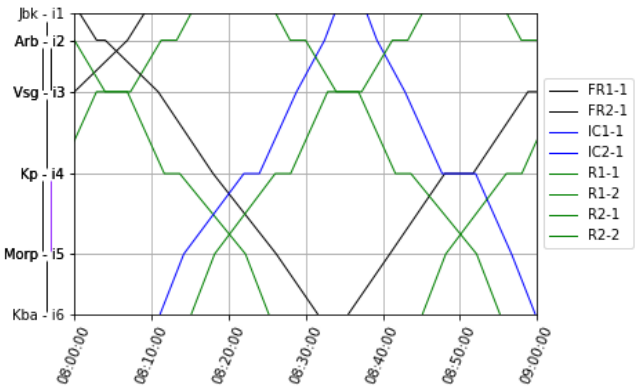


FIG 5: Optimal time-distance diagram obtained.

TABLE II

RESULTS OF THE MODEL

Infrastructure to upgrade	Kp-Morp to double tracks
Total operation costs	72,022 SEK
Total construction costs	2,536 million SEK
Objective value	0.242126
Number of variables	1,770 (incl. 1,062 binary)
Number of constraints	3,384
Number of iterations	522,390,696
CPU	1h 37min
Optimally gap	4.9%

The simulation model produces timetables feasible at the macroscopic level. However, their feasibility at the microscopic level is not guaranteed since several hypotheses have been made and the detailed signalling of the infrastructure has not been considered. Thus the timetables produced are simulated in RailSys to assess their feasibility at the microscopic level.

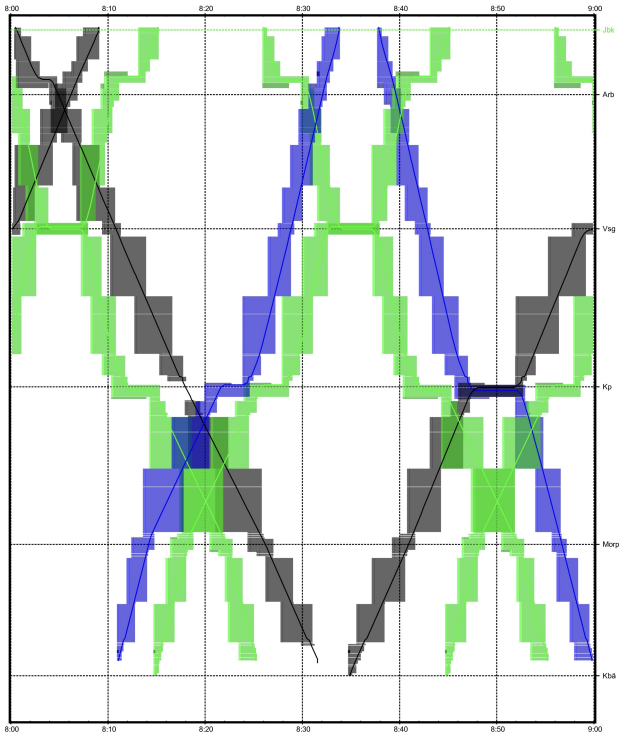


FIG 6: Microscopic simulation of the time-distance diagram given in Figure 5.

To assess the sensitivity of the optimal solution to that parameter, the same scenario has been tested with different values of α, keeping all the other parameters identical.

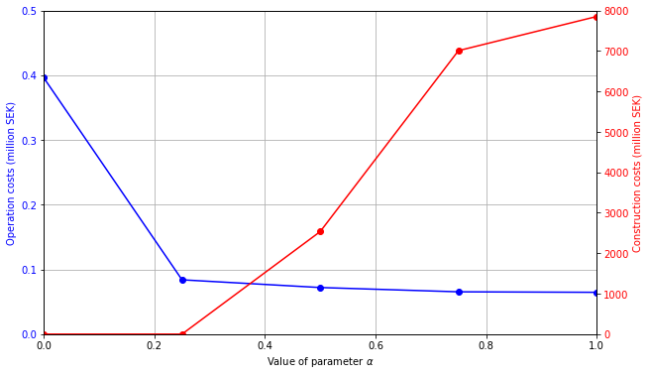


FIG 7: Operation and construction costs of the optimal solution, given various values of α.

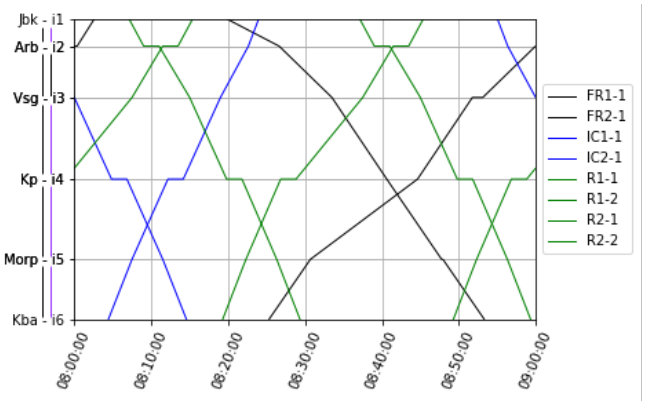


FIG 8: Optimal time-distance diagram obtained with $\alpha = 1$

V. Discussion

The results show that the model can handle the key constraints: the headway times, minimum running times, dwell times, and regularity constraints are all satisfied and no conflict is allowed where the infrastructure is only constituted of a single track. Since the buffer times have been introduced within the headway times, the model has to schedule a timetable with high headway times. The timetables are all Π-periodic. The current state of the infrastructure would not be able to meet the 2040 projected demand.

Freight trains have unsolicited stops at stations Arb and Vsg. This demonstrates that the model manages to prioritize train operations regarding their costs. This demonstrates the trade-off the model has to optimize: upgrading the infrastructure to avoid these additional stops and to lower as much as possible operations costs is not necessarily optimal. Seven different conflicts take place on the Kp-Morp portion of the line, which is upgraded to double tracks. This portion is the most expensive one to upgrade, but also the longest one and therefore the one having the most benefits on traffic. Therefore, if the model suggests an infrastructure upgrade, the optimal timetable takes as much as possible advantages of the new construction. The optimal infrastructure of the second scenario is a corridor of single - double - single - double - single tracks. Thus, this result also suggests that it is more efficient to spread the double tracks all along the corridor rather than having a long portion of double tracks. Indeed, this allows to have two distinct areas where opposite trains can meet each other.

Figure 8 shows that at the microscopic level, conflicts are detected between trains running in opposite directions. This aligns with our expectations since the infrastructure has not been upgraded in RailSys, and is therefore still composed of single tracks, even on the portions suggested to be upgraded to double tracks. Therefore, these conflicts can be omitted in this research. On the corridor Morp-Kp, blocking times of the regional and intercity trains overlap for a minute on two block sections of the corridor. The same goes for the regional and intercity trains whose blocking times overlap for a few seconds in one block section. This can be explained by the length of these two blocks which are longer than the other blocks of the corridor. The longer the block length, the longer the blocking time. Nevertheless, these three conflicts are minimal and could be easily resolved at the microscopic level by slightly re-dispatching one or the other train, to ensure their blocking times do not overlap. At the macroscopic level, these conflicts could be avoided by reinforcing the headway constraint on the Kp-Morp portion of the corridor, by increasing its value to four minutes (instead of three on the other block sections).

Figure 7 shows that, as expected, the construction costs increase with α. The higher α is, the more the model focuses on reducing operation costs. With $\alpha = 0$, the model only minimizes the construction costs without considering the operation costs, thus the model tries as much as possible to fit the train request without changing the infrastructure. This results in high operation costs and an unrealistic timetable. On the other hand, with $\alpha = 1$, the model only minimizes the operation costs without considering the construction costs, thus, the construction costs are high. This case is presented in Figure 8 and the model suggests upgrading every segment to double tracks. Compared to the scenario where the entire corridor is upgraded to double track (Figure 8, $\alpha = 1$), upgrading only one small portion of the corridor to double tracks (Figure 5, $\alpha = 0.5$) would ensure a feasible timetable, leading to 67% cut in construction costs while increasing the operation costs by only 11%. These results highlight the need to find the proper value of alpha to assess different scenarios. In other words, the parameter α reflects the time scale: if one wishes to investigate a long-term scenario with high demand, priority should be given to minimizing the operation costs. If one wishes to investigate short-middle term scenarios where with a reduced budget and time to build new tracks, the minimization of operation costs should be relaxed. These results demonstrate the flexibility and the strength of the model.

VI. Conclusion

This research introduces the possibility of building new tracks on a railway network and optimizing the cost-benefit trade-off of these upgrades by minimizing both the total construction costs and the

operation costs of the timetable. The model extends the flexible conflicts restriction to a flexible state of the infrastructure: when an infrastructure upgrade is suggested, the model maximizes the benefits of these modifications by presenting the greatest number of conflicts on the upgraded line portion. Results demonstrated the model's strengths: the model strongly reduces construction costs while maintaining almost minimal operation costs. Therefore, the model goes beyond the current line planning methods and does not simply suggest the construction of new tracks on the bottleneck portion of a line. Instead, the model explores all possible infrastructure improvements and minimizes the overall construction and operations costs. Thus, this extension of the PESP model offers a new approach to construction cost minimization in railway line planning.

REFERENCES

[1] P. Serafini and W. Ukovich, "A Mathematical Model for Periodic Scheduling Problems," *SIAM J. Discret. Math.*, vol. 2, pp. 550-581, 1989.

[2] X. Zhang and L. Nie, "Integrating capacity analysis with high-speed railway timetabling: A minimum cycle time calculation model with flexible overtaking constraints and intelligent enumeration," *Transportation Research Part C: Emerging Technologies*, vol. 68, pp. 509-531, 2016.

[3] E. Solinen, *Generella konstruktionsregler för tågplanekonstruktion*, version 1.0, Trafikverket, Röda vägen 1, 781 70 Borlänge, 2022, ISBN: 978-91-7725-920-6, no. 2021:163.

[4] Trafikverket, "Trafikuppgifter jarnvag T22 och bullerprognos 2040," 2021.

[5] Trafikverket, "Network Statement 2024," Edition 2023-12-15. For deliveries from 2023-12-10 to 2024-12-14, December 2023.

[6] Trafikverket and Jacobs, "New Main Lines: Cost Benchmarking Study," Trafikverket and Jacobs, Tech. Rep., March 2021.

B

Code

B.1. Functions

B.1.1. Processing functions

Create event

```
def create_event (train_request, interlocking, period):

    event = pd.DataFrame(columns=['eventID', 'lineID', 'event_type', 'event_location', '
        event_frequency']) # create empty table

    for idx, row in train_request.iterrows(): # iterate for each line requested
        baselineID = idx # the lineID without considering the different copy
        path = row['path'] # list
        stop_stations = row['stop_stations'] # list
        event_frequency = math.ceil(row['frequency'] * (period/60)) # number of copy of the
            lineID events per period

        for copy in range((event_frequency)): # iterate for each copy of the lineID
            lineID = idx + '-' + str(copy+1) # the new lineID considering the number of copy
                of each line

            for i, interlockingID in enumerate(path): # iterate for each interlocking in the
                path of the line
                station = interlocking.loc[interlockingID, 'station_code'] # station code of
                    the interlockingID

                if i == 0: # create only one row for the departure station
                    event_type = 'dep'
                    event.loc[len(event)] = ['e' + str(len(event) + 1), lineID, event_type,
                        interlockingID, event_frequency]

                elif i == len(path) - 1: # create only one row for the arrival station

                    event_type = 'arr'
                    event.loc[len(event)] = ['e' + str(len(event) + 1), lineID, event_type,
                        interlockingID, event_frequency]

                else: # create two rows for intermediate stations
                    event_type_arr = 'arr' if station in stop_stations else 'arr-pass'
                    event_type_dep = 'dep' if station in stop_stations else 'dep-pass'
                    event.loc[len(event)] = ['e' + str(len(event) + 1), lineID,
                        event_type_arr, interlockingID, event_frequency]
                    event.loc[len(event)] = ['e' + str(len(event) + 1), lineID,
                        event_type_dep, interlockingID, event_frequency]

    event.set_index('eventID', inplace=True) # set the 'eventID' as index

    return (event)
```

Create activity

```
def create_activity(train_request, interlocking, segment, event, rolling_stock, regulation,
    min_technical_running_times):

    activity = pd.DataFrame(columns=["activityID", "from_eventID", "to_eventID", "
        activity_type", "activity_location", "min_activity_duration", "activity_cost", "
        train_direction"]) # create empty table

    for idx, row in train_request.iterrows():
        baselineID = idx
        rolling_stock_value = row['rolling_stock']
        interaction = row['interaction']
        with_lineID = row['with_lineID']
        at_station = row['at_station']
        cost = row['cost']
        events_in_lineID = event[event['lineID'].str.startswith(baselineID + '-')] # events
            from the same line

        # ---- Run, dwell, pass ----

        for lineID, events_in_lineID_group in events_in_lineID.groupby('lineID'): # iterate
            for each event of the same line

            for i in range(len(events_in_lineID_group) - 1): # iterate for each pair of
                consecutive events
                from_eventID = events_in_lineID_group.index[i] # first event of the activity
                to_eventID = events_in_lineID_group.index[i+1] # second event of the activity
                from_interlockingID = event.loc[from_eventID, 'event_location'] # location of
                     the first event
                to_interlockingID = event.loc[to_eventID, 'event_location'] # location of the
                     second event

                if int(from_interlockingID[1]) < int(to_interlockingID[1]): # if the train
                    goes in direction 1
                        train_direction = 1

                if int(from_interlockingID[1]) > int(to_interlockingID[1]): # if the train
                    goes in direction -1
                    train_direction = -1

                if int(from_interlockingID[1]) == int(to_interlockingID[1]): # if the train
                    is dwelling at interlocking, we look at the next event
                    if i == 0: # if first event
                        next_eventID = events_in_lineID_group.index[i+2]
                        next_interlockingID = event.loc[next_eventID, 'event_location']
                        if int(to_interlockingID[1]) < int(next_interlockingID[1]): # if the
                            train goes in direction 1
                            train_direction = 1
                        else:
                            train_direction = -1
                    else:
                        previous_eventID = events_in_lineID_group.index[i-1]
                        previous_interlockingID = event.loc[previous_eventID, 'event_location
                            ']
                        if int(previous_interlockingID[1]) < int(to_interlockingID[1]): # if
                            the train goes in direction -1
                            train_direction = 1
                        else:
                            train_direction = -1

                # ---- Pass ----

                if event.loc[from_eventID, 'event_type'] == "arr-pass" and event.loc[
                    to_eventID, 'event_type'] == "dep-pass":
                    activity_type = "pass"
                    activity_location = from_interlockingID
                    min_activity_duration = regulation.loc['pass', 'min_time']
                    activity_cost = regulation.loc['pass', 'cost'] * cost
```

```
            # ---- Dwell ----

            elif event.loc[from_eventID, 'event_type'] == "arr" and event.loc[to_eventID,
                 'event_type'] == "dep":
                activity_type = "dwell"
                activity_location = from_interlockingID
                min_activity_duration = regulation.loc['dwell', 'min_time']
                activity_cost = regulation.loc['dwell', 'cost'] * cost

            # ---- Run ----

            else:
                activity_type = "run"

                if train_direction == 1:
                    activity_location = segment.loc[(segment['from_interlockingID'] ==
                        from_interlockingID) & (segment['to_interlockingID'] ==
                        to_interlockingID)].index[0]

                elif train_direction == -1:
                    activity_location = segment.loc[(segment['from_interlockingID'] ==
                        to_interlockingID) & (segment['to_interlockingID'] ==
                        from_interlockingID)].index[0]

                min_activity_duration = (1.08/1.03) * min_technical_running_times.loc[
                    rolling_stock_value][activity_location]
                activity_cost = regulation.loc['run', 'cost'] * cost

            activity.loc[len(activity)] = ['a' + str(len(activity) + 1), from_eventID,
                to_eventID, activity_type, activity_location, min_activity_duration,
                activity_cost, train_direction]

    # ---- Transfer, turnaround ----

    if isinstance(interaction, list): # if there is at least one interaction

        for i, activity_type in enumerate(interaction): # iterate for each interation of
            the baselineID
            line_frequency = train_request.loc[baselineID, 'frequency'] # number of time
            the line is repeated

            for copy in range(int(line_frequency)):
                lineID = baselineID + '-' + str(copy+1) # the lineID considering the
                    different copy
                interaction_station = at_station[i]
                activity_location = interlocking[interlocking['station_code'] ==
                    interaction_station].index[0]
                interacted_baselineID = with_lineID[i]
                interacted_lineID = interacted_baselineID + '-' + str(copy+1) # the
                    lineID of the interacted line considering the different copy
                cost_interacted_lineID = train_request.loc[interacted_baselineID, 'cost']
                from_eventID = event[(event['lineID'] == lineID) & (event['event_location
                    '] == activity_location) & (event['event_type'] == 'arr')].index[0]
                to_eventID = event[(event['lineID'] == interacted_lineID) & (event['
                    event_location'] == activity_location) & (event['event_type'] == 'dep
                    ')].index[0]

                # ---- Turnaround ----

                if activity_type == 'turnaround': # if the interaction is a turnaround
                    min_activity_duration = regulation.loc['turnaround', 'min_time']
                    activity_cost = regulation.loc['turnaround', 'cost'] * statistics.
                        mean([cost, cost_interacted_lineID]) # mean weight of the two
                        lines

                # ---- Transfer ----

                if activity_type == 'transfer':
                    min_activity_duration = regulation.loc['transfer', 'min_time'] # if
```

```
                    the interaction is a trasnfer
                activity_cost = regulation.loc['transfer', 'cost'] * statistics.mean
                    ([cost, cost_interacted_lineID]) # mean weight of the two lines

            activity.loc[len(activity)] = ['a' + str(len(activity) + 1), from_eventID
                , to_eventID, activity_type, activity_location, min_activity_duration
                , activity_cost, 0]

# ---- Headway ----

activity_cost = regulation.loc['headway', 'cost'] # cost of headway activity

# ---- Arrival headway ----

arr_rows = event[event['event_type'].isin(["arr", "arr-pass"])] # filter arrival events
arr_rows = arr_rows.copy()
arr_rows.reset_index(inplace=True)
arr_rows.rename(columns={'index': 'eventID'}, inplace=True) # convert the index to a
    column named 'eventID'
arr_pairs = arr_rows.merge(arr_rows, on='event_location') # merge arrival events
    happening in the same location
arr_pairs = arr_pairs[arr_pairs['lineID_x'] != arr_pairs['lineID_y']] # drop pair of
    events having the same eventID
arr_pairs = arr_pairs[arr_pairs['eventID_x'] < arr_pairs['eventID_y']].reset_index(drop=
    True) # drop duplicates to remove pairs that have been merged in both directions

for idx, row in arr_pairs.iterrows(): # iterate for each arrival event
    to_event_x = row['eventID_x'] # arrival event of the first line
    from_event_x = activity.loc[activity['to_eventID'] == to_event_x, 'from_eventID'].
        values[0] # previous departure event of the first line
    activity_location_x = activity.loc[(activity['to_eventID'] == to_event_x) & (activity
        ['from_eventID'] == from_event_x), 'activity_location'].values[0] # previous
        segment of the first line
    to_event_y = row['eventID_y'] # arrival event of the second line
    from_event_y = activity.loc[activity['to_eventID'] == to_event_y, 'from_eventID'].
        values[0] # previous departure event of the second line
    activity_location_y = activity.loc[(activity['to_eventID'] == to_event_y) & (activity
        ['from_eventID'] == from_event_y), 'activity_location'].values[0] # previous
        segment of the second line

    if activity_location_x == activity_location_y: # if the two trains are coming from
        the same segment
        min_headway = segment.loc[activity_location_x, 'min_headway'] + 60 # min headway
            of that segment
        if int(to_event_x[1:]) < int(to_event_y[1:]):
            activity.loc[len(activity)] = ['a' + str(len(activity) + 1), to_event_x,
                to_event_y, 'headway', row.loc['event_location'], min_headway,
                activity_cost, 0]
        else:
            activity.loc[len(activity)] = ['a' + str(len(activity) + 1), to_event_y,
                to_event_x, 'headway', row.loc['event_location'], min_headway,
                activity_cost, 0]

# ---- Departure headway ----

dep_rows = event[event['event_type'].isin(["dep", "dep-pass"])] # filter departure events
dep_rows = dep_rows.copy()
dep_rows.reset_index(inplace=True)
dep_rows.rename(columns={'index': 'eventID'}, inplace=True) # convert the index to a
    column named 'eventID'
dep_pairs = dep_rows.merge(dep_rows, on='event_location') # merge departure events
    happening in the same location
dep_pairs = dep_pairs[dep_pairs['lineID_x'] != dep_pairs['lineID_y']] # drop pair of
    events having the same eventID
dep_pairs = dep_pairs[dep_pairs['eventID_x'] < dep_pairs['eventID_y']].reset_index(drop=
    True) # drop duplicates to remove pairs that have been merged in both directions

for idx, row in dep_pairs.iterrows(): # iterate for each departure event
    from_event_x = row['eventID_x'] # departure event of the first line
    to_event_x = activity.loc[activity['from_eventID'] == from_event_x, 'to_eventID'].
        values[0] # next arrival event of the first line
```

```
        activity_location_x = activity.loc[(activity['to_eventID'] == to_event_x) & (activity
            ['from_eventID'] == from_event_x), 'activity_location'].values[0] # next segment
            of the first line
        from_event_y = row['eventID_y'] # departure event of the second line
        to_event_y = activity.loc[activity['from_eventID'] == from_event_y, 'to_eventID'].
            values[0] # next arrival event of the second line
        activity_location_y = activity.loc[(activity['to_eventID'] == to_event_y) & (activity
            ['from_eventID'] == from_event_y), 'activity_location'].values[0] # next segment
            of the second line

        if activity_location_x == activity_location_y: # if the two trains are going to the
            same segment
            min_headway = segment.loc[activity_location_x, 'min_headway'] + 60 # min headway
                of that segment
            if int(from_event_x[1:]) < int(from_event_y[1:]):
                activity.loc[len(activity)] = ['a' + str(len(activity) + 1), from_event_x,
                    from_event_y, 'headway', row.loc['event_location'], min_headway,
                    activity_cost, 0]
            else:
                activity.loc[len(activity)] = ['a' + str(len(activity) + 1), from_event_y,
                    from_event_x, 'headway', row.loc['event_location'], min_headway,
                    activity_cost, 0]

    # ---- Mixed headways ----

    dep_rows = event[event['event_type'].isin(["dep", "dep-pass"])]
    dep_rows.reset_index(inplace=True)
    dep_rows.rename(columns={'index': 'eventID'}, inplace=True)
    arr_rows = event[event['event_type'].isin(["arr", "arr-pass"])]
    arr_rows.reset_index(inplace=True)
    arr_rows.rename(columns={'index': 'eventID'}, inplace=True)
    mix_pairs = dep_rows.merge(arr_rows, on='event_location') # merge departure with arrival
        events happening in the same location

    for idx, row in mix_pairs.iterrows(): # iterate for each pair of mixed headways
        from_event_x = row['eventID_x'] # departure event of the first line
        to_event_x = activity.loc[(activity['from_eventID'] == from_event_x) & (activity['
            activity_type'] == 'run'), 'to_eventID'].values[0]
        activity_location_x = activity.loc[(activity['to_eventID'] == to_event_x) & (activity
            ['from_eventID'] == from_event_x), 'activity_location'].values[0] # next segment
            of the first line
        to_event_y = row['eventID_y'] # arrival event of the second line
        from_event_y = activity.loc[(activity['to_eventID'] == to_event_y) & (activity['
            activity_type'] == 'run'), 'from_eventID'].values[0] # previous departure event
            of the second line
        activity_location_y = activity.loc[(activity['to_eventID'] == to_event_y) & (activity
            ['from_eventID'] == from_event_y), 'activity_location'].values[0] # previous
            segment of the second line

        if activity_location_x == activity_location_y: # if the two trains are going to /
            coming from the same segment
            if activity_location_x == 's2':
                min_headway = 0
            else:
                min_headway = segment.loc[activity_location_x, 'min_headway'] + 60 # min
                    headway of that segment

            if int(from_event_x[1:]) < int(to_event_y[1:]):
                activity.loc[len(activity)] = ['a' + str(len(activity) + 1), from_event_x,
                    to_event_y, 'headway', row.loc['event_location'], min_headway,
                    activity_cost, 0]
            else:
                activity.loc[len(activity)] = ['a' + str(len(activity) + 1), to_event_y,
                    from_event_x, 'headway', row.loc['event_location'], min_headway,
                    activity_cost, 0]

    # ---- Regularity ----

    activity_cost = regulation.loc['reg', 'cost']
    min_activity_duration = regulation.loc['reg', 'min_time']
```

```
    for idx, row in train_request.iterrows(): # iterate for each line requested
        baselineID = idx # the lineID without considering the different copy
        events_in_lineID = event[event['lineID'].str.startswith(baselineID + '-')] # events
            from the same baselineID
        copy_of_same_event = events_in_lineID.groupby(['event_location', 'event_type']) #
            find every copy of the same event

        for _, group in copy_of_same_event: # iterate for each copy of the same event
            group['copy_number'] = group['lineID'].apply(lambda x: int(x.split('-')[-1])) #
                extract the copy number X of the lineID-X
            group = group.sort_values(by='copy_number') # sort the events by chronological
                order

            for i in range(len(group) - 1): # iterate for each line requested
                from_eventID = group.iloc[i].name
                to_eventID = group.iloc[i + 1].name
                from_line_num = group.iloc[i]['copy_number']
                to_line_num = group.iloc[i + 1]['copy_number']

                if to_line_num == from_line_num + 1: # we create reg activity only if the two
                     lines are consecutives
                    activity_type = group.iloc[i]['event_type']
                    activity_location = group.iloc[i]['event_location']

                    activity.loc[len(activity)] = ['a' + str(len(activity) + 1), from_eventID
                        , to_eventID, 'reg', activity_location, min_activity_duration,
                        activity_cost, 0]

    activity.set_index('activityID', inplace=True) # set the 'activityID' as index

    return activity
```

Create conflict

```
def create_conflict(event, activity):

        K = pd.DataFrame(columns=['conflictID', 'pair', 'd', 'conflict_location']) # create
            empty K dataframe
        grouped_activity = activity[activity['activity_type'].isin(['run'])].groupby('
            activity_location')
        for activity_location, group in grouped_activity:
            for pair in itertools.combinations(group.index, 2): # Generate all possible pairs
                 (combinations) within the group
                L = []
                for k in range(2):
                    L = L + [event.loc[activity.loc[pair[k], 'from_eventID'], 'lineID'].split
                        ('-')[0]]
                if len(L) == len(set(L)): # if the two events are from two different lines
                    K.loc[len(K)] = ['k' + str(len(K) + 1), pair, activity.loc[pair[0], '
                        train_direction'] * activity.loc[pair[1], 'train_direction'],
                        activity_location]
        K.set_index('conflictID', inplace=True)

        Q = pd.DataFrame(columns=['conflictID', 'combination_activity', 'd','
            conflict_location']) # create empty Q dataframe
        grouped_activity = activity[activity['activity_type'].isin(['dwell', 'pass', '
            turnaround'])].groupby('activity_location')
        for activity_location, group in grouped_activity:
            for pair in itertools.combinations(group.index, 2): # Generate all possible pairs
                 (combinations) within the group
                L = []
                for k in range(2):
                    L = L + [event.loc[activity.loc[pair[k], 'from_eventID'], 'lineID'].split
                        ('-')[0]]
                if len(L) == len(set(L)): # if the two events are from two different lines
                    Q.loc[len(Q)] = ['q' + str(len(Q) + 1), pair, activity.loc[pair[0], '
                        train_direction'] * activity.loc[pair[1], 'train_direction'],
                        activity_location]
        Q.set_index('conflictID', inplace=True)
```

```
        K_multiple = pd.DataFrame(columns=['combinationID', 'combination_activity', '
            combination_conflict', 'conflict_location']) # create empty K_multiple dataframe
        grouped_activity = activity[activity['activity_type'].isin(['run'])].groupby('
            activity_location')
        for activity_location, group in grouped_activity:
            for combination_activity in itertools.combinations(group.index, 3): # Generate
                all possible combination of 3 within the group
                L = []
                for k in range(3):
                    L = L + [event.loc[activity.loc[combination_activity[k], 'from_eventID'],
                        'lineID'].split('-')[0]]
                if len(L) == len(set(L)): # if the three events are all from different lines
                    combination_conflict = []
                    for pair in itertools.combinations(combination_activity, 2):
                        conflictID = K.index[K['pair'] == pair] # we find the corredonding
                            combination of conflictID from the dataframe K
                        if conflictID.empty:
                            raise ValueError(f"No matching index found for pair {pair} in
                                DataFrame K.")
                        combination_conflict = combination_conflict + [conflictID[0]]
                    K_multiple.loc[len(K_multiple)] = ['kk' + str(len(K_multiple) + 1), list(
                        combination_activity), combination_conflict, activity_location]
        K_multiple.set_index('combinationID', inplace=True)

        Q_multiple = pd.DataFrame(columns=['combinationID', 'combination_activity', '
            combination_conflict', 'conflict_location']) # create empty Q_multiple dataframe
        grouped_activity = activity[activity['activity_type'].isin(['dwell', 'pass', '
            turnaround'])].groupby('activity_location')
        for activity_location, group in grouped_activity:
            for i in range(3, len(group) + 1):
                for combination_activity in itertools.combinations(group.index, i): #
                    Generate all possible pairs (combinations) within the group
                    L = []
                    for k in range(i):
                        L = L + [event.loc[activity.loc[combination_activity[k], '
                            from_eventID'], 'lineID'].split('-')[0]]
                    if len(L) == len(set(L)):  # if all the events are from different lines
                        combination_conflict = []
                        for pair in itertools.combinations(combination_activity, 2):  # we
                            find the corredonding combination of conflictID from the
                            dataframe Q
                            conflictID = Q.index[Q['combination_activity'] == pair]
                            if conflictID.empty:
                                raise ValueError(f"No matching index found for pair {pair} in
                                    DataFrame Q")
                            combination_conflict = combination_conflict + [conflictID[0]]
                        Q_multiple.loc[len(Q_multiple)] = ['qq' + str(len(Q_multiple) + 1),
                            list(combination_activity),combination_conflict,
                            activity_location]
        Q_multiple.set_index('combinationID', inplace=True)

        Q_combined = pd.concat([Q, Q_multiple], axis=0, ignore_index=False)

        return(K, K_multiple, Q_combined)
```

B.1.2. Optimization function

```
def optimize_timetable(event, activity, K, K_multiple, Q, segment, interlocking,
    train_request, min_technical_running_times, start_time, period, tmax, alpha):

    # ---- Processing the inputs ----

    event = event.copy()
    activity = activity.copy()

    # ---- Creating the model ----

    model = Model('Macroscopic Timetabling') # creation of the model

    # ---- Parameters ----
```

```
M = period * 60 * 4 # very big value
epsilon = 0.001 # very small value

# ---- Variables ----

event = event.gppd.add_vars(model, name="event_time", lb=0, ub=period * 60 - 1, vtype=GRB
    .INTEGER)
activity = activity.gppd.add_vars(model, name="activity_duration", lb=0, ub=period * 60 -
     1, vtype=GRB.INTEGER)
activity = activity.gppd.add_vars(model, name="modulo_operator", vtype=GRB.BINARY)
K = K.gppd.add_vars(model, name="k", vtype=GRB.BINARY)
Q = Q.gppd.add_vars(model, name="q", vtype=GRB.BINARY)
segment = segment.gppd.add_vars(model, name="t_needed", lb=1, ub=4, vtype=GRB.INTEGER)
segment = segment.gppd.add_vars(model, name="t_after", lb=1, ub=4, vtype=GRB.INTEGER)
segment = segment.gppd.add_vars(model, name="k_same_direction", vtype=GRB.BINARY)
segment = segment.gppd.add_vars(model, name="k_opposite_direction", vtype=GRB.BINARY)
interlocking = interlocking.gppd.add_vars(model, name="t_needed", lb=1, vtype=GRB.INTEGER
    )
interlocking = interlocking.gppd.add_vars(model, name="t_after", lb=1, vtype=GRB.INTEGER)
model.update()

# ---- Objective Function ----

min_technical_running_times['sum'] = min_technical_running_times[['s1', 's2', 's3']].sum(
    axis=1)
tot_min_running_cost= 0
tot_max_running_cost= 0
for idx, row in train_request.iterrows():
    tot_min_running_cost += (min_technical_running_times.loc[row['rolling_stock'], 'sum']
        * row['cost'] * row['frequency'])
    tot_max_running_cost += 3600 * row['cost'] * row['frequency']

# Maximal value possible
max_operation_cost = tot_max_running_cost / 1000000 # in million SEK
max_construction_cost_segment = tmax * 100 # in million SEK
max_construction_cost_interlocking = tmax * 10 # in million SEK

# Minimal value possible
min_operation_cost = tot_min_running_cost / 1000000 # in million SEK
min_construction_cost_segment = 0  # in million SEK
min_construction_cost_interlocking = 0  # in million SEK

# Total costs
total_operation_cost = (activity['activity_duration'] * activity['activity_cost']).sum()
    / 1000000 # in million SEK
total_construction_cost_segment = (segment['cost'] * (segment['t_after'] - segment['
    t_before'])).sum() # in million SEK
total_construction_cost_interlocking = (interlocking['cost'] * (interlocking['t_after'] -
     interlocking['t_before'])).sum() # in million SEK

# Costs normalization
operation_cost_normalized = (total_operation_cost - min_operation_cost) / (
    max_operation_cost - min_operation_cost)
construction_cost_normalized_segment = (total_construction_cost_segment -
    min_construction_cost_segment) / (max_construction_cost_segment -
    min_construction_cost_segment)
construction_cost_normalized_interlocking = (total_construction_cost_interlocking -
    min_construction_cost_interlocking) / (max_construction_cost_interlocking -
    min_construction_cost_interlocking)

# Set the objective function using normalized total costs, alpha a coefficient reflection
     on the priority
model.setObjective(((alpha) * operation_cost_normalized) + ((1 - alpha)/2 *
    construction_cost_normalized_segment) + ((1 - alpha)/2 *
    construction_cost_normalized_interlocking), GRB.MINIMIZE)
model.update()

# ---- Constraints ----

# ---- Constraint 1: Definition of the activity duration ----
```

```
for index, activity_row in activity.iterrows():
    model.addConstr(activity_row['activity_duration'] == (event.loc[activity_row['
        to_eventID'], 'event_time']) - (event.loc[activity_row['from_eventID'], '
        event_time']) + activity_row['modulo_operator'] * period * 60, name=f"con1_{index
        }")
model.update()

# ---- Constraints 2: Activity duration between min and max or set to period if reg ----

for index, activity_row in activity.iterrows():
    if activity_row['activity_type'] == 'reg':
        line_period = period * 60 / event.loc[activity_row['from_eventID'], '
            event_frequency']
        model.addConstr(activity_row['activity_duration'] == line_period, name=f"con2a_{
            index}")
    else:
        model.addConstr(activity_row['activity_duration'] >= activity_row['
            min_activity_duration'], name=f"con2b_{index}")
        model.addConstr(activity_row['activity_duration'] <= period * 60 - activity_row['
            min_activity_duration'], name=f"con2c_{index}")
model.update()

# ---- Constraint 3: Definition of the modulo operator ----

for index, activity_row in activity.iterrows():
        model.addConstr(event.loc[activity_row['to_eventID'], 'event_time'] <= event.loc[
            activity_row['from_eventID'], 'event_time'] - epsilon + M * (1 - activity_row
            ['modulo_operator']), name=f"con3_{index}")
model.update()

# ---- Constraint 4: Total number of new tracks ----

model.addConstr((interlocking['t_after'] - interlocking['t_before']).sum() + (segment['
    t_after'] - segment['t_before']).sum() <= 1, name=f"con4_{index}")
model.update()

# ---- Constraint 5: Defition of the number of tracks after ----

for index, segment_row in segment.iterrows():
    model.addConstr(segment_row['t_after'] >= segment_row['t_needed'], name=f"con5a_{
        index}")
    model.addConstr(segment_row['t_after'] >= segment_row['t_before'], name=f"con5b_{
        index}")
model.update()

for index, interlocking_row in interlocking.iterrows():
    model.addConstr(interlocking_row['t_after'] >= interlocking_row['t_needed'], name=f"
        con5c_{index}")
    model.addConstr(interlocking_row['t_after'] >= interlocking_row['t_before'], name=f"
        con5d_{index}")
model.update()

# ---- Constraint 6: Definition number of tracks needed at segment ----

for index, segment_row in segment.iterrows():
    z1 = model.addVar(vtype=GRB.BINARY, name=f"z1_{index}")  # help variable
    z2 = model.addVar(vtype=GRB.BINARY, name=f"z2_{index}")  # help variable
    z3 = model.addVar(vtype=GRB.BINARY, name=f"z3_{index}")  # help variable

    # Add the constraints for y = kopp * ksame (both are binary variables)
    model.addConstr(z1 <= segment_row['k_opposite_direction'], name=f"con6a_{index}")
    model.addConstr(z1 <= segment_row['k_same_direction'], name=f"con6b_{index}")
    model.addConstr(z1 >= segment_row['k_opposite_direction'] + segment_row['
        k_same_direction'] - 1, name=f"con6c_{index}")
    model.addConstr(z2 == 1 - segment_row['k_same_direction'] - segment_row['
        k_opposite_direction'] + z1, name=f"con6d_{index}")
    model.addConstr(z3 == segment_row['k_opposite_direction'] - z1, name=f"con6e_{index}"
        )
    model.addConstr(segment_row['t_needed'] == (z2 * 1) + (z3 * 2) + (segment_row['
        k_same_direction'] * 4), name=f"con6f_{index}")
model.update()
```

```
# ---- Constraint 7: Definition K_same and K_opp ----

for index, segment_row in segment.iterrows():
    sum_same_direction = quicksum(K.loc[(K['d'] == 1) & (K['conflict_location'] == index)
        , 'k'])
    model.addConstr(sum_same_direction <= M * segment_row['k_same_direction'], name=f"
        con7a_{index}")
    model.addConstr(sum_same_direction >= segment_row['k_same_direction'], name=f"con7b_{
        index}")
    sum_opposite_direction = quicksum(K.loc[(K['d'] == -1) & (K['conflict_location'] ==
        index), 'k'])
    model.addConstr(sum_opposite_direction <= M * segment_row['k_opposite_direction'],
        name=f"con7c_{index}")
    model.addConstr(sum_opposite_direction >= segment_row['k_opposite_direction'], name=f
        "con7d_{index}")
model.update()

# ---- Constraint 8: Definition number of tracks needed at interlocking ----

for index, interlocking_row in interlocking.iterrows():
    Q_interlocking = Q[Q['conflict_location'] == index]
    if not Q_interlocking.empty:
        for i, Q_interlocking_row in Q_interlocking.iterrows():
            model.addConstr(interlocking_row['t_needed'] >= Q_interlocking_row['q'] * len
                (Q_interlocking_row['combination_activity']), name=f"con8a_{index}")
    else:
        model.addConstr(interlocking_row['t_needed'] == 1, name=f"con8b_{index}")

# ---- Constraint 9: Defintion of a conflict at interlocking: table Q ---

for index, Q_row in Q.dropna(subset=['d']).iterrows():

    z4 = model.addVar(vtype=GRB.INTEGER, name=f"z4_{index}") # help variable

    a, a_prime = Q_row['combination_activity'][0], Q_row['combination_activity'][1] # Get
         the activityID 'a' from the index.
    from_eventID_a = activity.loc[a, 'from_eventID']
    to_eventID_a = activity.loc[a, 'to_eventID']
    from_eventID_a_prime = activity.loc[a_prime, 'from_eventID']
    to_eventID_a_prime = activity.loc[a_prime, 'to_eventID']

    if Q_row['d'] == 1:
        condition1 = (activity['from_eventID'] == from_eventID_a) & (activity['to_eventID
            '] == from_eventID_a_prime) & (activity['activity_type'] == 'headway')
        condition2 = (activity['from_eventID'] == from_eventID_a_prime) & (activity['
            to_eventID'] == from_eventID_a) & (activity['activity_type'] == 'headway')
        matching_activityID = activity.index[condition1 | condition2][0]
        condition1_prime = (activity['from_eventID'] == to_eventID_a) & (activity['
            to_eventID'] == to_eventID_a_prime) & (activity['activity_type'] == 'headway'
            )
        condition2_prime = (activity['from_eventID'] == to_eventID_a_prime) & (activity['
            to_eventID'] == to_eventID_a) & (activity['activity_type'] == 'headway')
        matching_activityID_prime = activity.index[condition1_prime | condition2_prime
            ][0]

    elif Q_row['d'] == -1:
        condition3 = (activity['from_eventID'] == from_eventID_a) & (activity['to_eventID
            '] == to_eventID_a_prime) & (activity['activity_type'] == 'headway')
        condition4 = (activity['from_eventID'] == to_eventID_a_prime) & (activity['
            to_eventID'] == from_eventID_a) & (activity['activity_type'] == 'headway')
        matching_activityID = activity.index[condition3 | condition4][0]
        condition3_prime = (activity['from_eventID'] == to_eventID_a) & (activity['
            to_eventID'] == from_eventID_a_prime) & (activity['activity_type'] == '
            headway')
        condition4_prime = (activity['from_eventID'] == from_eventID_a_prime) & (activity
            ['to_eventID'] == to_eventID_a) & (activity['activity_type'] == 'headway')
        matching_activityID_prime = activity.index[condition3_prime | condition4_prime
            ][0]
```

```
        modulo_sum = activity.loc[a,'modulo_operator'] + activity.loc[a_prime,'
            modulo_operator'] + activity.loc[matching_activityID,'modulo_operator'] +
            activity.loc[matching_activityID_prime,'modulo_operator']
        model.addConstr(modulo_sum - 2 * z4 == Q_row.loc['q'], name=f"con9_{index}")
    model.update()

    # ---- Constraint 10: Defintion of a conflict at segment: table K ----

    for index, K_row in K.iterrows():

        z5 = model.addVar(vtype=GRB.INTEGER, name=f"z5_{index}") # help variable

        a, a_prime = K_row['pair'][0], K_row['pair'][1] # Get the activityID 'a' from the
            index.
        from_eventID_a = activity.loc[a, 'from_eventID']
        to_eventID_a = activity.loc[a, 'to_eventID']
        from_eventID_a_prime = activity.loc[a_prime, 'from_eventID']
        to_eventID_a_prime = activity.loc[a_prime, 'to_eventID']

        if K_row['d'] == 1:
            condition1 = (activity['from_eventID'] == from_eventID_a) & (activity['to_eventID
                '] == from_eventID_a_prime) & (activity['activity_type'] == 'headway')
            condition2 = (activity['from_eventID'] == from_eventID_a_prime) & (activity['
                to_eventID'] == from_eventID_a) & (activity['activity_type'] == 'headway')
            matching_activityID = activity.index[condition1 | condition2][0]
            condition1_prime = (activity['from_eventID'] == to_eventID_a) & (activity['
                to_eventID'] == to_eventID_a_prime) & (activity['activity_type'] == 'headway'
                )
            condition2_prime = (activity['from_eventID'] == to_eventID_a_prime) & (activity['
                to_eventID'] == to_eventID_a) & (activity['activity_type'] == 'headway')
            matching_activityID_prime = activity.index[condition1_prime | condition2_prime
                ][0]

        elif K_row['d'] == -1:
            condition3 = (activity['from_eventID'] == from_eventID_a) & (activity['to_eventID
                '] == to_eventID_a_prime) & (activity['activity_type'] == 'headway')
            condition4 = (activity['from_eventID'] == to_eventID_a_prime) & (activity['
                to_eventID'] == from_eventID_a) & (activity['activity_type'] == 'headway')
            matching_activityID = activity.index[condition3 | condition4][0]
            condition3_prime = (activity['from_eventID'] == to_eventID_a) & (activity['
                to_eventID'] == from_eventID_a_prime) & (activity['activity_type'] == '
                headway')
            condition4_prime = (activity['from_eventID'] == from_eventID_a_prime) & (activity
                ['to_eventID'] == to_eventID_a) & (activity['activity_type'] == 'headway')
            matching_activityID_prime = activity.index[condition3_prime | condition4_prime
                ][0]

        modulo_sum = activity.loc[a,'modulo_operator'] + activity.loc[a_prime,'
            modulo_operator'] + activity.loc[matching_activityID,'modulo_operator'] +
            activity.loc[matching_activityID_prime,'modulo_operator']
        model.addConstr(modulo_sum - 2 * z5 == K_row.loc['k'], name=f"con10_{index}")
    model.update()

    # ---- Constraint 11: Multiple conflicts at interlocking: table Q_multiple ----

    for index, Q_row in Q.dropna(subset=['combination_conflict']).iterrows():
        quick_sum = LinExpr()
        for conflictID in Q_row['combination_conflict']:
            quick_sum += Q.at[conflictID, 'q']
        model.addConstr(quick_sum - len(Q_row['combination_conflict']) <= M * Q_row.loc['q'],
             name=f"con11a_{index}")
        model.addConstr(quick_sum - len(Q_row['combination_conflict']) >= epsilon - M * (1 -
            Q_row.loc['q']), name=f"con11b_{index}")
    model.update()

    # ---- Constraint 12: Multiple conflicts at segment not allowed: table K_multiple ----

    for index, K_multiple_row in K_multiple.iterrows():
        quick_sum = LinExpr()
        for conflictID in K_multiple_row['combination_conflict']:
            quick_sum += K.at[conflictID, 'k']
```

```
        model.addConstr(quick_sum <= 2, name=f"con12_{index}")
    model.update()

    # ---- Solve ----

    model.setParam('MIPGap', 0.05) # Allow a 5% optimality gap
    model.setParam('NodefileStart', 0.25) # Start writing nodes to disk at 0.5 GB memory
        usage
    model.optimize()

    # ---- Print results ----

    print ('\n-----------------------------------------------------------------------\n')

    if model.status == GRB.OPTIMAL:  # if optimal solution is found
        feasible = True
        # read the optimized values into the DataFrame
        for index, activity_row in activity.iterrows():
            activity.loc[index, 'activity_duration'] = activity_row['activity_duration'].x
            activity.loc[index, 'modulo_operator'] = activity_row['modulo_operator'].x
        for index, event_row in event.iterrows():
            event.loc[index, 'event_time'] = event_row['event_time'].x
        for index, K_row in K.iterrows():
            K.loc[index, 'k'] = K_row['k'].x
        for index, segment_row in segment.iterrows():
            segment.loc[index, 't_needed'] = segment_row['t_needed'].x
            segment.loc[index, 't_after'] = segment_row['t_after'].x
            segment.loc[index, 'k_same_direction'] = segment_row['k_same_direction'].x
            segment.loc[index, 'k_opposite_direction'] = segment_row['k_opposite_direction'].
                x
        for index, Q_row in Q.iterrows():
            Q.loc[index, 'q'] = Q_row['q'].x
        for index, interlocking_row in interlocking.iterrows():
            interlocking.loc[index, 't_needed'] = interlocking_row['t_needed'].x
            interlocking.loc[index, 't_after'] = interlocking_row['t_after'].x
        event['event_time'] = event['event_time'].apply(lambda x: start_time + pd.Timedelta(
            seconds=x))

        result_data = {"model_name": model.ModelName,
                       "optimization_status": model.Status,
                       "objective_value": model.ObjVal,
                       "operation_cost_optimal": operation_cost_normalized.getValue(),
                       "construction_cost_segment_optimal":
                           construction_cost_normalized_segment.getValue(),
                       "construction_cost_interlocking_optimal":
                           construction_cost_normalized_interlocking.getValue(),
                       "construction_costs": (interlocking['cost'] * (interlocking['t_after']
                            - interlocking['t_before'])).sum() + (segment['cost'] * (segment[
                           't_after'] - segment['t_before'])).sum(),
                       "operation_costs": ((activity['activity_duration'] * activity['
                           activity_cost']).sum()),
                       "number_of_variables": model.NumVars,
                       "number_of_binary_variables": sum(1 for v in model.getVars() if v.
                           VType == GRB.BINARY),
                       "number_of_constraints": model.NumConstrs,
                       "number_of_nonzeros": model.NumNZs,
                       "number_of_iterations": model.IterCount,
                       "optimization_runtime_seconds": model.Runtime,
                       "mip_gap": model.MIPGap if model.IsMIP else None}
        result_df = pd.DataFrame(result_data.items(), columns=["Metric", "Value"])

    else: # if no optimal solution is found
       print("No feasible solution found")
       feasible = False

    return(feasible, event, activity, K, K_multiple, Q, segment, interlocking, result_df)
```

B.1.3. Visualization function

```
def plot_timetable(event, interlocking, plot_start_time, plot_end_time, tick_interval, period
    , corridor, train_request, file_path):

    distance = [0]  # initialize an empty list to store the cumulative distances

    for i in range(1, len(corridor)):  # iterate through the corridor and calculate the
        cumulative distances
        if int(corridor[i - 1][1:]) < int(corridor[i][1:]):  # find the segment length
            between the current and previous interlocking
            segment_length = segment.loc[(segment['from_interlockingID'] == corridor[i - 1])
                & (segment['to_interlockingID'] == corridor[i]), 'length'].values[0]
        else:
            segment_length = segment.loc[(segment['from_interlockingID'] == corridor[i]) & (
                segment['to_interlockingID'] == corridor[i-1]), 'length'].values[0]

        cumulative_distance = distance[i - 1] + segment_length  # add the segment length to
            the previous cumulative distance
        distance.append(cumulative_distance)  # append the new cumulative distance to the
            list

    df_plot = pd.DataFrame({'interlockingID': corridor, 'cumulated_distance': distance})  #
        create a DataFrame with interlockingID and cumulated_distance columns
    event_filtered = event[event['event_location'].isin(corridor)]  # only the event
        happening in the corridor have to be plotted
    event_grouped = event_filtered.groupby('lineID')

    fig, ax = plt.subplots()  # create graph
    ax.xaxis.set_major_formatter(DateFormatter('%H:%M:%S'))  # set major formatter for x-axis
         to display time as hh:mm:ss
    plotted_lines = set()

    for copy in [-period, 0]:

        for lineID, group in event_grouped:

            location_line = group['event_location']
            time_line = group['event_time'] + pd.DateOffset(minutes=copy)

            # check for decreasing values in time_line and adjust them
            for i in range(len(time_line) - 1):
                if time_line.iloc[i] > time_line.iloc[i + 1]:
                    time_line.iloc[i + 1] += timedelta(minutes=period)
            base_lineID = lineID.split('-')[0]

            for period_idx in range(0, (plot_end_time - plot_start_time).seconds, int((period
                 * 60))):  # iterate over each period, if still within the time window
                X = [t + timedelta(minutes=period_idx / 60) for t in time_line]
                service_type = train_request.loc[base_lineID, 'service_type'] # define the
                    color of the line regarding the type of activity
                if service_type == 'IC':
                    c = 'blue'
                elif service_type == 'R':
                    c = 'green'
                elif service_type == 'F':
                    c = 'black'
                elif service_type == 'HS':
                    c = 'red'
                else:
                    c = 'gray'

                Y = []  # list of cumulated distance of each interlocking of the corridor
                Y_labels = []  # list of corresponding interlocking label

                for interlockingID in location_line:
                    cumulative_distance = df_plot[df_plot['interlockingID'] == interlockingID
                        ]['cumulated_distance'].values[0]
                    Y.append(cumulative_distance)
                    station_code = interlocking.loc[interlockingID, 'station_code']
                    if not pd.isnull(station_code):
                        Y_labels.append(f"{station_code} - {interlockingID}")
                    else:
```

```
                    Y_labels.append(interlockingID)

                ax.plot(X, Y, linewidth=1, color=c, label=lineID if lineID not in
                    plotted_lines else None)

                if lineID not in plotted_lines:
                    plotted_lines.add(lineID)

    # set y-axis labels to interlocking names
    ax.set_yticks(Y)
    ax.set_yticklabels(Y_labels)
    ax.set_ylim(Y[0], Y[-1])

    # set x-axis
    plt.xticks(rotation=67.5) # rotate
    ax.set_xlim(plot_start_time, plot_end_time) # time window
    tick = mdates.drange(plot_start_time, plot_end_time + timedelta(seconds=0.1), timedelta(
        seconds=tick_interval)) # calculate tick locations starting from start_time
    ax.xaxis.set_major_locator(ticker.FixedLocator(tick))

    # add legend based on the chosen colors and lineIDs
    handles, labels = ax.get_legend_handles_labels()
    unique_labels = [label for label in labels if label]
    ax.legend(handles, labels, loc='center␣left', bbox_to_anchor=(1, 0.5))

    ax.grid(True)
    #plt.show()
    plt.savefig(file_path, bbox_inches='tight')
    plt.close()
```

C

Data used for the case study

C.1. Traffic 2022 and forecast 2040

TOTAL NUMBER OF TRAINS

Jkb-Kbå corridor	Type of train	Freight train	Regional train	Intercity	Fast train	Total
	Rolling stock	*Godståg*	*ER1*	*X40*	*X50-54*	
2022	06:00 to 18:00	11,4	20,0	24,5	-	55,9
	18:00 to 22:00	5,6	6,8	7,3	-	19,7
	22:00 to 06:00	7,0	1,7	2,9	-	11,7
	Total 24h	24,0	28,5	34,7	-	87,3
2040	06:00 to 18:00	13,0	59,1	17,3	-	89,4
	18:00 to 22:00	6,4	20,0	5,2	-	31,6
	22:00 to 06:00	8,0	5,0	2,1	-	15,1
	Total 24h	27,4	84,2	24,5	-	136,1

Evolution	Total 24h	14,0%	195,4%	-29,5%	-

FREQUENCIES PER HOUR

Jkb-Kbå corridor	Type of train	Freight train	Regional train	Intercity
	Rolling stock	*Godståg*	*ER1*	*X40*
2022	06:00 to 18:00	0,9	1,7	2,0
	18:00 to 22:00	1,4	1,7	1,8
	22:00 to 06:00	0,9	0,2	0,4
	Total 24h	1,0	1,2	1,4
2040	06:00 to 18:00	1,1	4,9	1,4
	18:00 to 22:00	1,6	5,0	1,3
	22:00 to 06:00	1,0	0,6	0,3
	Total 24h	1,1	3,5	1,0

ROUNDED FREQUENCIES, PER DIRECTION PER HOUR

Jkb-Kbå corridor	Type of train	Freight train	Regional train	Intercity
	Rolling stock	*Godståg*	*ER1*	*X40*
2022	06:00 to 18:00	1	1	1
	18:00 to 22:00	1	1	1
	22:00 to 06:00	-	-	-
	Total 24h	2	2	2
2040	06:00 to 18:00	1	2	1
	18:00 to 22:00	1	3	1
	22:00 to 06:00	1	-	-
	Total 24h	3	5	2

Figure C.1: Database used for the case study

Data highlighted in blue are not directly available and have been calculated assuming that the proportion of the trains running on the different time slots in 2040 is the same than in 2022. Moreover, it is assumed that 50% of the trains run from Jbk to Kbä, and 50% run in the opposite direction.

D

Results of the case study

D.1. Scenario 1

eventID	lineID	event_type	event_location	event_frequency	event_time
e1	FR1-1	dep	i1	1	08:26:38
e2	FR1-1	arr-pass	i2	1	08:28:49
e3	FR1-1	dep-pass	i2	1	08:28:49
e4	FR1-1	arr-pass	i3	1	08:35:41
e5	FR1-1	dep-pass	i3	1	08:35:41
e6	FR1-1	arr-pass	i4	1	08:42:49
e7	FR1-1	dep-pass	i4	1	08:06:09
e8	FR1-1	arr-pass	i5	1	08:13:17
e9	FR1-1	dep-pass	i5	1	08:13:17
e10	FR1-1	arr	i6	1	08:18:39
e11	FR2-1	dep	i6	1	08:49:39
e12	FR2-1	arr-pass	i5	1	08:55:01
e13	FR2-1	dep-pass	i5	1	08:55:01
e14	FR2-1	arr-pass	i4	1	08:02:09
e15	FR2-1	dep-pass	i4	1	08:02:09
e16	FR2-1	arr-pass	i3	1	08:09:17
e17	FR2-1	dep-pass	i3	1	08:09:17
e18	FR2-1	arr-pass	i2	1	08:17:39
e19	FR2-1	dep-pass	i2	1	08:17:39
e20	FR2-1	arr	i1	1	08:19:50
e21	R1-1	dep	i1	1	08:00:36
e22	R1-1	arr	i2	1	08:02:28
e23	R1-1	dep	i2	1	08:04:28
e24	R1-1	arr-pass	i3	1	08:09:17
e25	R1-1	dep-pass	i3	1	08:13:17
e26	R1-1	arr	i4	1	08:18:05
e27	R1-1	dep	i4	1	08:20:05
e28	R1-1	arr-pass	i5	1	08:24:36
e29	R1-1	dep-pass	i5	1	08:24:36
e30	R1-1	arr	i6	1	08:27:37
e31	R2-1	dep	i6	1	08:45:39
e32	R2-1	arr-pass	i5	1	08:48:40
e33	R2-1	dep-pass	i5	1	08:48:40
e34	R2-1	arr	i4	1	08:53:11
e35	R2-1	dep	i4	1	08:55:11
e36	R2-1	arr-pass	i3	1	08:59:59
e37	R2-1	dep-pass	i3	1	08:59:59
e38	R2-1	arr	i2	1	08:04:28
e39	R2-1	dep	i2	1	08:06:28

eventID	lineID	event_type	event_location	event_frequency	event_time
e40	R2-1	arr	i1	1	08:08:20
e41	IC1-1	dep	i1	1	08:12:20
e42	IC1-1	arr-pass	i2	1	08:13:39
e43	IC1-1	dep-pass	i2	1	08:13:39
e44	IC1-1	arr-pass	i3	1	08:17:17
e45	IC1-1	dep-pass	i3	1	08:17:17
e46	IC1-1	arr	i4	1	08:22:07
e47	IC1-1	dep	i4	1	08:24:07
e48	IC1-1	arr-pass	i5	1	08:28:47
e49	IC1-1	dep-pass	i5	1	08:28:47
e50	IC1-1	arr	i6	1	08:31:51
e51	IC2-1	dep	i6	1	08:37:05
e52	IC2-1	arr-pass	i5	1	08:40:09
e53	IC2-1	dep-pass	i5	1	08:40:09
e54	IC2-1	arr	i4	1	08:44:49
e55	IC2-1	dep	i4	1	08:46:49
e56	IC2-1	arr-pass	i3	1	08:51:39
e57	IC2-1	dep-pass	i3	1	08:51:39
e58	IC2-1	arr-pass	i2	1	08:55:17
e59	IC2-1	dep-pass	i2	1	08:55:17
e60	IC2-1	arr	i1	1	08:56:36

Table D.1: Output event DataFrame of scenario 1

segmentID	t_before	length	cost	t_needed	t_after	k_same_direction	k_opp_direction
s1	1	2264	844	1	1	0	0
s2	2	4355	1624	2	2	0	1
s3	1	6920	2581	1	1	0	0
s4	1	6799	2536	1	1	0	0
s5	1	5072	1892	1	1	0	0

Table D.2: Output segment DataFrame of scenario 1

interlockingID	station_code	t_before	cost	t_needed	t_after
i1	Jbk	2	25	1	2
i2	Arb	2	51	2	2
i3	Vsg	2	25	2	2
i4	Kp	3	64	3	3
i5	Morp	2	25	2	2
i6	Kba	2	30	1	2

Table D.3: Output interlocking DataFrame of scenario 1

D.2. Scenario 2

eventID	lineID	event_type	event_location	event_frequency	event_time
e1	FR1-1	dep	i1	1	08:00:41
e2	FR1-1	arr-pass	i2	1	08:02:52
e3	FR1-1	dep-pass	i2	1	08:03:59
e4	FR1-1	arr-pass	i3	1	08:10:51
e5	FR1-1	dep-pass	i3	1	08:10:51
e6	FR1-1	arr-pass	i4	1	08:17:59
e7	FR1-1	dep-pass	i4	1	08:17:59
e8	FR1-1	arr-pass	i5	1	08:26:11
e9	FR1-1	dep-pass	i5	1	08:26:11

eventID	lineID	event_type	event_location	event_frequency	event_time
e10	FR1-1	arr	i6	1	08:31:33
e11	FR2-1	dep	i6	1	08:35:33
e12	FR2-1	arr-pass	i5	1	08:40:55
e13	FR2-1	dep-pass	i5	1	08:40:55
e14	FR2-1	arr-pass	i4	1	08:48:03
e15	FR2-1	dep-pass	i4	1	08:51:43
e16	FR2-1	arr-pass	i3	1	08:58:51
e17	FR2-1	dep-pass	i3	1	08:00:00
e18	FR2-1	arr-pass	i2	1	08:06:52
e19	FR2-1	dep-pass	i2	1	08:06:52
e20	FR2-1	arr	i1	1	08:09:03
e21	R1-1	dep	i1	2	08:56:07
e22	R1-1	arr	i2	2	08:57:59
e23	R1-1	dep	i2	2	08:59:59
e24	R1-1	arr-pass	i3	2	08:03:58
e25	R1-1	dep-pass	i3	2	08:06:51
e26	R1-1	arr	i4	2	08:11:39
e27	R1-1	dep	i4	2	08:13:39
e28	R1-1	arr-pass	i5	2	08:22:11
e29	R1-1	dep-pass	i5	2	08:22:11
e30	R1-1	arr	i6	2	08:25:12
e31	R1-2	dep	i1	2	08:26:07
e32	R1-2	arr	i2	2	08:27:59
e33	R1-2	dep	i2	2	08:29:59
e34	R1-2	arr-pass	i3	2	08:33:58
e35	R1-2	dep-pass	i3	2	08:36:51
e36	R1-2	arr	i4	2	08:41:39
e37	R1-2	dep	i4	2	08:43:39
e38	R1-2	arr-pass	i5	2	08:52:11
e39	R1-2	dep-pass	i5	2	08:52:11
e40	R1-2	arr	i6	2	08:55:12
e41	R2-1	dep	i6	2	08:45:10
e42	R2-1	arr-pass	i5	2	08:48:11
e43	R2-1	dep-pass	i5	2	08:48:11
e44	R2-1	arr	i4	2	08:56:03
e45	R2-1	dep	i4	2	08:58:03
e46	R2-1	arr-pass	i3	2	08:02:51
e47	R2-1	dep-pass	i3	2	08:07:16
e48	R2-1	arr	i2	2	08:11:15
e49	R2-1	dep	i2	2	08:13:15
e50	R2-1	arr	i1	2	08:15:07
e51	R2-2	dep	i6	2	08:15:10
e52	R2-2	arr-pass	i5	2	08:18:11
e53	R2-2	dep-pass	i5	2	08:18:11
e54	R2-2	arr	i4	2	08:26:03
e55	R2-2	dep	i4	2	08:28:03
e56	R2-2	arr-pass	i3	2	08:32:51
e57	R2-2	dep-pass	i3	2	08:37:16
e58	R2-2	arr	i2	2	08:41:15
e59	R2-2	dep	i2	2	08:43:15
e60	R2-2	arr	i1	2	08:45:07
e61	IC1-1	dep	i1	1	08:37:56
e62	IC1-1	arr-pass	i2	1	08:39:15
e63	IC1-1	dep-pass	i2	1	08:39:15
e64	IC1-1	arr-pass	i3	1	08:42:53
e65	IC1-1	dep-pass	i3	1	08:42:53
e66	IC1-1	arr	i4	1	08:47:43
e67	IC1-1	dep	i4	1	08:52:03
e68	IC1-1	arr-pass	i5	1	08:56:43
e69	IC1-1	dep-pass	i5	1	08:56:43

eventID	lineID	event_type	event_location	event_frequency	event_time
e70	IC1-1	arr	i6	1	08:59:47
e71	IC2-1	dep	i6	1	08:11:07
e72	IC2-1	arr-pass	i5	1	08:14:11
e73	IC2-1	dep-pass	i5	1	08:14:11
e74	IC2-1	arr	i4	1	08:21:59
e75	IC2-1	dep	i4	1	08:23:59
e76	IC2-1	arr-pass	i3	1	08:28:49
e77	IC2-1	dep-pass	i3	1	08:28:49
e78	IC2-1	arr-pass	i2	1	08:32:27
e79	IC2-1	dep-pass	i2	1	08:32:27
e80	IC2-1	arr	i1	1	08:33:46

Table D.4: Output event DataFrame of scenario 2

segmentID	t_before	length	cost	t_needed	t_after	k_same_direction	k_opp_direction
s1	1	2264	844	1	1	0	0
s2	2	4355	1624	2	2	0	1
s3	1	6920	2581	1	1	0	0
s4	1	6799	2536	2	2	0	1
s5	1	5072	1892	1	1	0	0

Table D.5: Output segment DataFrame of scenario 2

interlockingID	station_code	t_before	cost	t_needed	t_after
i1	Jbk	2	25	1	2
i2	Arb	2	51	2	2
i3	Vsg	2	25	2	2
i4	Kp	3	64	3	3
i5	Morp	2	25	2	2
i6	Kba	2	30	1	2

Table D.6: Output interlocking DataFrame of scenario 2

D.3. Scenario 3

eventID	lineID	event_type	event_location	event_frequency	event_time
e1	FR1-1	dep	i1	1	08:30:47
e2	FR1-1	arr-pass	i2	1	08:32:58
e3	FR1-1	dep-pass	i2	1	08:35:19
e4	FR1-1	arr-pass	i3	1	08:42:11
e5	FR1-1	dep-pass	i3	1	08:42:11
e6	FR1-1	arr-pass	i4	1	08:49:19
e7	FR1-1	dep-pass	i4	1	08:49:19
e8	FR1-1	arr-pass	i5	1	08:56:27
e9	FR1-1	dep-pass	i5	1	08:56:27
e10	FR1-1	arr	i6	1	08:01:49
e11	FR2-1	dep	i6	1	08:33:49
e12	FR2-1	arr-pass	i5	1	08:39:11
e13	FR2-1	dep-pass	i5	1	08:39:11
e14	FR2-1	arr-pass	i4	1	08:46:19
e15	FR2-1	dep-pass	i4	1	08:46:19
e16	FR2-1	arr-pass	i3	1	08:53:27
e17	FR2-1	dep-pass	i3	1	08:53:29
e18	FR2-1	arr-pass	i2	1	08:00:21
e19	FR2-1	dep-pass	i2	1	08:01:20

eventID	lineID	event_type	event_location	event_frequency	event_time
e20	FR2-1	arr	i1	1	08:03:31
e21	R1-1	dep	i1	2	08:56:28
e22	R1-1	arr	i2	2	08:58:20
e23	R1-1	dep	i2	2	08:00:20
e24	R1-1	arr-pass	i3	2	08:04:19
e25	R1-1	dep-pass	i3	2	08:04:19
e26	R1-1	arr	i4	2	08:09:07
e27	R1-1	dep	i4	2	08:11:07
e28	R1-1	arr-pass	i5	2	08:15:38
e29	R1-1	dep-pass	i5	2	08:15:38
e30	R1-1	arr	i6	2	08:18:39
e31	R1-2	dep	i1	2	08:26:28
e32	R1-2	arr	i2	2	08:28:20
e33	R1-2	dep	i2	2	08:30:20
e34	R1-2	arr-pass	i3	2	08:34:19
e35	R1-2	dep-pass	i3	2	08:34:19
e36	R1-2	arr	i4	2	08:39:07
e37	R1-2	dep	i4	2	08:41:07
e38	R1-2	arr-pass	i5	2	08:45:38
e39	R1-2	dep-pass	i5	2	08:45:38
e40	R1-2	arr	i6	2	08:48:39
e41	R2-1	dep	i6	2	08:45:39
e42	R2-1	arr-pass	i5	2	08:48:40
e43	R2-1	dep-pass	i5	2	08:48:40
e44	R2-1	arr	i4	2	08:53:11
e45	R2-1	dep	i4	2	08:55:11
e46	R2-1	arr-pass	i3	2	08:59:59
e47	R2-1	dep-pass	i3	2	08:59:59
e48	R2-1	arr	i2	2	08:03:58
e49	R2-1	dep	i2	2	08:05:58
e50	R2-1	arr	i1	2	08:07:50
e51	R2-2	dep	i6	2	08:15:39
e52	R2-2	arr-pass	i5	2	08:18:40
e53	R2-2	dep-pass	i5	2	08:18:40
e54	R2-2	arr	i4	2	08:23:11
e55	R2-2	dep	i4	2	08:25:11
e56	R2-2	arr-pass	i3	2	08:29:59
e57	R2-2	dep-pass	i3	2	08:29:59
e58	R2-2	arr	i2	2	08:33:58
e59	R2-2	dep	i2	2	08:35:58
e60	R2-2	arr	i1	2	08:37:50
e61	IC1-1	dep	i1	1	08:19:44
e62	IC1-1	arr-pass	i2	1	08:21:03
e63	IC1-1	dep-pass	i2	1	08:21:03
e64	IC1-1	arr-pass	i3	1	08:24:41
e65	IC1-1	dep-pass	i3	1	08:24:41
e66	IC1-1	arr	i4	1	08:29:31
e67	IC1-1	dep	i4	1	08:31:31
e68	IC1-1	arr-pass	i5	1	08:36:11
e69	IC1-1	dep-pass	i5	1	08:36:11
e70	IC1-1	arr	i6	1	08:39:15
e71	IC2-1	dep	i6	1	08:56:23
e72	IC2-1	arr-pass	i5	1	08:59:27
e73	IC2-1	dep-pass	i5	1	08:59:27
e74	IC2-1	arr	i4	1	08:04:07
e75	IC2-1	dep	i4	1	08:06:07
e76	IC2-1	arr-pass	i3	1	08:10:57
e77	IC2-1	dep-pass	i3	1	08:10:57
e78	IC2-1	arr-pass	i2	1	08:14:35
e79	IC2-1	dep-pass	i2	1	08:14:35

eventID	lineID	event_type	event_location	event_frequency	event_time
e80	IC2-1	arr	i1	1	08:15:54
e81	HS1-1	dep	i1	1	08:51:22
e82	HS1-1	arr-pass	i2	1	08:52:44
e83	HS1-1	dep-pass	i2	1	08:52:44
e84	HS1-1	arr-pass	i3	1	08:56:27
e85	HS1-1	dep-pass	i3	1	08:56:27
e86	HS1-1	arr-pass	i4	1	08:00:58
e87	HS1-1	dep-pass	i4	1	08:00:58
e88	HS1-1	arr-pass	i5	1	08:05:26
e89	HS1-1	dep-pass	i5	1	08:05:26
e90	HS1-1	arr	i6	1	08:08:34
e91	HS2-1	dep	i6	1	08:27:04
e92	HS2-1	arr-pass	i5	1	08:30:12
e93	HS2-1	dep-pass	i5	1	08:30:12
e94	HS2-1	arr-pass	i4	1	08:34:40
e95	HS2-1	dep-pass	i4	1	08:34:40
e96	HS2-1	arr-pass	i3	1	08:39:11
e97	HS2-1	dep-pass	i3	1	08:39:11
e98	HS2-1	arr-pass	i2	1	08:42:54
e99	HS2-1	dep-pass	i2	1	08:42:54
e100	HS2-1	arr	i1	1	08:44:16

Table D.7: Output event DataFrame of scenario 3

segmentID	t_before	length	cost	t_needed	t_after	k_same_direction	k_opp_direction
s1	1	2264	844	1	1	0	0
s2	2	4355	1624	2	2	0	1
s3	1	6920	2581	2	2	0	1
s4	1	6799	2536	2	2	0	1
s5	1	5072	1892	2	2	0	1

Table D.8: Output segment DataFrame of scenario 3

interlockingID	station_code	t_before	cost	t_needed	t_after
i1	Jbk	2	25	1	2
i2	Arb	2	51	2	2
i3	Vsg	2	25	2	2
i4	Kp	3	64	3	3
i5	Morp	2	25	2	2
i6	Kba	2	30	1	2

Table D.9: Output interlocking DataFrame of scenario 3

Bibliography

Analys, T. (2024). Railway transport 2023 quarter 4. %5Curl%7Bhttps://www.trafa.se/globalassets/statistik/bantrafik/jarnvagstransporter/2024/jarnvagstransporter-2023-kvartal-4.pdf%7D

Berner, A., Johansson, J., & Andersson, H. (2021). *Mälarbanan, nyttor av dubbelspår* (Report). Sweco. https://www.oslo-sthlm.se/wp-content/uploads/2022/03/kpgqbr1ha9zr0hbivumk.pdf

Bešinović, N., Goverde, R. M., Quaglietta, E., & Roberti, R. (2016). An integrated micro–macro approach to robust railway timetabling. *Transportation Research Part B: Methodological*, *87*, 14–32. https://doi.org/https://doi.org/10.1016/j.trb.2016.02.004

Corman, F., D'Ariano, A., Pacciarelli, D., & Pranzo, M. (2009). Evaluation of green wave policy in real-time railway traffic management. *Transportation Research Part C: Emerging Technologies*, *17*(6), 607–616. https://doi.org/https://doi.org/10.1016/j.trc.2009.04.001

de Bortoli, A., & Féraille, A. (2024). Banning short-haul flights and investing in high-speed railways for a sustainable future? *Transportation Research Part D: Transport and Environment*, *128*, 103987. https://doi.org/https://doi.org/10.1016/j.trd.2023.103987

ERTMS. (n.d.). Ertms in brief.

ERTMS. (2021). Ertms deployment in sweden.

Eurostat. (2024a). Goods transport by rail.

Eurostat. (2024b). Rail transport of passengers.

Goverde, R., & Hansen, I. (2013). Performance indicators for railway timetables. *IEEE ICIRT 2013 - Proceedings: IEEE International Conference on Intelligent Rail Transportation*, 301–306. https://doi.org/10.1109/ICIRT.2013.6696312

Goverde, R., & Scheepmaker, G. (2023). Energy-efficient train timetabling. In *Energy-efficient train operation: A system approach for railway networks* (pp. 69–101). Springer International Publishing. https://doi.org/10.1007/978-3-031-34656-9_4

Gurobi. (n.d.-a). French rail transport giant sncf teams up with gurobi optimization to transform its end-to-end rail freight operations [Accessed on February 2024].

Gurobi. (n.d.-b). Optimization for the transportation industry [Accessed on February 2024].

Hansen, I., & Pachl, J. (Eds.). (2008). *Railway timetabling & operations*. Eurail press.

Herrigel, S., Laumanns, M., Szabo, J., & Weidmann, U. (2018). Periodic railway timetabling with sequential decomposition in the pesp model. *Journal of Rail Transport Planning Management*, *8*(3), 167–183. https://doi.org/https://doi.org/10.1016/j.jrtpm.2018.09.003

Jernbanedirektoratet & Trafikverket. (2022, February). *Feasibility study oslo–stockholm*. %5Curl%7Bhttps://www.oslo-sthlm.se/wp-content/uploads/2022/11/255_Mulighetsstudie_221003_summary_eng.pdf%7D

Kroon, L., Huisman, D., Abbink, E., Fioole, P.-J., Fischetti, M., Maroti, G., Schrijver, A., Steenbeek, A., & Ybema, R. (2008). The new dutch timetable: The or revolution. *Erasmus University Rotterdam, Econometric Institute, Econometric Institute Report*.

Landex, A., Schittenhelm, B., Kaas, A., & Schneider-Tilli, J. (2008). Capacity measurement with the uic 406 capacity method, 55–64. https://doi.org/10.2495/CR080061

Liebchen, C. (2004). Symmetry for periodic railway timetables. *Electronic Notes in Theoretical Computer Science*, *92*, 34–51. https://doi.org/10.1016/j.entcs.2003.12.021

Liebchen, C., & Möhring, R. (2008, January). The modeling power of the periodic event scheduling problem: Railway timetables — and beyond. https://doi.org/10.1007/978-3-540-73312-6_7

Lusby, R., Larsen, J., Ehrgott, M., & Ryan, D. (2011). Railway track allocation: Models and methods. *OR Spectrum*, *33*, 843–883. https://doi.org/10.1007/s00291-009-0189-0

Odijk, M. A. (1996). A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B: Methodological*, *30*(6), 455–464. https://doi.org/https://doi.org/10.1016/0191-2615(96)00005-7

Palmqvist, C.-W., Olsson, N. O. E., & Hiselius, L. W. (2018). The planners' perspective on train timetable errors in sweden. *Journal of Advanced Transportation*, *2018*. https://doi.org/https://doi.org/10.1155/2018/8502819

Polinder, G.-J., Schmidt, M., & Huisman, D. (2020). Timetabling for strategic passenger railway planning. *ERIM Report Series Reference Forthcoming*. https://doi.org/http://dx.doi.org/10.2139/ssrn.3526757

ProRail. (2023, September). Lecture 1.3: Rail traffic simulation in practice [Class lecture given in CIEM6301: Railway Traffic Management, Delft University of Technology, Delft, the Netherlands, September 2023.].

Serafini, P., & Ukovich, W. (1989). A mathematical model for periodic scheduling problems. *SIAM J. Discret. Math.*, *2*, 550–581. https://api.semanticscholar.org/CorpusID:19409360

Solinen, E. (2022). *Generella konstruktionsregler för tågplanekonstruktion*. Trafikverket.

Sweco. (2017). Oslo-stockholm wider socioeconomic benefit analysis 2040 [Project number: 7002339000]. %5Curl%7Bhttps://www.oslo-sthlm.se/wp-content/uploads/2024/02/lhkuxtegsmbdv8rht7lw#:~:text=Sweco%20can%20conclude%20that%20the,2016%C2%B4s%20price%20level.%7D

Trafikverket. (n.d.-a). Improved capacity [Accessed on March 2024].

Trafikverket. (n.d.-b). Mälarbanan [Accessed on July 2024].

Trafikverket. (n.d.-c). Marknadsanpassad planering av kapacitet (mpk) – arbetssätt och verktyg för framtiden [Accessed on March 2024].

Trafikverket. (2021). Trafikuppgifter jarnvag t22 och bullerprognos 2040.

Trafikverket. (2022). Proposal national plan for transport infrastructure 2022-2023 [Accessed on March 2024].

Trafikverket. (2023a, February). New main lines - a new generation railway. %5Curl%7Bhttps://bransch.trafikverket.se/en/startpage/planning/new-main-lines---a-new-generation-railway/#:~:text=The%20Swedish%20Transport%20Administration%20has,Bor%C3%A5s%20and%20H%C3%A4ssleholm%E2%80%93Lund%20projects.%7D

Trafikverket. (2023b, December). Network statement 2024 [Edition 2023-12-15. For deliveries from 2023-12-10 to 2024-12-14 made by Trafikverket.].

Trafikverket. (2024a). Network statement 2024 (2024-06-26). %5Curl%7Bhttps://bransch.trafikverket.se/contentassets/7476ee2129b2457da7163d77e0963edb/ns_2024_2024-06-26.pdf%7D

Trafikverket. (2024b, May). Fyrspåret malmö–lund.

Trafikverket & Jacobs. (2021, March). *New main lines: Cost benchmarking study* (tech. rep.). Trafikverket and Jacobs. https://bransch.trafikverket.se/contentassets/60ecb96cb94a4cac994aae8bea032992/18-maj-2021/new-main-lines---ru205---cost-benchmarking-study.pdf

University of Birmingham. (2013). On time. a framework for developing an objective function for evaluating work package solutions (cost function).

Van Wee, B., & Banister, D. (2016). How to write a literature review paper? *Transport Reviews*, *36*(2), 278–288. https://doi.org/https://doi.org/10.1080/01441647.2015.1065456

Varför är oslo-sthlm viktigt? [Accessed on July 2024]. (n.d.). *Oslo-Sthlm 2.55*.

Vigren, A. (2017). Competition in swedish passenger railway: Entry in an open access market and its effect on prices. *Economics of Transportation*, *11-12*, 49–59. https://doi.org/https://doi.org/10.1016/j.ecotra.2017.10.005

Warg, J. (2016). Timetable evaluation with focus on quality for travellers [QC 20160902].

Williams, H. P. (2013). *Model building in mathematical programming* (5th ed.). John Wiley & Sons.

Zhang, X., & Nie, L. (2016). Integrating capacity analysis with high-speed railway timetabling: A minimum cycle time calculation model with flexible overtaking constraints and intelligent enumeration. *Transportation Research Part C: Emerging Technologies*, *68*, 509–531. https://doi.org/https://doi.org/10.1016/j.trc.2016.05.005