Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Informatik

Train Schedule Optimization with focus on Robustness

Bachelor thesis FS2020

Kevin Reust

Supervisor: Dr. Donnacha Daly

Lucerne University of Applied Sciences and Arts - Department of Information Technology
June 5, 2020
Version 1.0

Bachelor thesis at the Lucerne University of Applied Sciences and Arts - Department of Information Technology

Titel: Train Schedule Optimization with focus on Robustness

Student 1: Kevin Reust

Student 2: -

Course of studies: BSc Informatik

Graduation year: 2020

Supervisor: Dr. Donnacha Daly

Expert: Sandro Cilurzo

Coding / classification of the work:

× **A: Insight (normal case)**

☐ **B: Consultation** (Duration: Year / Years)

☐ **C: Restriction** (Duration: Year / Years)

Affidavit

I/we hereby declare that I/we have prepared the present work independently and without unauthorized assistance, that I/we have indicated all sources, literature and other aids used, that I/we have marked any passages taken either literally or in terms of content as such, that I/we will protect the client's interest in confidentiality and that I/we will respect the copyright regulations of the Lucern University of Applied Sciences and Arts (see market sheet "Studentische Arbeiten" on MyCampus).

Place / Date, Signature: Buchs ZH / 05.06.2020 K Reust

Place / Date, Signature: _____

Submission of the work to the portfolio database

Confirmation signature student
I confirm that I have correctly filed the bachelor thesis on the portfolio database according to the information sheet. I have handed over the responsibility as well as the permissions, so that I cannot make any changes or upload further files.

Place / Date, Signature: Buchs ZH / 05.06.2020 K Reust

Place / Date, Signature: _____

Acknowledgment
I would like to thank all those who have supported and motivated me during the implementation of this work. A special thanks goes to my supervisor Dr. Donnacha Daly, who supported me throughout the whole project with advice and gave me inputs on how I can present my work more effectively in a professional environment. I would also like to thank my family, who made my studies possible in the first place and who supported me throughout my studies.

Work received (to be filled in by the secretariat):

Rotkreuz, den _____ Visum: _____

Abstract

Scheduling of railway networks is an old problem, which is extremely well studied and understood. Switzerland is clearly a great example, having a railway network which functions very well in delivering punctual, densely connected and frequent train services across the nation. Nevertheless, there are two pressing issues which confound scheduling planners and need good solutions for future scheduling:

(a) The number of passengers is increasing at a fast rate, which requires more trains running more frequently on the same network.

(b) The impact of delays in a more highly utilized network is more significant, with cascading effects causing rolling delays in over congested parts of the network.

This thesis is dedicated to the investigation of methods to optimize train schedules so that the resulting schedules can absorb such delays and stress in the network.

To achieve this, a simplified, discrete train network simulation was developed and combined with an genetic algorithm to produce robust or efficient timetables. The genetic algorithm was then used to perform a series of experiments with star networks of different sizes (6, 9, 10 and 14 stations), whereby the schedules were optimized once for robustness and again for efficiency. The resulting performance of the robust and the efficient schedules was then compared. The comparison showed, that efficient schedules outperformed their robust counterparts under normal network conditions, while robust schedules were able to maintained their performance under congested conditions much longer than efficient schedules.

This trade-off between efficient and robust train systems was then discussed in terms of the benefits that an analysis of the trade-off could bring to train schedule planners.

Contents

1. Introduction to the Importance of Robustness in Train Scheduling

1.1. Problem Definition

Railway systems are an important part of the modern economy. A statistic of the Swiss Federal Statistical Office from 2017 showed, that 17% of the Swiss work commuters and 41% of the people, who commute for study and training purposes, use the railway system as their main means of transport. (Bundesamt für Statistik, 2017).

However, fast rising passenger numbers pose a big challenge to the schedule planners, as the used infrastructure cannot be expanded much further (see Table 1.1). Consequently, railway companies utilize the existing infrastructure more heavily by running more trains, more frequently on the same network. At the same time, the effects of delays are more significant in a more congested network, with cascade effects leading to rolling delays in congested parts of the network.

	2018	2019	Δ %
Increasing number of trains on the network (numbers/day)	10'708	10'991	+ 2.6%
Increasing number of passenger (Mio. Pkm / year)	18'608	19'689	+ 5.8 %
More stress on the network results in less punctuality (Customer punctuality)	90.1%	89.5%	- 0.6%
Number of tracks can't increase much further (Network km operated)	3'228	3'236	+ 0.3%

Table 1.1.: Development of selected key figures in Swiss railway system from 2018 to 2019 ("SBB Statistics", 2020)

The goal of this project is to explore different methods for optimizing train schedules in a simplified simulated setting, such that the resulting schedules can absorb delays and stress in the network. The optimized schedules should then be used to examine the trade-off between efficient and robust train networks.

Thesis: Simulation based optimization can be used to produce train schedules which have a suitable trade-off between efficiency and robustness.

1.2. Expected Results

The following results are produced as part of this project:

1. A simplified, scalable model that can be used flexibly for different types of train networks and traffic uses.
2. Definitions of metrics to measure the robustness of the network.
3. An implementation for an optimizer that can be used to optimize schedules for a given network in terms of robustness.
4. Results of experiments with the model under different conditions with emphasis on the trade-off between robustness and efficiency.

1.3. Content structure and methodical procedure

The content of this document is structured as follow:

- In chapter 2 the current state of research is discussed. An introduction to public rail transport planning will be given, before the most important research topics such as the periodic event scheduling problem, optimization and complexity are discussed.
- In chapter 3 the concept of simulation based optimization is discussed and it is shown with which architecture and model the problem was approached. Additionally, a comparison between different optimization methods and their applicability for the simulation-based approach is performed.
- Chapter 4 describes the implementation and the most important functional principles of the train network simulator. Furthermore the genetic optimizer and its operations are presented. In the last part of the realization the results of experiments, which were conducted on different simulated train networks, are shown.
- In the discussion, the results obtained will be elaborated on. A comparison is made between the performance of optimized schedules in different networks. In addition, the trade-off between robust and efficient schedules is shown.
- In the last part of the paper the conclusion is formulated and a proposal for further research is made.

The entire project was planned and executed using a waterfall model. The corresponding project management plan can be found in the appendix.

2. The Train Scheduling Problem and recent Approaches to solve it

This chapter gives a short introduction into public rail transport planning and the process of train schedule planning before explaining some basic concepts and recent approaches to the train scheduling problem.

2.1. Public Rail Transport Planning

Modern train networks are highly complex systems consisting of large amounts of infrastructure and personnel (Table 2.1). Due to their importance for the economy they are the focal point of many different interest groups; companies, politicians, commuters, employees, scientists, etc. The goal of public rail transport planning is to evaluate those interests and produce a long-term strategy for transporting passengers and goods efficiently and safely between several locations with the available resources. To achieve this goal, various interdependent sub-problems like scheduling, network design, infrastructure expansion or routing must be solved (Lindner, 2000, p.1).

1.32	m passengers per day
10'991	Trains on network per day
3236	km of network
32'535	yearly average headcount (full-time equivalent)
4.37	bn CHF public-sector funding
5.65	bn CHF purchasing volume

Table 2.1.: Facts and figures of the Swiss railway company SBB *(All numbers are collected from the SBB Statistics Portal at reporting.sbb.ch)*

One of the dominating challenges in this process is the time it takes to create or overhaul a rail transport plan. For instance, the Expansion step 2035 (STEP ES 2035) of the Swiss Railway Company SBB is based on a report from April 2014 (BAV, 2014), was accepted by the Swiss National Council in June 2019 (Strategisches Entwicklungsprogramm Eisenbahninfrastruktur. Ausbauschritt 2035, 2018, p. 18.078) and should be realized in 2035. At the same time, the report predicts an increase in passenger demand of 51% until 2040. That success of the rail transport plan is therefore highly dependent on the predictions made in the beginning of the process.

As a result of the size, duration and complexity of the planning process, it is often divided into several steps (additionally see Bussieck, 1998, p.6). The diagram 2.1 visualizes the hierarchical decomposition of the planning process with the associated planning levels.

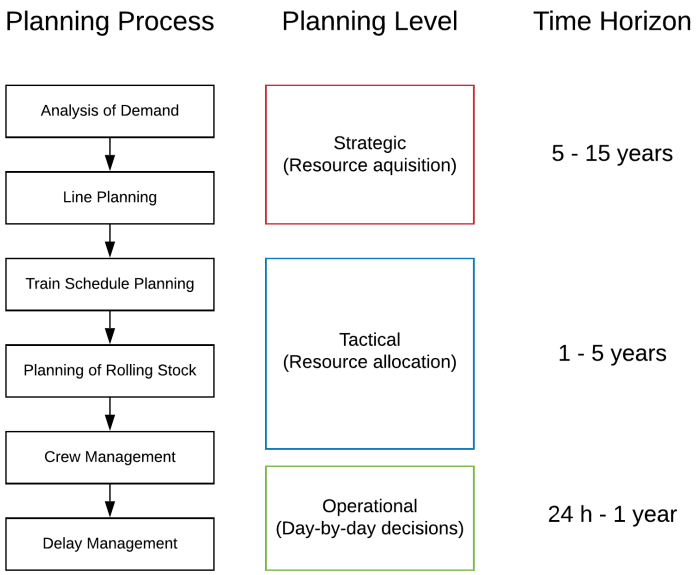


Figure 2.1.: Hierarchical planning process with planning levels
(Based on Lindner, 2000, Fig. 1.2)

2.1.1. Rail Transport Planning Process

At the begin of the planning process the demand has to be analysed. The most important information needed is a forecast of future passenger demand, but other factors such as legal requirements and existing infrastructure should also be taken into account. The result is generally in form of a origin-destination matrix (OD-Matrix) where every entry (row, column) indicates the demand from location *row* to location *column* (Bussieck, 1998, p. 7).

In the next step the acquired information are used to determine the train lines. Train lines consists of a route and a corresponding periodicity, where the periodicity defines how often the line is served and the route describes the path through the network (Lindner, 2000, p. 5, Bussieck, 1998, p. 8). For instance, the period on the SBB long distance routes is currently 30 minutes. The problem of finding suitable lines for a given network and periodicity is called line planning problem (see Goossens, 2004). On the basis of the determined routes, the infrastructure may have to be expanded or altered.

Subsequently in train schedule planning, all arrival and departure times of the planed lines are determined. As this step is the main focus of this paper it is further described in Section 2.1.2.

After the train schedule planning, the rolling stock must be configured and assigned to a line according to the given demand and infrastructure (track width, weight/length restrictions, etc.). Similarly, staff has to be recruited, trained and assigned to operate rolling stock, infrastructure, etc (Lindner, 2000, p. 6).

The last category consists of short-term decisions which are needed to react to unexpected events like delays, train breakdown or damaged infrastructure. Good long-term planning like buffer times or redundant infrastructure help to resolve those short-term problems without changing the system extensively.

All the described processes are hard to solve and have their own specialised models and algorithms to tackle them. Because of the dependencies between the processes, it is possible that an optimal solution in one sub-process (e.g. Train Schedule Planning) can therefor lead to an overall worse final result. To solve the sub-problems sequentially is on the other hand much better manageable than trying to solve the whole planning problem at once (Lindner, 2000).

2.1.2. Train Schedule Planning

Train schedule planning is an integral part of the rail planning process as train schedules can be perceived directly by the customers of railway companies. When talking about schedules, they are generally divided into the categories 'non-periodic' and 'periodic' where a periodic schedule runs at consistent intervals and non-periodic schedules are irregular and purely demand driven. While highly congested, urban networks (e.g. underground trains, trams, etc.) historically almost always used periodic schedules, they were much less common in early long-distance train connections (Lindner, 2000, p. 7). For example, the SBB did not use a Switzerland-wide, periodic schedule prior to 1982 (Matti, 2019, p. 82).

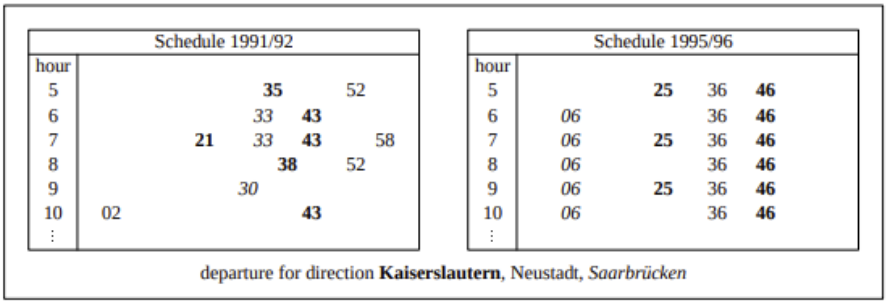


Figure 2.2.: Comparison of non-periodic (l.) and periodic (r.) schedule (Lindner, 2000, Fig. 1.4)

An even more sophisticated type of schedule is the integrated fixed interval schedules (IFIS) (Rudolf Gäbertshahn, 1993, pp. 357–362). IFIS's are periodic schedules in which the trains meet at predefined nodes at a similar time frame. This enables passengers to switch to a connecting train with less waiting time at the station. An example for an node station in an IFIS can be seen in Figure 2.3.

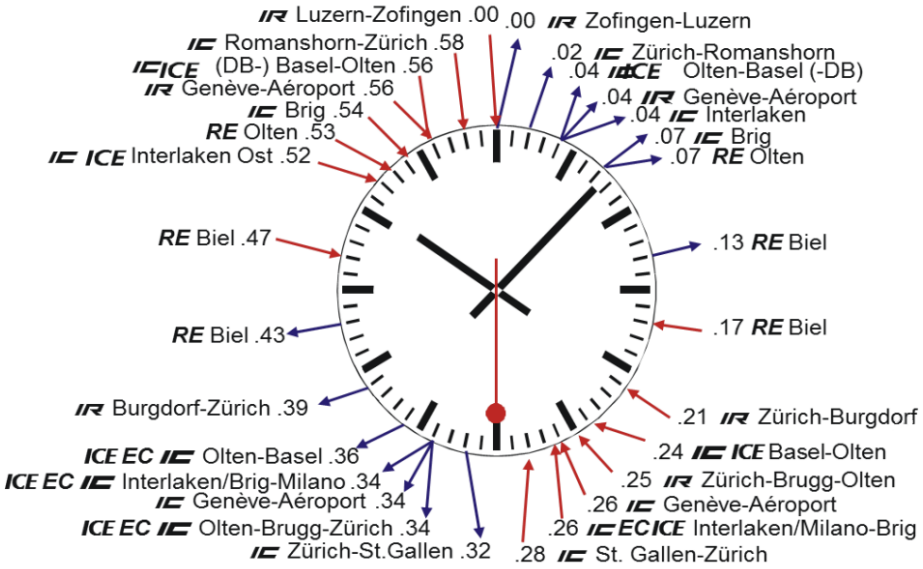


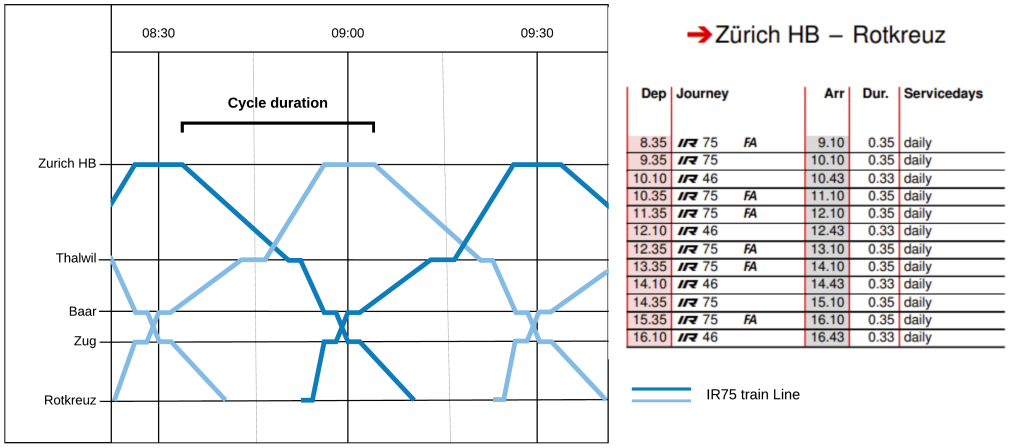
Figure 2.3.: Visualisation of arrival and departure times of long-distance lines at Bern Main Station in 2016 (Heidron Buttler, 2017)

Regardless of the type used, a train schedule generally contains information about arrival and departure time of a lines at certain points in the train network. In simple networks, those points are usually station while larger networks use additional operating points (e.g. signals, junctions) to coordinate multiple lines (Lindner, 2000, p. 6). To illustrate, the GIS dataset of SBB contains 1'769 such operating points (Infrastruktur, 2020) which are used to create a higher resolution schedule.

Additionally, a schedule has to satisfy some constrains to be valid. Possible constrains for train schedules are dwelling times at stations, travel duration, safety distances between trains or fixed arrival and/or departure intervals for certain stations and lines. Those constraints, together with the information about the network, can then be used to calculate valid schedules. Because this process usually produces multiple valid schedules, performance indicators or objectives are introduced to the planning process. The performance indicators are used to rate the schedules and are selected based on the needs of the organisation which operates the networks. A commonly used performance indicator is travel time (Lindner, 2000, p. 7). Further performance indicators are described in section 2.1.3. This process of finding an optimal timetable in a given network with given constraints is referred to as the

train schedule problem (TSP). The most common way to model the TSP for periodic timetables is in form of the Periodic Event Scheduling Problem (PESP) (Berkan, 2009, p. 28). Because of the importance of the PESP it is further described in section 2.2.

Since the 2000s the role of computer software in the planning process became more important. Most train companies use several programs to develop their timetables. For instance, SBB uses OnTime[1], OpenTrack[2] as well as performance data of their previous timetables, to make improvements to their schedules (Heidron Buttler, 2017, p. 52). Those or similar programs output predictions about the performance of a given timetable or generate feasible timetables based on a set of input parameters. The results are then visualised in the form of a time space diagram (See figure 2.4) which can be used by experienced train schedule planners to find good schedules.



Dep	Journey		Arr	Dur.	Servicedays
8.35	IR 75	FA	9.10	0.35	daily
9.35	IR 75		10.10	0.35	daily
10.10	IR 46		10.43	0.33	daily
10.35	IR 75	FA	11.10	0.35	daily
11.35	IR 75	FA	12.10	0.35	daily
12.10	IR 46		12.43	0.33	daily
12.35	IR 75	FA	13.10	0.35	daily
13.35	IR 75	FA	14.10	0.35	daily
14.10	IR 46		14.43	0.33	daily
14.35	IR 75		15.10	0.35	daily
15.35	IR 75	FA	16.10	0.35	daily
16.10	IR 46		16.43	0.33	daily

Figure 2.4.: Visualisation of the IR75 line between Zurich HB and Rotkreuz in form of a time-space diagram

Even though software for train schedule planning became increasingly sophisticated over the years, the entire process still depends heavily on the train schedule planners. However, the emergence of machine learning in earlier human-dominated areas, such as strategy games (“Match 1 - Google DeepMind Challenge Match”, 2016), has led to new approaches to solving the time scheduling problem. Some of those approaches are described in section 2.3.

[1] Software which uses a stochastic assessment method to predict the punctuality of future schedules based on reference data

[2] Software to model, simulate and analyze rail systems (www.opentrack.ch)

2.1.3. Performance Indicators for Train Schedules

To evaluate and compare the performance of train schedules, one or multiple performance indicators have to be used. Those performance indicators have a big impact on the quality of the final results as they determine the properties which are optimized during the schedule planning process. Therefore every train company can have different implementations of performance indicators. As an example, a timetable for passenger lines is likely to weight travel time more heavily, while a timetable for freight trains is more focused on infrastructure utilisation. As part of the European ON-TIME project, Goverde and Hansen described five general types of timetable performance indicators and introduced a schema to categorise them. (Goverde & Hansen, 2013).

Timetable Design Levels

Goverde and Hansen suggested, that the design of a train timetable can be divided into four levels with increasing performance when exposed to delays and disruptions (Goverde & Hansen, 2013, p. 5-6).

	Deterministic	Stochastic
Macroscopic	Level 1 - Stable	Level 3 - Robust
Microscopic	Level 2 - Feasible	Level 4 - Resilient

Table 2.2.: Timetable design levels depending on timetabling methods *(Goverde & Hansen, 2013, Fig. 2.)*

Level 1 Timetables which are stable and therefore able to handle initial and primary delays which can be caused by small disruptions such as door malfunctions. Stability is the minimum requirement for any timetable to be considered good (Goverde & Hansen, 2013, p. 5).

Level 2 Timetables which are feasible or conflict-free and therefore guarantee that train path conflicts are resolved in the timetable (under normal conditions). With such a timetable, no train in the network would have to stop unplanned because a critical resource is occupied by another train. To test for feasibility requires a microscopic view of the railway system including information about network layout, train behaviour and safety restrictions (Goverde & Hansen, 2013, p. 5).

Level 3 Timetables which are robust and therefore can absorb a certain amount of stress (statistical variation) without collapsing. To test a timetable for robustness requires stochastic data such as distributions of travel times or delays. Those information are then used to add a sufficient amount of time supplements and buffer times without being too conservative (Goverde & Hansen, 2013, p. 5). The Software OnTime, which was mentioned in the section 2.1.2 is used by SBB for such stochastic assessments on their networks.

Level 4 Timetables which are resilient and therefore require to include real-time traffic management to be assessed as well. A resilient timetable has to be stable and robust only in combination with dynamic traffic management.

This classification method was used by Goverde and Hansen in 2013 to assess the timetables of several European countries with the following results (Goverde & Hansen, 2013, p. 6):

- Level-0: France, Italy
- Level-1: Netherlands, United Kingdoms
- Level-2: Germany
- Level-3: Switzerland, Sweden

Timetable Performance Indicators

The following table provides an overview of the performance criteria introduced in Goverde and Hansen, 2013, chapter II:.

Infrastructure occupation	Share of time required to operate trains on a given railway infrastructure according to a given timetable pattern. It measures how much infrastructure capacity a timetable pattern occupies where the theoretical capacity of a track section is defined as the maximum number of trains that can be operated in a given time period. *(see Goverde and Hansen, 2013, Chapter II-A, p.2)*
Timetable feasibility	Ability of trains to adhere to their given schedule. A timetable is feasible if the individual processes are realizable within their scheduled process times (e.g. running time, dwell time, turn-around time, etc) and the scheduled train paths are conflict free. If a schedule is feasible no train has to stop or slow down unscheduled due to a conflict (e.g. other train uses track or platform) under normal conditions. *(see Goverde and Hansen, 2013, Chapter II-B, p.2-3)*
Timetable stability	Ability of a timetable to absorb initial and primary delays so that delayed trains return to their scheduled train paths. If a timetable is stable, any train delay can be absorbed by the time allowance in the timetable without active dispatching (e.g. A five minute departure delay of Train A can be absorbed by time supplements within the schedule). *(see Goverde and Hansen, 2013, Chapter II-C, p.3)*
Timetable robustness	The ability of a timetable to withstand design errors, parameter variations, and changing operational conditions. Timetable robustness aims at minimizing the occurrence of primary and secondary delays *(Secondary delays are caused by the propagation of primary delays to other lines in a network)*. While a primary delay only occurs if a realized train process time exceeds the scheduled process time (running, dwelling, turning), secondary delay only occurs if a given delay exceeds the buffer time between two train paths at a critical block section. Robustness is strongly related with stability as the time supplements and buffer times that are built in the timetable to prevent primary and secondary delays (robustness) are also used to recover train delays. *(see Goverde and Hansen, 2013, Chapter II-D, p.3-4)*

Timetable resilience	Flexibility of a timetable to prevent or reduce secondary delays using dispatching (re-timing, re-ordering, re-routing). For a timetable to be resilient, it has to be flexible enough so that the traffic management can react and overcome bigger delays in the system. For Example, by using a small arrival time interval for multiple lines before a critical path, the traffic manager has the possibility to change the order in which the trains enter the critical section in case of a delay. *(see Goverde and Hansen, 2013, Chapter II-C, p.4-5)*

Table 2.3.: Overview of the timetable performance criteria introduced in "Performance Indicators for Railway Timetables" *based on (Goverde & Hansen, 2013, Chapter II)*

2.2. The Periodic Event Scheduling Problem (PESP)

The PESP was introduced by Serafini and Ukovich in 1989 as a generalisation of the periodic Train Schedule Problem (TSP) and was defined similar to figure 2.4 (Serafini & Ukovich, 1989). The underlying idea of the PESP is to create a schedule for only one period (e.g. 1h) which can then be repeated for the whole day (Berkan, 2009, p. 28). The feasibility (ref Sec. 2.1.3) of the schedule is guarantied by introducing a set of interval constraints (e.g. safety regulations, minimum dwelling times) and the goal is to find a schedule that satisfies all those constraints. Serafini and Ukovich proved in their work that the PESP is NP-Complete (Serafini and Ukovich, 1989, Lindner, 2000).

Given:	T	time period
	$\mathcal{E}$	set of periodic events
	C	set of periodic interval constraints for $\mathcal{E}$
Find:	$\pi : \mathcal{E} \rightarrow \mathbb{R}$	schedule satisfiying all constraints from C or state infeasibility

Table 2.4.: Formal description of the Periodic Event Scheduling Problem *(Lindner, 2000, Fig 2.2)*

The downside of the PESP is, that it is not able to produce any results in systems that have no feasible schedules. An example for such a situation would be if a constraint for minimum buffer time was introduced into the problem to generate more stable timetables. It is now possible that at some congested stations the buffer time cannot be incorporated into the schedule without violation other constraints. To deal with such situations, some constraints have to be relaxed which makes the whole problem even more complex because it may not be obvious, which constraints have to be relaxed to get valid schedules (Lindner, 2000, p. 17).
Another downside of the PESP is, that it provides no information whether a found schedule is optimal. It is therefore necessary to create an optimization model by introducing metrics like those described in section 2.1.3.

Although mathematical models based on the PESP are able to produce optimal timetables,

they often perform poorly on larger problems. This is due to the underlying complexity of the PESP. Recent approaches to mitigate this problem by using machine learning algorithms will be discussed in the next section.

2.3. Recent Approaches for solving Schedule Problems

The previous section described that, due to the NP-completness of the problem, finding optimal schedules using a mathematical model (e.g. PESP) does lead to impractical computing times when used on large networks with many constraints. However in real world networks it can often be sufficient to have a near-optimal solutions. Many machine learning techniques such as reinforcement learning or genetic algorithms have showed promising results in approximating optimal solutions for complex problems and large search spaces. A prominent example is the 2006 match between AlphaGo and the world-renowned Go player Lee Sedol which ended in an 4:1 win for AlphaGo (Borowiec, 2016).

The following two subsections showcase two promising examples of alternative approaches to scheduling problems.

2.3.1. Genetic Algorithm

In 2015, Arenas et al. suggested a genetic algorithm (GA) approach for solving the train timetabling problem and compared the performance of their GA with an mixed integer programming (MIP) approach. They observed, that the GA was slower than the MIP approach in obtaining near-optimal solutions for small request numbers (train journeys and time horizons). However, with increasing complexity the performance benefit of the GA became more evident. While the MIP approach always produced better schedules (see profit in Table 2.5) the GA could create near optimal schedules in considerably less time, when dealing with complex systems (Arenas et al., 2015). The original results can be seen in table 2.5,

Req. Journeys	Time Horizon	Const. Count	CPLEX Profit	CPLEX Time	Max. Eval.	GA Profit	GA Time
8	45	171	3500	0.05s	1.5K	3500	0.08s
11	77	247	4226.13	0.73s	11K	4193.66	2.43s
14	72	340	5789.36	0.15s	10K	5783.67	3.69s
15	22	409	5467.72	1.03s	16K	5456.06	5.91s
46	86	3524	16949.39	1m6s	10K	15294.79	28.23s
55	79	4721	18009.22	22m19s	55K	17460.15	3m44s
106	55	17536	25644	1h	70K	23508.35	17m13s

Table 2.5.: Results of experiments conducted by Arenas et al. (CPLEX is Software used as MIP solver) *(Arenas et al., 2015, Table 1)*

2.3.2. Reinforcement Learning

Khadilkar, 2019 introduced a table-based Q-learning approach to generate schedules for large scale networks, which should be closer to optimal than with heuristic approaches, while

still maintaining low computation times. He tested his algorithm on four different scenarios from which two were real world train lines in India. A comparison with the Fixed Priority Travel Advance Heuristic (TAH-FP) and Critical First Travel Advance Heuristic (TAH-CF) showed, that the reinforcement learning agent was able to outperform the heuristics in all four scenarios (Khadilkar, 2019, pp. 734–736).

2.4. General Overview of Optimization Methods

2.4.1. Integer Programming

Mathematical optimization problems in which all variables only take discrete integer values are called integer programms (IP's). IP's are most prominent in decision problems because they are discrete (e.g. yes/no, etc.) and have a finite number of options. A program where some variables take integer values while others can take fractional values is called a mixed-integer program (MIP) (Beasley, n.d.). An example for an MIP could be a program which decides whether or not to invest in an obligation (discrete decision) and if yes how much to invests (continuous value) to optimize the profit. The mathematical formulation for an IP problem is as follow (Taha, 2014, p. 3):

maximize (or minimize)

$$z = g_0(x_1, x_2, \ldots, x_n)$$

subject to

$$g_i(x_1, x_2, \ldots, x_n) \left\{ \begin{array}{c} \leqslant \\ = \\ \geqslant \end{array} \right\} b_i, \quad i \in M \equiv \{1, 2, \ldots, m\}$$

$$x_j \geqslant 0, \quad j \in N \equiv \{1, 2, \ldots, n\}$$

$$x_j \text{ an integer}, \quad j \in I \subseteq N$$

2.4.2. Genetic Algorithms

Genetic Algorithms are a subset of evolutionary algorithms used to find sufficiently good enough solutions to an optimization problem by modelling the process of natural selection. They do this by performing operations such as selection, crossover and mutation, which are inspired by biological processes normally found in the field of genetics. In figure 2.5 some examples for the most common genetic operations are visualised.

Genetic algorithms usually start by generating a base population of (often random) solutions. Those solutions are then evaluated using a fitness function. In the selection phase, the fittest n members of the population are chosen to generate a new generation of solutions. This is achieved by using genetic operations such as crossover or mutations to generate slight variations of the selected members. This process is repeated until a sufficiently good solution is found or the fitness of the following generations is not increasing significantly anymore.

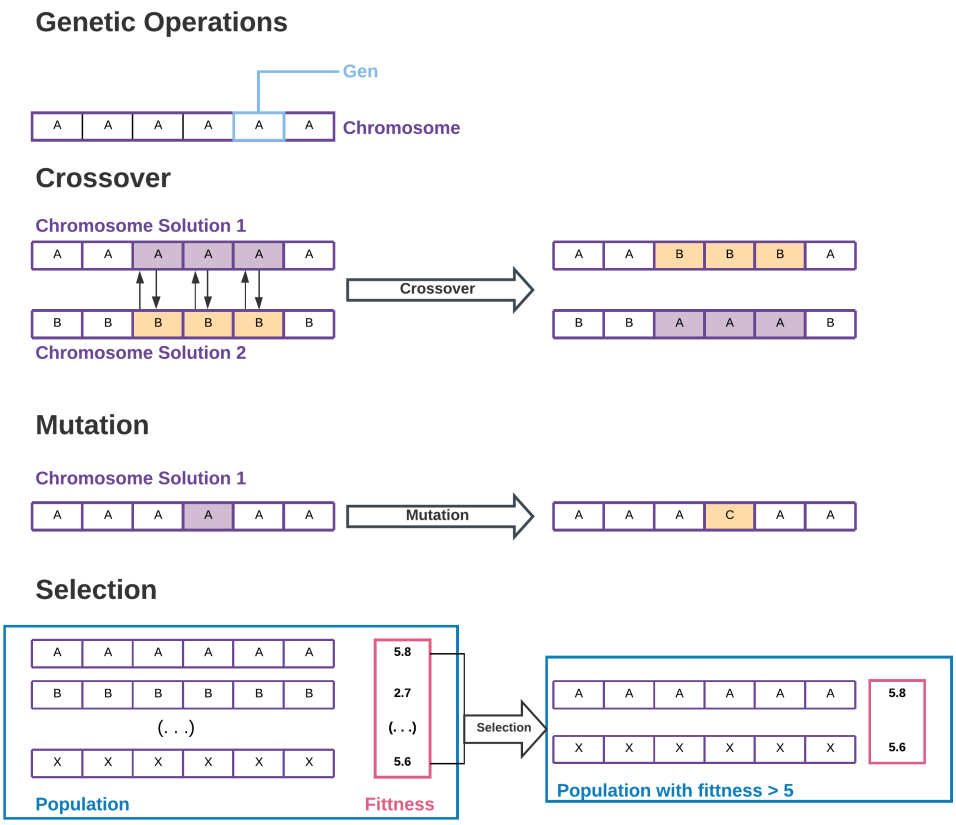


Figure 2.5.: Visualisation of common genetic operations used by genetic algorithms

Because the performance of the algorithm is very dependent on the initial population, it is common that the process is repeated multiple times.

2.4.3. Reinforcement Learning

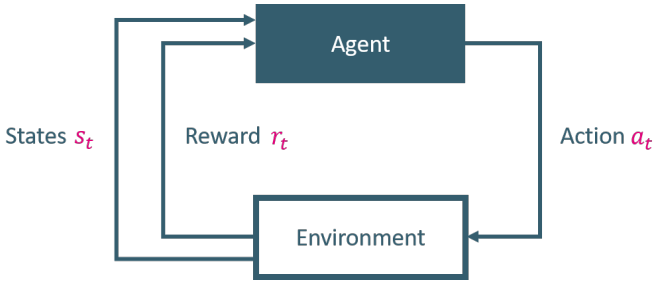


Figure 2.6.: Visualisation of a reinforcement agent and its interactions with the environment

Reinforcement Learning (RL) is a sub domain of machine learning which focuses on learning what action to take in a given situation so that a numerical, cumulative reward signal is maximized (Sutton & Barto, 2015, p. 2). In the standard RL model, an agent can sense its environment by a set of sensors and based on the perceived state it can select an action. The action changes the environment of the agent which leads to a new state and a reinforcement/reward signal to the agent. The job of the agent is to explore the action space and find a policy for choosing actions in each state so that it can maximise its cumulative reward on the long run (Sutton & Barto, 2015).

A big challenge for RL algorithms is the trade-off between exploration and exploitation. If the agent only exploits, it will always use the action policy which it thinks will give the highest reward without exploring other possible policies which may or may not lead to better long-term reward. The other extreme is an agent which always explores without ever exploiting its knowledge. Such an agent will exhaustively search the whole action space and ignore the reward. One common method to deal with the trade-off between exploration and exploitation is ϵ-greedy where $0 < \epsilon < 1$ is a hyper parameter controlling the proportions between exploration and exploitation. With ϵ-greedy the agent will choose the action which it thinks yields the best long-term reward with a probability of $1 - \epsilon$ while exploring new actions with a probability of ϵ.

2.5. Introduction to Computational Complexity

When designing programs or algorithms, we are generally interested in the time it takes them to compute a solution (time complexity) and the amount of memory they need to operate (space complexity). Because those values are highly dependent on the parameters used (e.g. sorting an already sorted list vs. a completely random one), the complexity of an algorithm is determined by the worst case scenario. Likewise, the time or space required by an algorithm often depends on the size of the input. Therefore the complexity of an algorithm is expressed as a function of its input size n. In most notaion to define the worst case complexity of an algorithm is "big O":

"An algorithm is said to have worst case running time (or complexity) of $O(f(n))$ if there are constants $c > 0$ and $n_0 \in \mathbb{N}$ such that the running time does not exceed $c \cdot f(n)$ for each instance of size $n \geq n_0$." (Lindner, 2000, p. 101)

2.5.1. P and NP Complexity classes

Complexity classes are a way to group problems which have similar, resource-based complexity. Two complexity classes which are fundamental for decision and optimization problems are P and NP. The class P is a set of all problems which can be solved using polynomial time (e.g. $O(n^2)$, $O(n^3$, ...) while NP is a set of all problems, where a given solution for a given problem can be verified in polynomial time. Because of the fact that, if a problem can be solved in polynomial time it can also be verified in polynomial time (by solving it), all problems in P are also in NP (Lindner, 2000, p. 102). Although not proven (P-NP Problem)

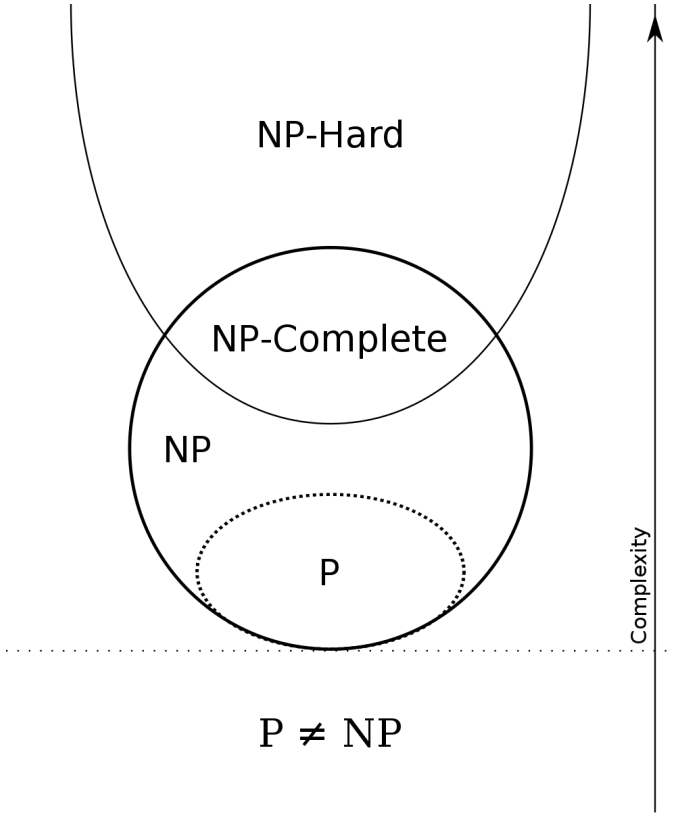


Figure 2.7.: Euler diagram for P, NP, NP-complete, and NP-hard set of problems. (Esfahbod Behnam, 2007)

it is generally assumed that P $\neq$ NP, which is essential for many applications like one-way functions in asymmetric cryptography.

A third important complexity class for decision problems is NP-Complete. A Problem is only in NP-Complete if it is in NP and if any other problem in NP can be reduced to it in polynomial time. If there would be an algorithm which is able to solve an NP-Complete problem in polynomial time it would be posible to solve any NP problem in polynomial time because any NP Problem can be reduced to the NP-complete problem (Lindner, 2000, p. 102).

3. Concept and Model for creating robust Timetables with Simulation-based Optimization

3.1. Simulation Based Approach

Optimization problems are often based on a mathematical formulation and a set of constrains. This allows to utilize exact algorithms such as MIP solver (see section 2.4.1) to find, given enough time, globally optimal solutions. But there are some problems or systems which are prohibitively complex to model purely mathematically. A prominent example for such a system is weather and climate forecasting. Although many physical processes, which have an influence on the weather, are known and can be described mathematical, the system as a whole is so globally interconnected and contains so many variables, that it is currently infeasible to model it perfectly and so simulation methods are preferred.

In cases where the system under investigation is overly complex, simulation-based optimization can be applied. In simulation-based optimization, the mathematical objective function is replaced by a simulation which approximates the functionality of the simulated system. The quality of this approximation varies depending on the application, since a higher accuracy also results in a longer calculation time. To illustrate, while a simple mobile game flight simulator can be accurate enough to showcase the basics of 3-axis control, a more sophisticated simulation is needed to train AI algorithms or pilots to land a specific aircraft under changing weather conditions (see also Koller and Wullschleger, 2019). In general, simulation-based optimization is a numerical optimization and only approximates optimal solutions, but does this more efficiently than exact methods, when applied on complex systems.

As established in the previous section, train networks are also highly complex and dynamic systems which additionally have a stochastic nature (train breakdowns, accidents, passenger behaviour, etc.). Although there are mathematical formulation for schedule optimization problems (see Section 2.2), they are often very general and suffer under the points mentioned earlier. Moreover, simulations can be extended more easily and, when programmed correctly, could also be used for different projects in the same domain (e.g. implementing an AI based traffic management). Because of those reasons, a simulation based approach was chosen for this project.

While the detailed description of the implementation can be found in section 4.1, a basic concept is shown in figure 3.1.

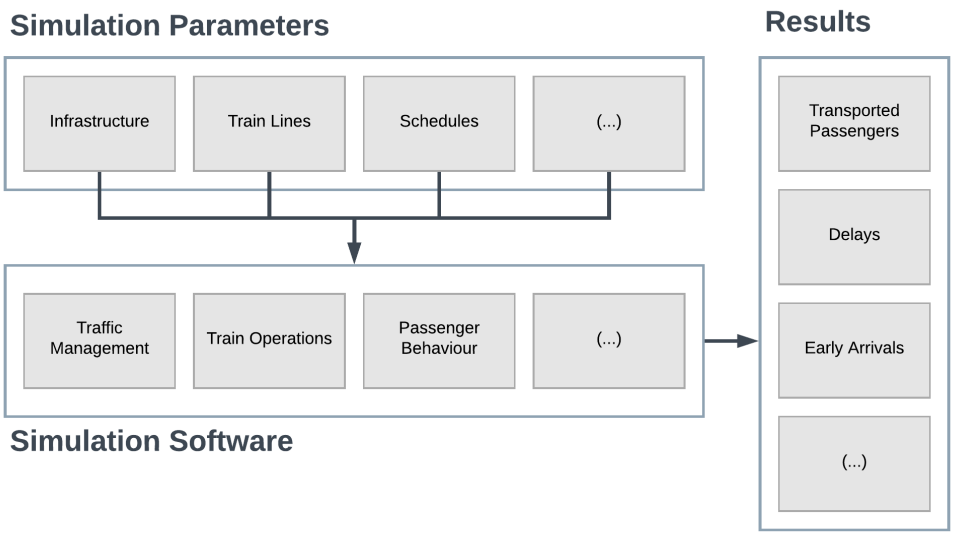


Figure 3.1.: Visualization of the basic components of the simulator with some example building blocks.

3.2. Description of the Architecture and its Building Blocks

This section describes the architecture (see 3.2) which is later used to create robust timetables. A brief overview of the four main building blocks is given in the following list, while a more detailed description of each block is given later in this section:

Model The model defines the basic components (e.g. Stations, Tracks, etc.) as well as the relationships between the components (e.g. Track tAB connects station A with station B), such that an arbitrary train network can be created. It also contains the definition of a train timetable and the constraints for the timetable.

Simulation The simulation handles all interactions between the components defined in the model (e.g. Train movement, passenger movement, etc.).

Results The results are used to analyse and rate the performance of a train network with a given timetable. They contain human readable information in form of plots/visualizations, as well as performance data used by the optimizer.

Optimizer The optimizer utilizes multiple simulation as its objective function to generate robust timetables

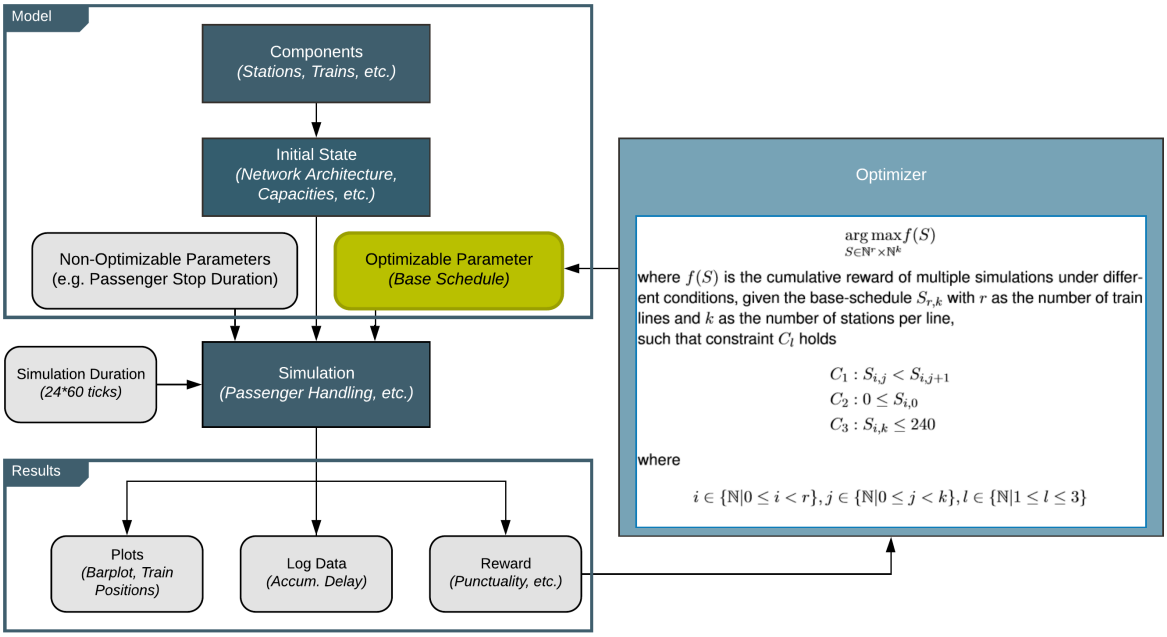


Figure 3.2.: Visualization of the designed Architecture.

3.2.1. Defining Model based on the Train Scheduling Problem

In chapter 2, the Train Scheduling Problem (TSP) was described as the process of finding an optimal timetable in a given network with given constrains. Because the exact definition of the TSP varies between different sources, it is defined in this project as follow:

Definition 3.2.1. Train Schedule Problem (TSP): Creating a schedule for a given infrastructure, rolling stock and a set of lines, such that a given criteria is optimized, whilst respecting capacity and security constraints.

Therefore the three main components infrastructure, rolling stock and lines have to be modelled. All those components can be very detailed in a real train network but modelling all the details would make the computation time of the simulation infeasible. As established in Section 2.1.3, a macroscopic view on the network is enough to design robust schedules an therefore the following simplification was chosen:

Tracks Tracks connect two stations an have a defined length, maximum speed and a number of parallel rails. If a track has only one rail, it can only be used in one direction at a time, but depending on its length, several trains can travel on the same track in the same direction. If the track has multiple parallel rails, the described restrictions apply to each rail individually (See Figure 3.3). The track gauge is not part of the used model and therefore any train can run on any track.

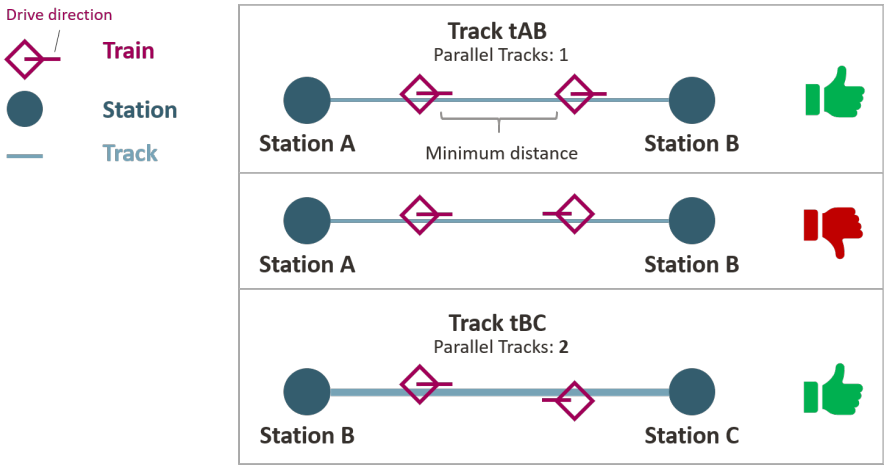


Figure 3.3.: Visualization of allowed track usage by several trains depending on the number of parallel tracks.

Stations Stations are places where trains can load/unload passengers, stop, dwell, overtake and change direction. They have a capacity which represents the number of platforms present at the station. Every station must be connected by a track to at least one other station. Additionally, the stations have a geographical position, which is only used to place the station in visualizations.

Trains/Rolling Stock Trains utilize tracks to transport passengers between stations. They are characterized by passenger capacity, maximum speed and acceleration.

Lines Lines define the routes that one or more trains run on the network (e.g. IR75, S6, etc.). There can be multiple lines and each line can have one or multiple trains traveling that line. A line is defined by a list of alternating track and station segments.

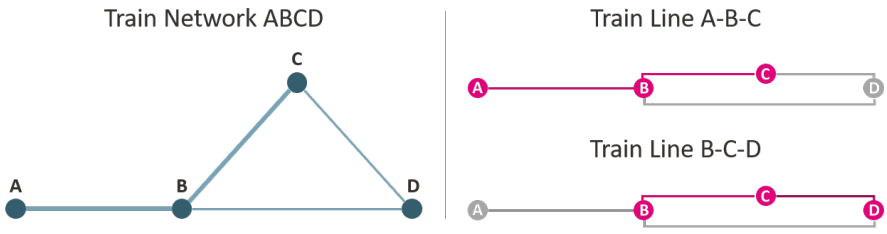


Figure 3.4.: Visualization of a train network (l.) with two train lines (r.).

Initial State (State 0)

The initial state defines a network design by relating the previously described components to each other, as well as defining the capacities of the components. An Example for an initial state can be seen in Fig 3.5

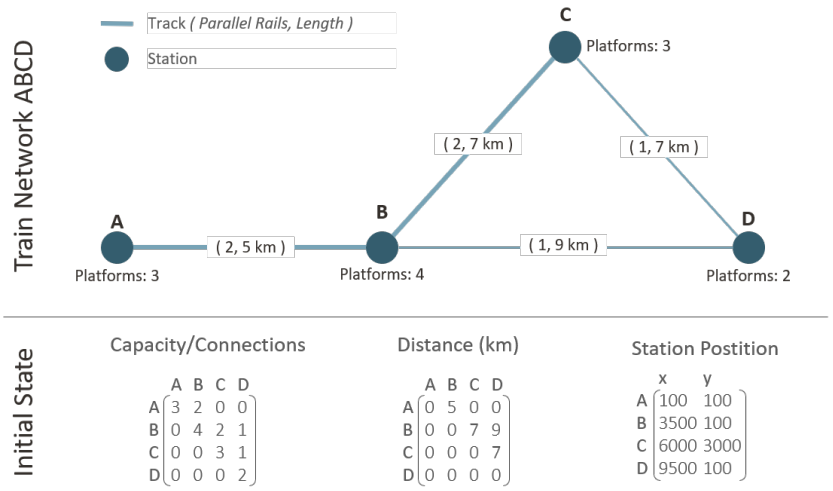


Figure 3.5.: Visualization of a train network (t.) with the corresponding initial state as a set of matrices(b.)

Non-Optimizable Parameters

Non-Optimizable Parameters (NOP) are properties that affect the entire train network and are, similar to the initial state, predefined. In contrast to the initial state, NOPs have no influence on the architecture, but they provide important information that is later used in the simulation. For example, the following properties are part of the NOPs:

- Minimum time required for a train to stop at a station to allow passengers to change.
- Minimum time required for a train to change direction.
- Minimum distance between two trains on a track running in the same direction.
- Periodicity of the train lines.
- etc..

Base Schedule Concept

The schedule is the central element of the model, as it is the parameter that is optimized. In a real network, timetables can be fairly detailed by defining not only arrival and departure

times at stations, but also at certain other points in the network (e.g. signals, junctions) (Lindner, 2000, p. 6). Additionally, a train schedule usually covers a whole day and there can be multiple different schedules for trains operating on the same line. For example, the local train network in the Canton of Zurich operates with a periodicity of half an hour and therefore it needs at least two schedules with a 30 minute offset from each other. The level of detail of real-world schedules is a problem when trying to optimize them, because the more details the schedule has, the tighter the constraints under which the optimizer must operate.

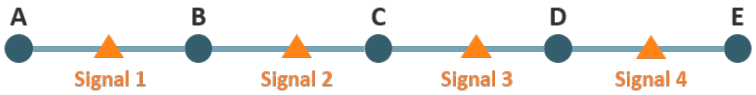


Figure 3.6.: Visualization of a example train network with five stations and four signals.

Example 3.2.1. *This example is based on the network presented in figure 3.6. Let our schedules only contain information about arrival times (in minutes) and let the arrival times only take values between 0 and 1440 (total number of minutes per day). Lets now have a schedule* S_1 *which holds arrival times for both stations and signals and has therefore a total of 9 entries. Without constraints, this results in* 1440^9 *possible schedules. Lets compare this to a schedule* S_2*, which for the same network only holds arrival times at stations and therefore reduces the number of entries to 5. The size of the search space without constraints is now reduced to only* 1440^5*.*

One way to tackle the problem with schedule size was introduced with the Periodic Event Scheduling Problem (see Section 2.2). When the trains in the network operate with a fixed period (e.g. 1h, 30min, etc.) it is sufficient enough to create a schedule for only one period and then extrapolate this base schedule. The generation of a set of schedules from a base schedule is shown in Fig 3.7, while Fig 3.8 visualises the extrapolation of schedules in a periodic network.

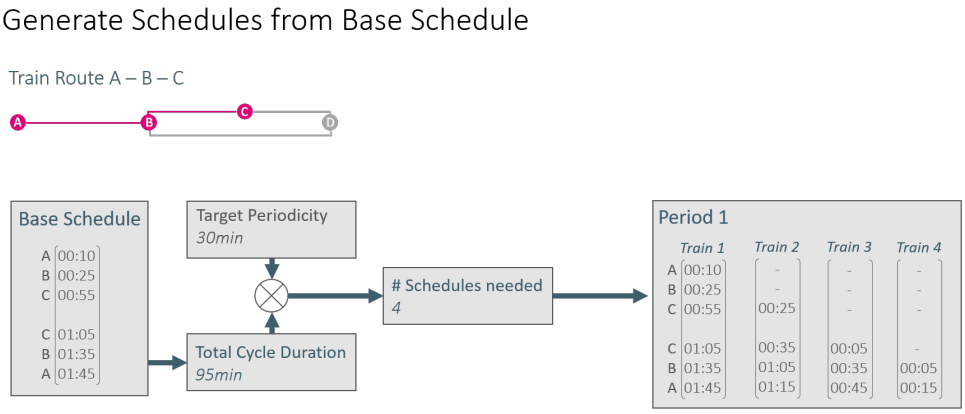


Figure 3.7.: Generation of four train schedules for train line A-B-C from a base schedule. The target periodicity is given by the user while the total cycle duration represents the time a train would need to complete the base schedule (One cycle in this example beginns at 00:10 and ends at 01:45, which results in a cycle duration of 95 min)

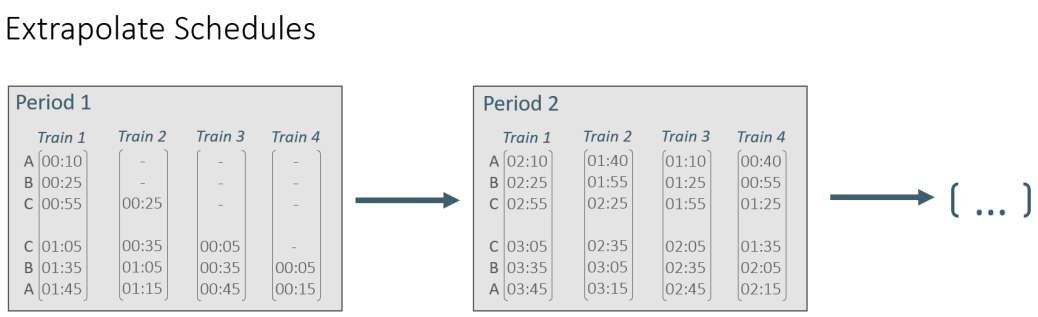


Figure 3.8.: Extrapolation of schedules in a periodic train network

The second measure to decrease the level of details in the schedule is, to only include the planed arrival time at each station. The departure time can then be calculated using non-optimizable parameters such as the minimum stop time. This still allows to measure the punctuality of the system and at the same time reduces the size of the schedule by 50%. The downside of this approach is, given that two trains arrive at a station at the same time, that the order in which the trains leave the station is dynamically determined by simulator (equivalent to traffic manager in real train networks) rather than the schedule. This problem can be mitigated by implementing the traffic management in the simulator as a deterministic function and thus enable the optimizer to adapt.

3.2.2. Description of Simulator Functionality

The simulator implements the previously described components and simulates the interactions between them. It does so by simulation one day (1440 minutes) in discreet one minute time steps. As many railway companies use timetables which repeat daily (e.g. SBB, ZVV), it was decided to only simulate single days (Heidron Buttler, 2017). The objective is then to measure the performance of a given schedule in this simplified, simulated train network. To achieve this, the simulator uses the following abstractions of a real-world railway system behaviour:

Passenger Movement Management Train Movement Management describes the task of generating and move passengers in the network. This includes:

- Generation of passengers at stations depending on the time of day and the demand at the station.
- Loading and unloading of passengers on the trains and stations, while taking the capacity of the trains into account.
- Removing passengers from the system when they arrive at their destination.

Traffic Management Traffic Management describes the task of coordinating trains and managing the resources of the network. This includes task such as:

- Managing the order trains access resources on the network.
- Rescheduling of delayed trains.

Train Movement Management The trains in the network have to be operated according to their individually assigned line and schedule. Train movement management includes the following task:

- Request access to network resources such as stations and tracks.
- Following the assigned route through the network.
- Stop at stations according to the assigned line.
- Control the train's speed according to the speed limits of the used track.

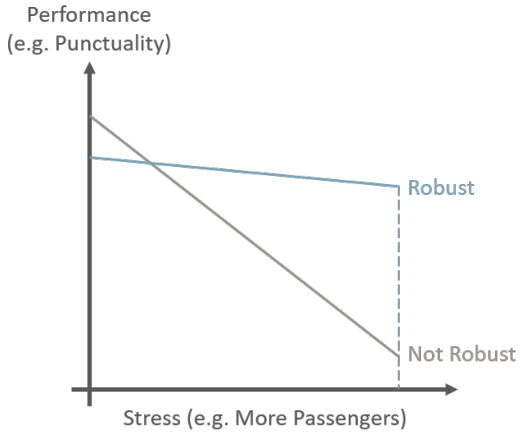


Figure 3.9.: Visualization of the concept to separate robust and non-robust schedules

3.2.3. Results for visualizing and comparing Schedules

The results produced by the simulator are divided into three different categories, as shown in Table 3.1:

Plots	Human readable diagrams of performance measurements or visualizations of the simulation.
Log Data	Data about the state of the simulation to analyse or replay a simulation run.
Reward	Performance metric of the simulation with a given schedule.

Table 3.1.: Categories of results with their corresponding description

Although all three categories are important to analyze a given system, the reward category is the most important for the optimization. The reward may include several different performance indicators, depending on whether the timetables should be optimised in terms of passenger satisfaction, robustness, efficiency, etc. (see Section 2.1.3). As the goal of this project is to optimize schedules for robustness, the most important performance indicators are delay and efficiency. But because robustness can be defined as the ability of a system to remain operational under increasing amounts of stress, the performance indicators delay and efficiency are not sufficient enough to measure it. To do so, the change in delay and efficiency has to be measured while increasing the stress on the system. This is achieved in this project by doing multiple simulations with the same schedule whilst increasing the load on the network (e.g. More Passengers, Trans, Delays, etc.). The robustness of the schedule can then be measured by how quickly performance drops under stress (see Fig 3.9).

3.2.4. Definition of the Optimization Problem

The optimization problem for this project is defined as follows:

$$\arg\max_{S \in \mathbb{N}^r \times \mathbb{N}^k} f(S)$$

where $f(S)$ is the cumulative reward of multiple simulations under different conditions, given the base-schedule $S_{r,k}$ with r as the number of train lines and k as the number of stations per line,
such that constraint C_l holds

$$C_1 : S_{i,j} < S_{i,j+1}$$
$$C_2 : 0 \leq S_{i,0}$$
$$C_3 : S_{i,k} \leq 240$$

where

$$i \in \{\mathbb{N} | 0 \leq i < r\}, j \in \{\mathbb{N} | 0 \leq j < k\}, l \in \{\mathbb{N} | 1 \leq l \leq 3\}$$

The constraints can be alternatively described as follows:

C_1 A base-schedule always has to go forward in time.

C_2 The arrival times in the base-schedule have to be positive.

C_3 The period of a schedule cannot be longer than 240 minutes.

The last constraint serves to further reduce the search space 3.2.1) and can only be applied because the networks studied in the experiments are small enough that trains can cover the assigned distance in less than four hours (2 hours for each direction). For larger networks this constraint would have to be relaxed.

Figure 3.10 illustrates the constraints by providing examples for valid and invalid base-schedules.

3.3. Comparison of Optimization Methods

To find a suitable optimization algorithm for optimizing schedules, several methods were examined and compared. The results are summarized in Table 3.2.
For the simulation based optimization which is used in this project, reinforcement learning and genetic algorithms are best suited because they optimize numerically and can handle large search spaces. When comparing those two methods, it can be seen that they have very similar advantages and disadvantages. The main difference is, that reinforcement learning has a perception of the network, while genetic algorithm have a better adaptability to different network architectures and sizes. Therefore if a reinforcement learning agent is trained on many different network designs, it should learn good scheduling patters for certain network features (e.g. using central stations as hubs for passengers to change trains)

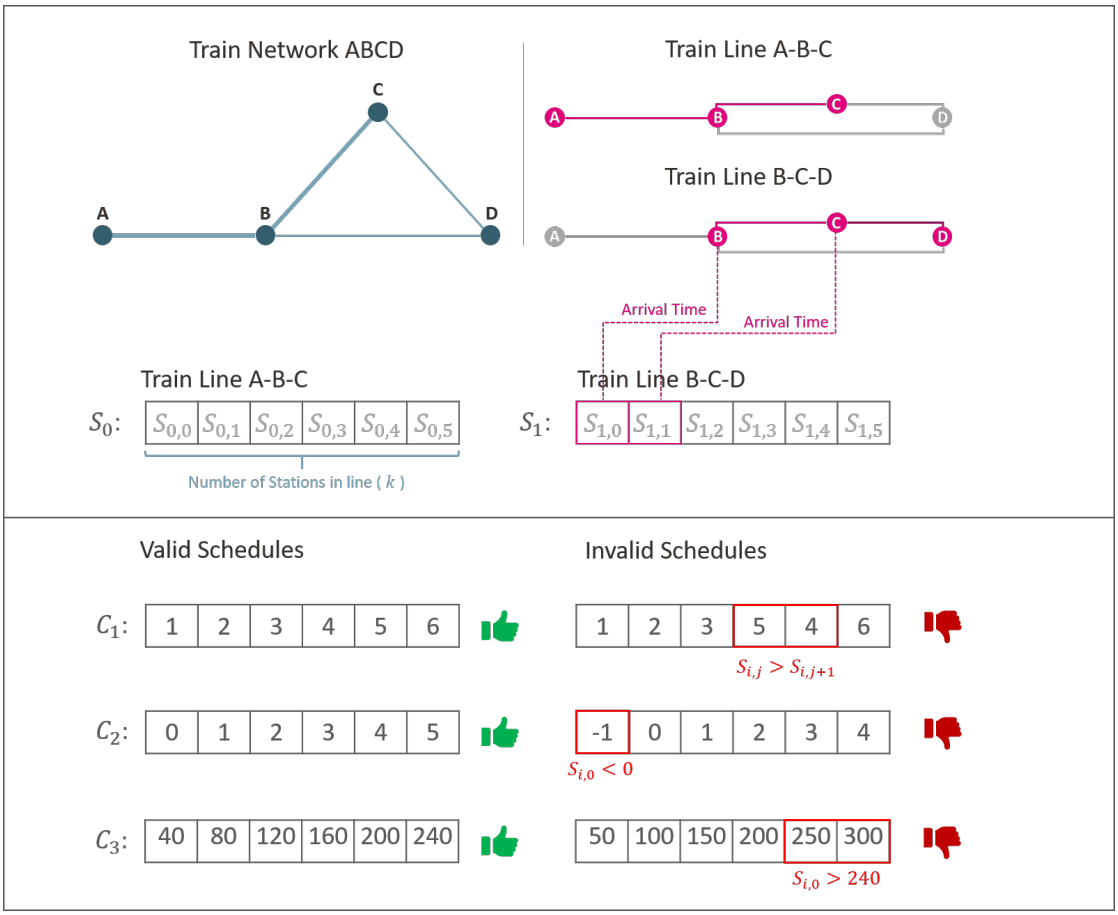


Figure 3.10.: Illustration of valid and invalid constraints for base-schedules in an demonstrative network

	Integer Programming	Constraint Programming	(Deep) Reinforcement Learning	Genetic Algorithm
Advantages	• Fast for small problems • Optimality proof	• Fast for small problems • Optimality proof • Usage of specialized constraints not available with integer programming	• Learns a policy instead of just finding an optimal solution • Faster for large search spaces than conventional algorithms • Showed good performance in similar real world problem • Can be used with a simulation based model	• Can deal with different network architectures and network sizes without changing the optimizer • Faster for large search spaces than conventional algorithms • Can be used with a simulation based model
Disadvantages	• Requires algebraic model • Performance when dealing with a large search space	• Requires an algebraic/logical model • Performance when dealing with a large search space	• Only approximates an optimal solution • Changing the train network architecture can requires change of the optimizer	• Only approximates an optimal solution • Has no perception of the network or state

Table 3.2.: Comparison of different optimization methods

and then be able to utilize those patterns in new networks. But to learn those patterns, the agent needs to have a perception of the train network architecture which most likely will be represented as a connectivity matrix (Or a set of matrices). Now if the representation of the network is changed or if larger networks (larger connectivity matrices) are used, the architecture of the agent would most likely have to adapt.

As one of the goals of this project is to run multiple experiments with different network sizes and architectures, the adaptability of the generic algorithms is preferred over the perception of the reinforcement learning agent. Another reason why a genetic algorithm was chosen as optimizer is, that its simplicity allows to create an own implementation while an approach with reinforcement learning would have to rely in the context of a bachelor project on existing frameworks and agent architectures.

4. Implementation of the Simulator and the Genetic Optimizer

4.1. Implementation of the Simulator and its Components

The simulator was implemented on the basis of the architecture and model described in Section 3 by using Python 3.7 and the framework SimPy[1]. It is designed to simulate the operations in a train network during one day in discrete one minute time steps. The following chapters give an insight into the functionality of the simulator by showing the used methods with examples and visualizations. The actual code can be accessed via the Git repository of the project[2].

4.1.1. Track Component

The track component is designed as a resources which connect two stations together and enables trains to move between those stations. It is defined by the following properties:

- **Id:** A unique id to identify the track in the network. The structure of the id for tracks is "t+<from station><to station>" where the stations are ordered alphabetically (e.g. tAB, tCF, etc.) which makes the naming more consistent (Otherwise a track between station A and station B could be either named tAB or tBA).
- **Length:** The total length of the track in meters.
- **Maximum speed:** The maximum allowed drive speed in km/h for trains on the track segment.
- **Adjacent nodes:** The id and reference to the two connected stations.
- **Number of parallel rails:** Number of rails between the two stations which can be independently accessed.
- **Maximum capacity per rail:** The maximum number of trains each rail can hold such that a minimum distance between all trains can be guaranteed.

The most important function of the track component is to control the access of trains in such a way that there are no train crossings or overtaking on a single rail track (see Figure 4.1). To achieve this, the track components include a simple speed and access management.

[1] https://simpy.readthedocs.io
[2] https://github.com/iareust/train_schedule_optimization

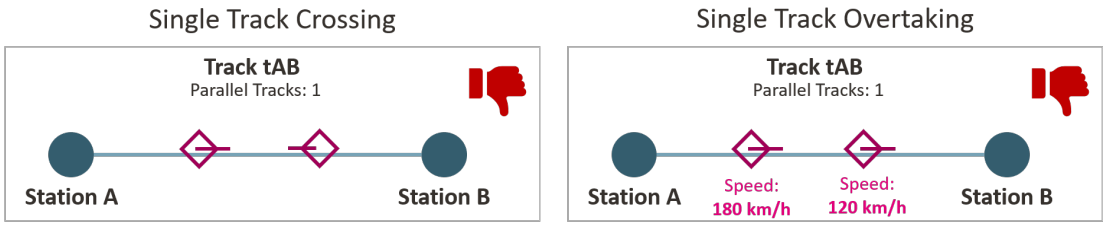


Figure 4.1.: Visualization of situations which would result in train crashes an therefore have to be prevent by the access management of the train component

Speed Management

The speed management guarantees that trains with different maximum speeds cannot overtake each other and that no train can drive faster than the maximum allowed speed of the track. Each track keeps a speed register which contains information about all trains on the track and the speed at which each train runs. This allows the track to limit the speed of fast trains to the speed of the slowest train currently on the track. The following example illustrates the functionality of the speed register:

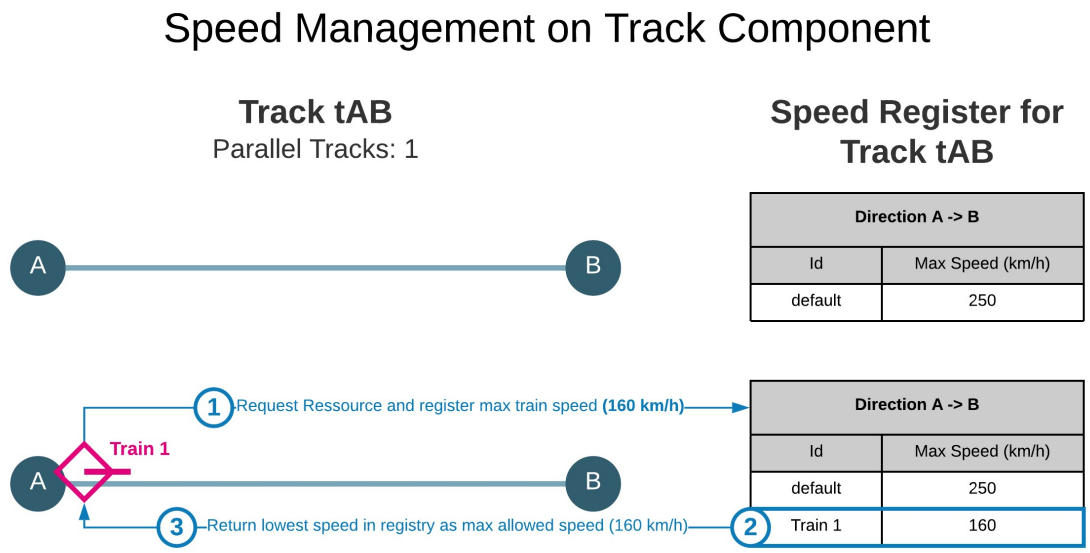


Figure 4.2.: Example of speed management with a single train on track

1. Train 1 request access to the track and registers its maximum speed of 160 km/h.
2. A new register entry is created for Train 1. As the speed of Train 1 is lower than the default allowed speed of 250 km/h, its speed is entered in the register.

3. The speed management returns 160 km/h as the speed which Train 1 is allowed to drive on the track.

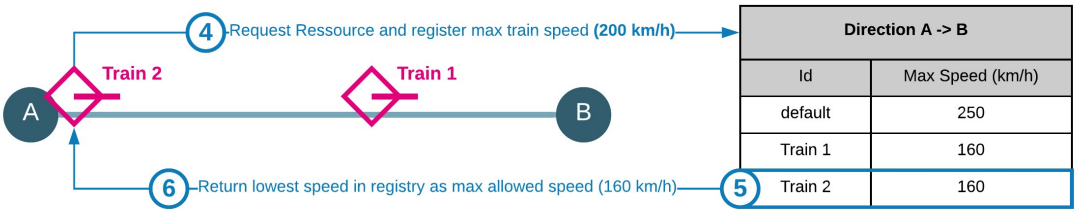


Figure 4.3.: Example of speed management with a second train accessing the track

4. A second Train requests access to the track and registers with its maximum speed of 200 km/h.
5. The speed management compares the maximum speed of Train 2 (200 km/h) with the lowest speed currently registers (160 km/h). To avoid overtaking it reduces the speed of Train 2 to the lowest value and saves that speed.

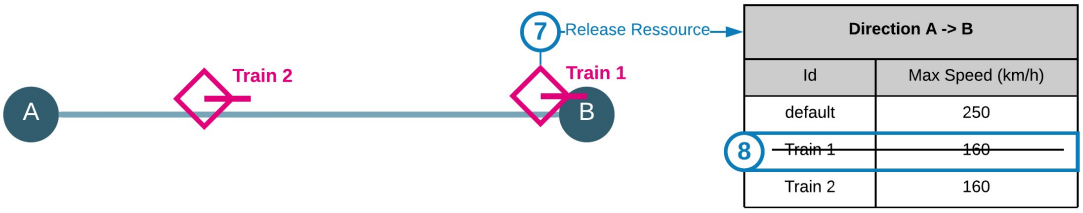


Figure 4.4.: Example of speed management with a train leafing the track

6. The drive speed of 160 km/h is returned to Train 2.
7. Train 1 leaves the track and the corresponding entry in the register is removed.

Its worth to mention, that the described speed management works because trains in the simulation always maintain their maximum allowed speed for the whole transfer between two stations. A more sophisticated speed management could be implemented by updating the speed information at every tick of the simulation and informing all trains about changes in the speed register. Since this would mean many additional operations per tick and thus slow down the simulation, the simplified management was used instead.

Access Management

The access management ensures that the capacity of a track is not exceeded and therefore a minimal distance between trains can be kept. It additionally prevents trains from crossing

each other on the same rail, which would otherwise result in train collisions. The access management is implemented by wrapping one SimPy resource object for each track direction. The SimPy resources provide functionality for blocking requests (if the capacity is fully utilized) and releases, while the wrapper synchronizes the resources in both direction. The following illustration shows the process in a simple example:

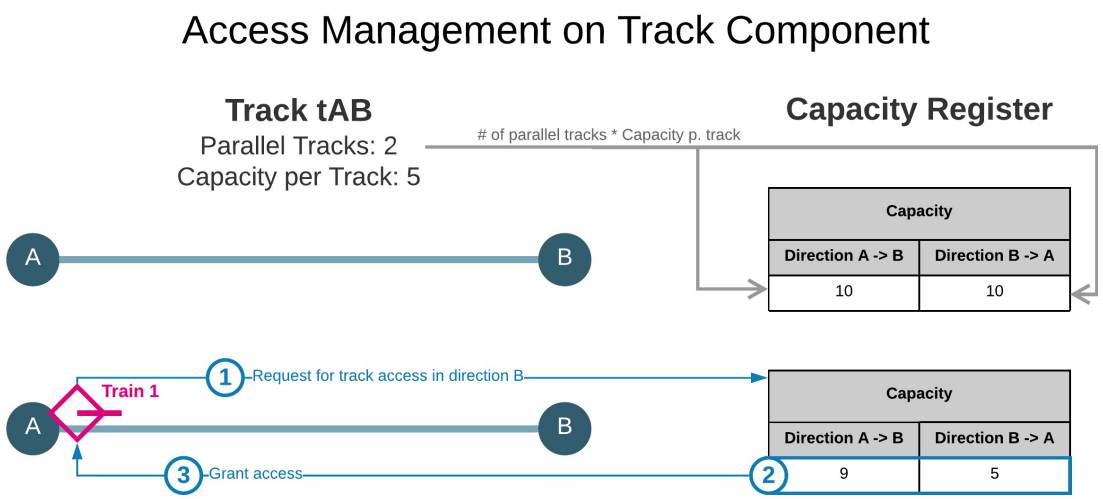


Figure 4.5.: Access management with a single train accessing the track

1. Train 1 requests access to the track for traveling to station B.
2. The access manager checks whether the current capacity of track direction A->B (10) is a multiple of the 'Capacity per track' property (5). Because this is the case, it subtracts 5 from the capacity of the opposite direction because one out of the two rails is not longer available for traffic going from B to A. Additionally the capacity of A->B is reduced by 1.
3. Train 1 is granted access to the track.

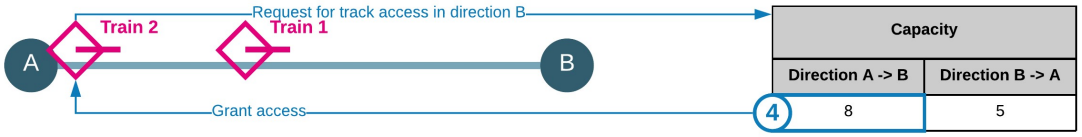


Figure 4.6.: Access management with two trains going into the same direction

4. Train 2 had requests access to travel to station B. Because the current capacity A->B (9) is not a multiple of 5 it is only decreased by 1 while the capacity B->A remains at 5.

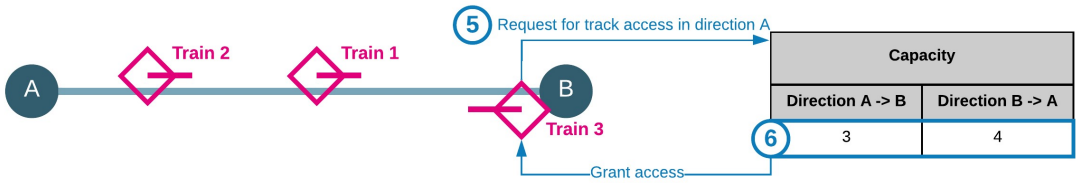


Figure 4.7.: Access management with a three train using the track where the train go into opposite directions

5. Train 3 requests access to travel into direction of station A.
6. Analogous to step 2, the capacity of B->A is a multiple of the 'Capacity per track' property and therefore 5 is subtracted from the capacity of opposite direction. The capacity of B->A is then decreased by 1 and Train 3 is granted access to the track (second rail).

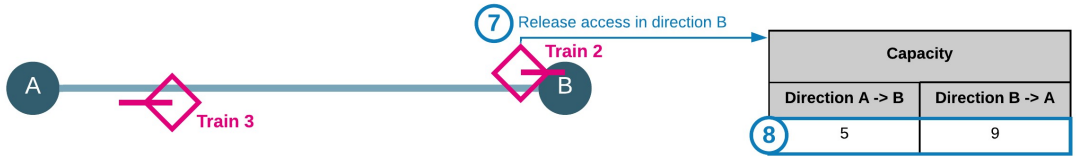


Figure 4.8.: Access management where one train leaves the track

7. Train 1 and Train 2 have entered station B and release their access on the track.
8. The capacity of direction A->B is increased by 2. Because the new capacity is a multiple of the 'Capacity per track' property, and therefore one rail is freed up again, the capacity of the opposite direction (B->A) is also increased by 5.

4.1.2. Station Component

The station components perform two important functions in the simulated network; They are generators and sinks for the passengers as well as points at which trains can perform manoeuvers such as stopping/dwelling, changing passengers and turnarounds. The following properties are required for the station to work properly:

- **Id:** A unique id to identify the station in the network. The structure of the id for stations is "s+<station name>" (e.g. sA, sB, etc.)
- **Number of platforms:** The number of platforms corresponds to the number of trains that the station can accommodate at any one time. The platforms are, similarly to tracks, implemented with SimPy resource objects.
- **Passenger count:** Amount of passengers waiting at the station.

- **Passenger destination factor:** A value between 0% and 100% that determines how many people leaving a train at the station leave the network and how many remain for connecting trains. For example, a hub station such as Zurich HB will have a lower destination factor (e.g. 70%) compared to a rural terminal station (e.g. 1.0/100%).
- **Maximum passengers per hour:** Defines the maximum number of passengers generated at the during rush hours.
- **Passengers handles:** A counter that records the total number of passengers arriving at their destination.
- **Position:** A tuple of x/y coordinates that define the position of the station on a Cartesian plane. The position property is only used to place the station in visualizations, but has no effect on the simulation itself. If the positioning of the stations in the network would cause tracks to cross, those crossings would have to be modeled by the network architect using the existing components (e.g. a station with no passenger demand and a single platform can be used at the cross-point of two tracks to represent an intersection). The simulator makes a clear distinction between the network visualisation and the network architecture.

Passenger Management

The passenger management controls the flow of passengers at stations. This includes generating passengers dependent on the current time, as well as the handling of passengers who disembark or board the train. The amount of passengers generated at any station is defined by the 'Maximum passengers per hour' property and a demand function. The former determines the total maximum passenger volume (e.g. Zurich HB: max 20'000 P. per h, Rotkreuz: max 1'200 P. per h) while the latter defines the passenger demand for every minute of the simulated day. The demand function can be any function to which the following applies:

Let the demand function f be a function such that:

$$f : \{t \in \mathbb{N} | 0 \leq t \leq 1440\} \rightarrow \{y \in \mathbb{R} | 0 \leq y \leq 1\} \tag{4.1}$$

where t is the current simulation time and y is the demand factor

Figure 4.9 shows the demand function which was used for this project. It has its rush hours at 08:00 and 17:30. The placement of the peak hours is loosely based on the analysis of public transport carriers conducted by Anderhub et al., 2008, pp. 60, 61.
The demand function is then used together with the maximum passenger volume to generate passengers at each minute of the simulation. The process of generating passengers at a specific time is shown in Fig 4.10.

1. The current time in the simulation is entered into the demand function, which returns a passenger demand factor of 0.8.

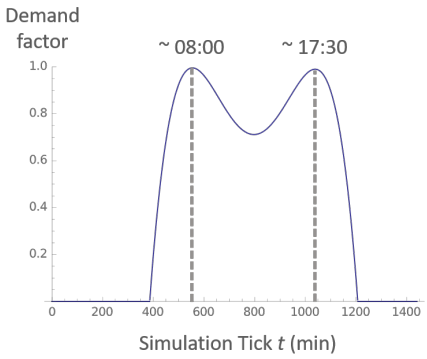


Figure 4.9.: Example for a demand function with peek demand at 08:00 and 17:30

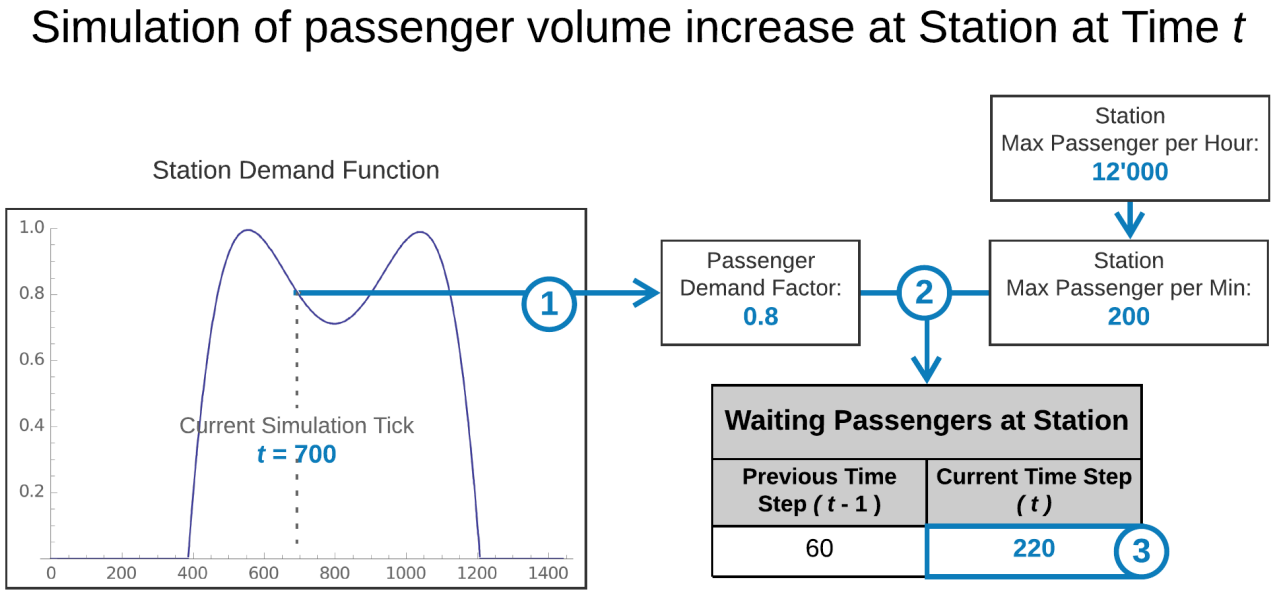


Figure 4.10.: Visualization of passenger generation at a Station at time t

2. The number of new passengers is then calculated by $\text{passenger}_{\text{new}} = \frac{\text{passengerdemand}_{\text{hour}}}{60} \times \text{demand factor}$. This results in 160 new passengers for this example.

3. The 160 new passengers are added to count of waiting passengers.

Passenger management not only generates passengers, but also takes care of the transfer of passengers to and from the trains. When passengers disembark from a train, they can either be at their destination or waiting to get onto another train. The relationship between these two types of passengers is modelled with the passenger destination factor. The example in Figure 4.11 illustrates how this works:

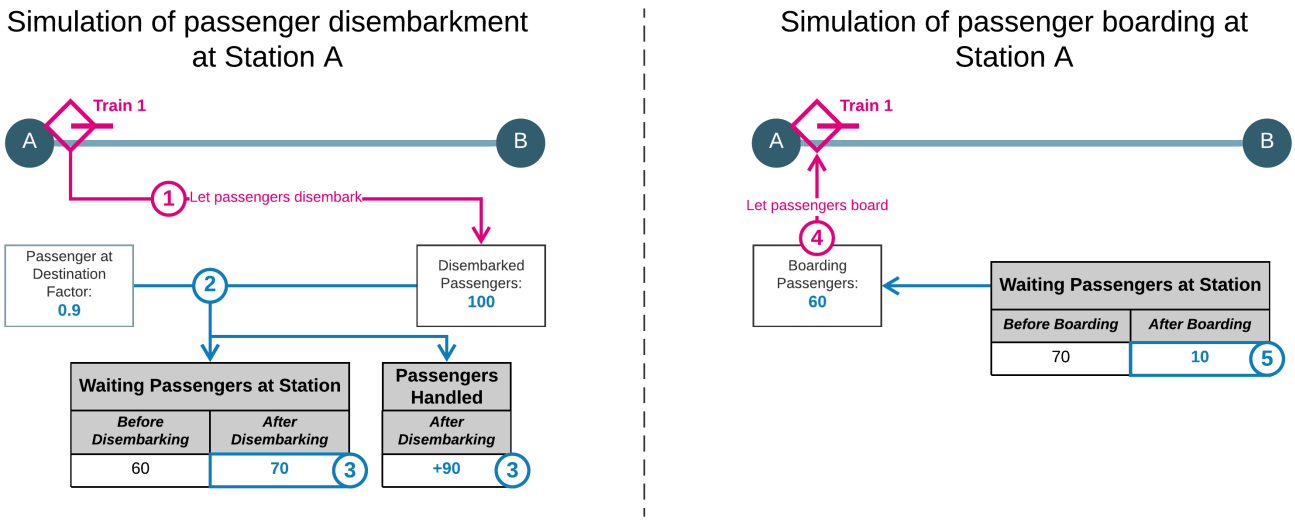


Figure 4.11.: Visualization of passenger transfer between station A and train 1 from the perspective of the station

1. 100 Passengers disembark from Train 1 at Station A.
2. The disembarked passengers are multiplied with the passenger destination factor which results in 90 passengers.
3. Those 90 passengers are at their destination and therefore the handled passenger count is increased by 90. The remaining 10 passengers are added to the waiting passenger count.
4. After the passengers have got off, train 1 is loading the passengers from the station (60 passengers). The amount of passengers loaded is controlled by the train, but the train never loads passengers who have previously got off the same train.
5. The waiting passenger count at station A is reduced by the amount of passengers which boarded Train 1

4.1.3. Train Line Collection

Train line collections are a combination of the schedule and the route a train has to follow (e.g. IR75, S3). They are generated from a line configuration, which is created by the user as part of the initial state (see Section 3.2.1), and the provided base schedule. Each line collection consists of a number of sections. The sections contain information about the type of section, the id of the infrastructure used in that section and, in the case of stations, also the planned arrival time according to the base schedule. The section types are implemented as an enumeration with the following values:

- **START:** A section which marks the start of the line collection.

- **PASSENGER_STOP:** Defines a section where the train stops and allows passengers to transfer between the train and a station. In contrast to the other sector types, PASSENGER_STOP sectors contain additional information about the planned arrival time, as well as an unloading factor. The latter is explained in more detail in section 4.1.4.
- **TRAVEL:** A section which is traversed by the train. This would usually be a track but it can also be a station at which the train only passes through.
- **TURNAROUND:** A section which indicated the train to change its driving direction.
- **END:** Indicates that the line is finished.

The following illustration portrays a train collection for a simple two station line and a given base schedule:

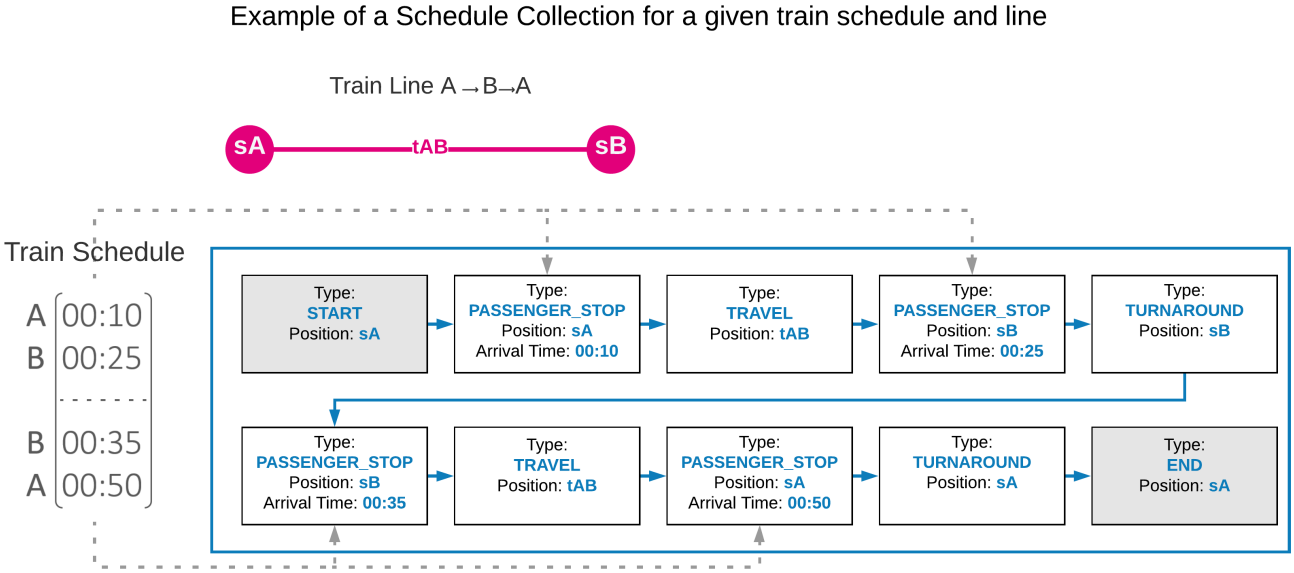


Figure 4.12.: Visualization of a train line collection in a simple two station line and a given base schedule

4.1.4. Train Component

Trains utilize the previously described components to transport passenger on their designated line. They are defined by the following properties:

- **Id:** A unique id to identify the train in the network. The structure of the train id is "t+<line id>+<unique int>".

- **Capacity:** The maximum number of passengers that the train can accommodate. The simulation does not differentiate between different passenger classes (e.g. first- and second class).
- **Maximum Speed:** The maximum speed of the train in m/s.
- **Acceleration:** The acceleration of the train in m/s^2.
- **Train line collection:** A train line collection object that defines the route and schedule that the train follows.
- **Position:** The x/y position of the train which is used to visualize the train on a Cartesian plane.

The behaviour of the train is controlled by a simple train logic, which is presented as a flowchart in Fig 4.13.

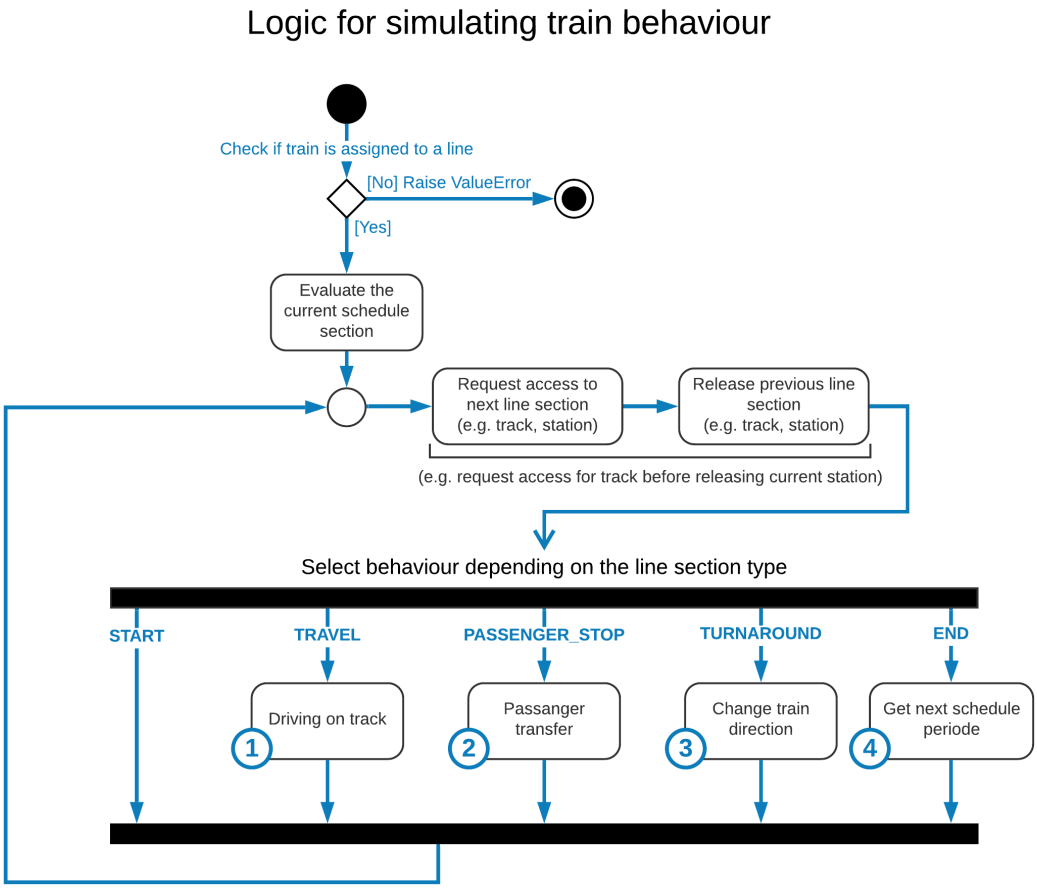


Figure 4.13.: Flowchart of the train logic used in the train network simulation

1. The train calculates the number of time steps needed to cover the track section. The calculation depends on the length of the track, acceleration/deceleration and maximum allowed speed.

2. The process for transferring passengers from the view of a station component was already described in section 4.1.2. The difference between a station and a train is, that a train has a limited passenger capacity. It therefor only loads passengers until the capacity limit is reached. The amount of passengers a train unloads is decided by the unload factor of the current section (see Fig 4.14). The unload factor determines how much percent of the trains current passengers leaves at the section (e.g. Zurich HB 80%, Rotkreuz 20%).

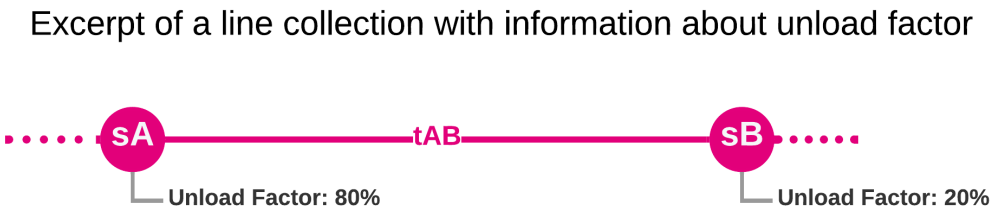


Figure 4.14.: Visualization of a part of an example line collection with information about the unload factor shown. Station A represents a popular or central station where lots of passengers leave, whereas station B represents a smaller, less popular station.

3. A change in train direction is simulated by forcing the train to stop a minimum amount of time (e.g. train driver changes driver's cabin). The time required for a turnaround is defined by the user in the simulation settings.

4. When a train has reached the end of its current schedule period, it will generate the following schedule period by shifting the arrival times in the schedule collection. This process takes the delay of the train into account. If the train's arrival delay at its final destination was longer than the time between two consecutive trains on the same line, one timetable is skipped.

4.1.5. Plot of simulated Network

The simulation offers the ability to show a live visualisation of the current network state. This can be used to analyse problems in the network, such as congestion or deadlocks. The downside of the visualisation is, that it does drastically increases the simulation time and therefore it is disabled during optimization. An example for the visualization can be seen in Fig 4.15.

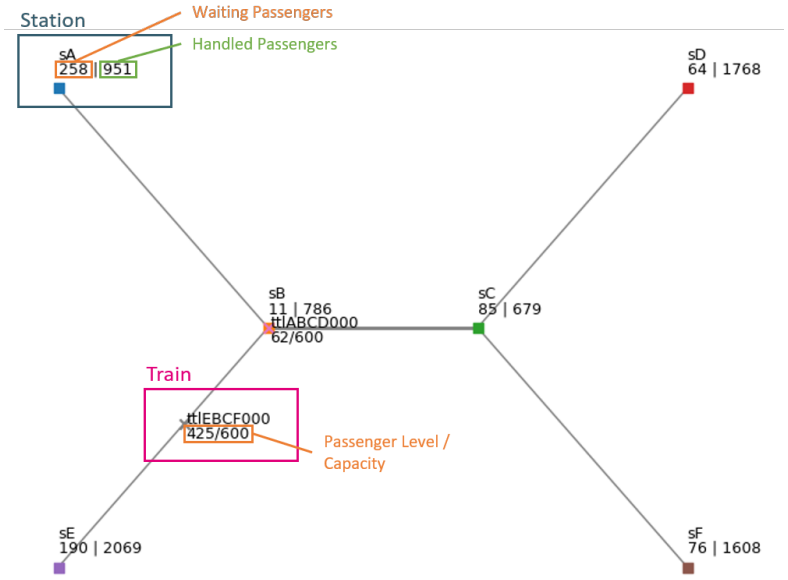


Figure 4.15.: Visualization of a simulation run with an example network consisting of six stations and two trains

4.1.6. Performance Indicators

To evaluate the operational performance of a network during simulation, the simulation environment tracks the following values:

- **Total train operation time:** The total amount of time (min) that trains operated in the network. This includes traveling, dwelling, passenger stops and turnarounds.
- **Total arrival delay:** The accumulative arrival delay of all trains during the simulation.
- **Total early arrivals:** Early arrivals are logged, when a train arrives at a station before its schedules arrival time.
- **Total passenger volume:** To total amount of passenger which were generated in the network during the whole day.
- **Total handled passengers:** The total amount of passengers which arrived at their destination.

Those key values are then used at the end of each simulation to calculate the performance of the network. The three performance indicators which are used in this project are described in the following subsections.

Passenger Performance

Passenger performance indicates the ability of a train network to transport all passengers to their destination. We define it in this project as:

$$f_{passenger}(x) = 1 - \sqrt{1 - \left(\frac{x}{a}\right)^2} \tag{4.2}$$

where x is the number of transported passengers and a is the total number of passengers generated, and:

$$x \in \{\mathbb{N} | 0 \leq x \leq a\}, a \in \{\mathbb{N} | 0 < a\} \tag{4.3}$$

The function is designed so that the drop in performance for the first few not transported passengers is greater than the drop in performance when most passengers are unhandled. For example, in a good working system the passengers find it very annoying if they do not reach their destination in time because of full trains. On the other hand, in a very bad working system it does not make much difference when instead of 20% suddenly 30% of the passengers are transported. In an earlier version of the passenger performance function, the performance was linearly dependent on the handled passengers. However, tests performed with the linear function showed that it did not guide the optimizer toward good schedules as effectively as this function does. This behaviour of the function can be seen in Fig 4.16.

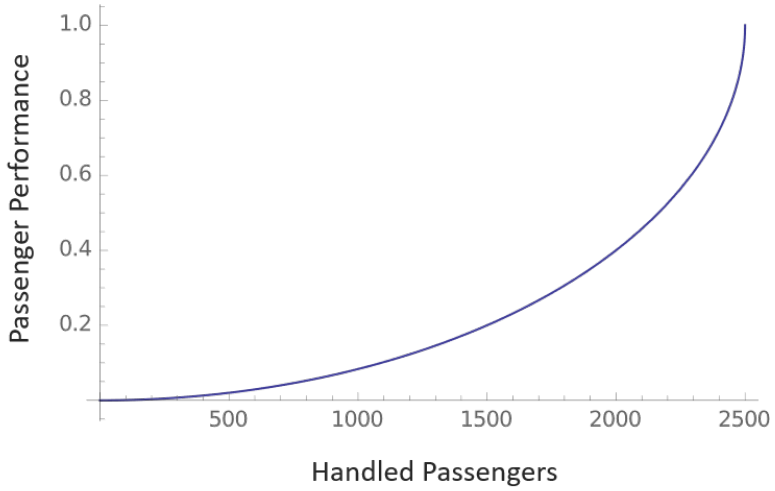


Figure 4.16.: Plot of the passenger performance function with 2500 as the total amount of passengers

Delay Performance

Delay performance indicates the ability of a train network to provide punctual connections for the customer. We define it in this project as:

$$f_{delay}(x) = \begin{cases} 1 & \text{for } x \leq b \times c \\ \frac{(a-x)^2}{(b \times c - a)^2} & \text{for } x > b \times c \end{cases} \tag{4.4}$$

where x is the total arrival delay, a is the total operation time of all trains, b is the tolerated amount of delay per train line per day and c is the number of train lines. Additionally, the following holds:

$$x \in \{\mathbb{N} | 0 \leq x \leq a\} \tag{4.5}$$

$$a \in \{\mathbb{N} | 0 < a\}, b \in \mathbb{N}, c \in \{\mathbb{N} | 0 < c\} \tag{4.6}$$

$$b \times c - a \neq 0 \tag{4.7}$$

The delay performance function incorporates a delay tolerance for each train line per day. With this tolerance, it is still possible for a network to reach the maximum performance level, even if a train was slightly delayed, as shown in Fig 4.17. To tolerate small deviation from the schedule is also common in real-world railway systems. For example, SBB tolerates deviations of up to 3 minutes according to their website (SBB, 2020).

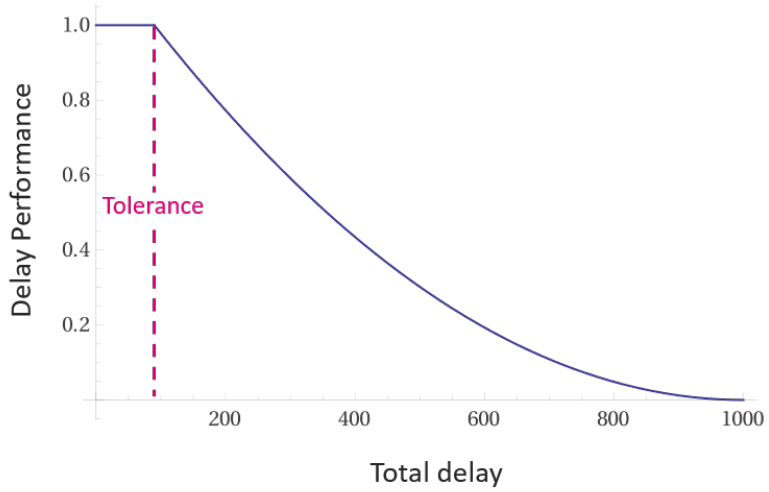


Figure 4.17.: Plot of the delay performance function with 1000 as the total operation time, 30 minutes delay tolerance per line and 3 train lines

Early Arrival Performance

Early arrival performance specifies how well a network utilized its trains. We define it in this project as:

$$f_{early}(x) = \begin{cases} \sqrt{1 - 4\left(\frac{x}{a}\right)^2} & \text{for } x \leq a \\ 0 & \text{for } x > \frac{a}{2} \end{cases} \tag{4.8}$$

where x is the total early arrival time and a is the total operation time of all trains such that the following holds:

$$x \in \{\mathbb{N} | 0 \leq x \leq a\} \tag{4.9}$$

$$a \in \{\mathbb{N} | 0 < a\} \tag{4.10}$$

Early arrivals are not as problematic as delays because passengers generally are more sensitive towards arriving late than arriving early. Nevertheless if the trains in the network arrive early all the time, it indicated that the available trains are not used effective. For this project we defined that systems which utilize their trains less than 50% of the time have zero early arrival performance while the convex nature of the function still allows the optimizer to schedule dwelling times in an already punctual system without much performance drop.

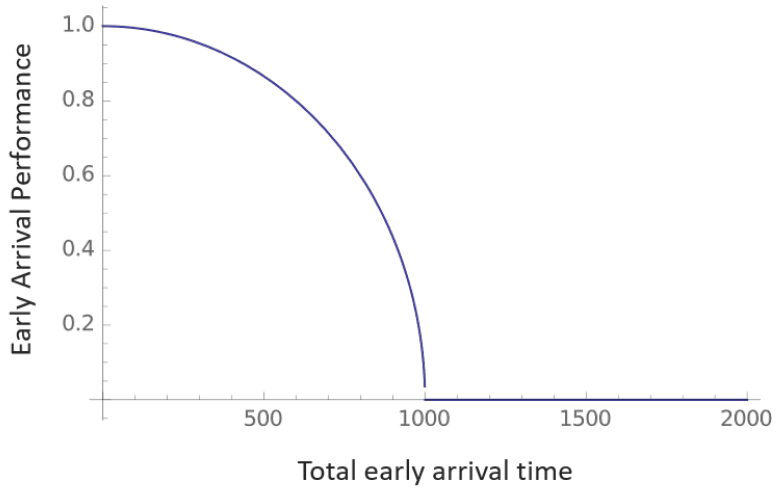


Figure 4.18.: Plot of the early arrival performance function with 2000 as the total train operation time

4.2. Genetic Optimizer

The genetic optimizer uses the simulator described above to optimize schedules for efficiency or robustness. It was implemented with Python 3.7 specifically for the task at hand. The main algorithm behind the optimizer is detailed in Algorithm 1. Each functionality will be described in the following subsections.

Algorithm 1: Genetic schedule optimization

```
Generate initial population P with random schedules;
foreach p ∈ P do
    Evaluate reward for p using the simulator;
end
Sort P descending by the evaluated reward;

for n generations do
    Remove the lower 90% of population P;
    Repopulate P with the remaining members using genetic operations;
    foreach p ∈ P do
        Evaluate reward for p using the simulator;
    end
    Sort P descending by the evaluated reward;
end
```

4.2.1. Population and Population Members

A Population consists of multiple population members. Each population member contains one or more base schedules, depending on the number of lines in the network on which the optimization is carried out. For example, in a network with two different train lines, each population member contains two schedules. The schedules themselves represent a chromosome while the time information in the schedules represent single gens (see Sec 2.4.2). The usage of multiple chromosomes per member differs from the common genetic algorithm approach to only use one chromosome per member (e.g. Arenas et al., 2015). Using multiple chromosomes per member makes it easier to optimize networks with train lines of different lengths (e.g. line 1 with 4 stations, line 2 with 8 stations, etc.). However, this also presupposes that genetic operations can only be conducted between homologous chromosomes (e.g. chromosome of line 1 cannot be crossed with chromosome of line 2). Figure 4.19 visualizes the different hierarchical levels used by the optimizer.

4.2.2. Generation of initial Population

The basic concept for generating the initial population is to create members with random schedules, ensuring that they do not violate the constrains described in Equation 3.2.4. The code with which the initial population is generated is shown in Figure 4.21. The following

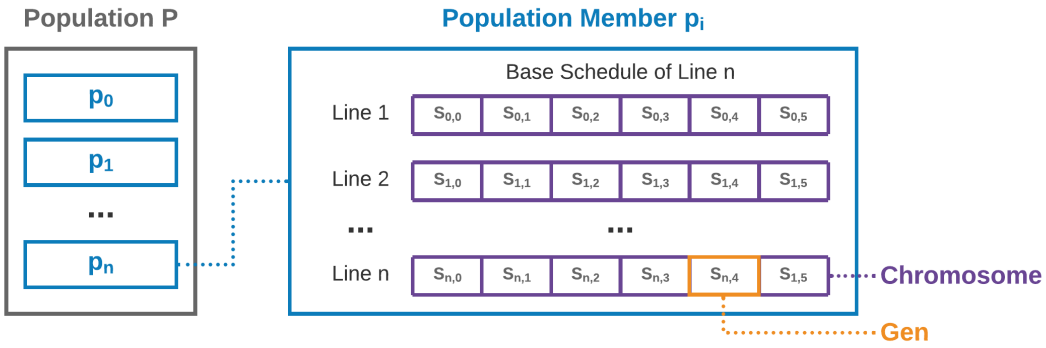


Figure 4.19.: Visualization of different hierarchical levels used in the optimizer. All chromosomes in this graphic have an equal length of six to simplify the visualization. The actual chromosomes used in real optimization can vary in length between the different lines.

comments explain the most important sections of the code:

Line 10: The function uses a ProcessPoolExecutor and futures to utilize all available processor cores to run multiple evaluations and simulations simultaniously (see Section 4.2.3). This reduces processing time when the underlying system has multiple cores or processors.

Line 18: The schedules are generated with a pseudo-random number generator provided by NumPy. The generated list of integer numbers is distributed uniformly between 0 and 240. This ensures that constraint C_3 of Equation 3.2.4 is not violated. By sorting the list constraint C_2 can be guaranteed. Fig 4.20 shows an example for this process.

Figure 4.20.: Generation of a list with random numbers followed by sorting to create a random base schedule

Line 21,26: The schedules are evaluated by simulating the provided network with the generated schedules and calculating the rewards based on the results from the simulation. This process is further described in Section 4.2.3.

```
from simulation.simulator import Simulator
from concurrent.futures import ProcessPoolExecutor
import numpy as np

def generate_initial_population(network, population_size) -> []:
    sim = Simulator(network)
    future_collection = []
    population = []

    with ProcessPoolExecutor() as e:

        for _ in range(population_size):

            # Generate the chromosomes/timetables for the member
            member = []
            for line in NETWORK.LINES:
                line_length = get_line_length(line)
                member.append(sorted(list(np.random.randint(low=0, high=240, size=line_length))))

            # Run timetable evaluation in separate process
            future_collection.append((member, e.submit(sim.evaluate_timetable, member)))

        # Fetch results from futures
        for member, future in future_collection:
            results = future.result()
            total_reward = calc_total_reward_from_results(results, member)
            population.append((member, total_reward, results))

    return population


```

Figure 4.21.: Python function to generate a random initial population for a given network and target population size

4.2.3. Evaluation of Reward using the Simulator

In Section 4.1.6 the performance indicators of the simulator were introduced. Although those indicators can be used to evaluate the efficiency of a timetable they are not sufficient to determine weather a timetable is robust or not. This is due to the fact that the performance indicators only represent the performance of the network under one set of conditions and not under changing conditions. As noted in Chapter 3, this problem can be circumvented by running multiple simulations and then examining the change in performance. Consequently the optimizer calculates the reward of the timetables with the following three scenarios:

- **Passenger increase:** The base amount of passengers which are generated by the simulator is increased by a factor (e.g. 1x, 2x, etc.)
- **Train increase:** The period of the train line is decreased (e.g. 1 train all 60 min, 1 train all 30 min, etc.) and therefore the amount of trains in the system increases.
- **Train breakdown:** A random train in the system is forced to stop for an increasingly long time (e.g. 5 min, 10 min, 30 min, etc.).

As can be seen in Table 4.1, the optimizer evaluates the reward of each population member by running a total of 12 simulations (4 for each scenario) per member. In a first step the total reward for all three scenarios is calculated. Figure 4.22 shows this process using the Train Increase scenario, but the process for the other two scenarios is the same.

Scenario	Parameter used in Optimizer	Description
Passenger increase	[1, 2, 4, 8]	*Multiple of base passenger volume*
Train increase	[1, 2, 3, 4]	*Number of train arrivals per hour at each station*
Train breakdown	[5, 30, 60, 120]	*Duration of a single train breakdown (min)*

Table 4.1.: Properties and scenarios used by the optimizer to test schedules for their robustness

Example of evaluating the performance of a schedule under changing circumstances

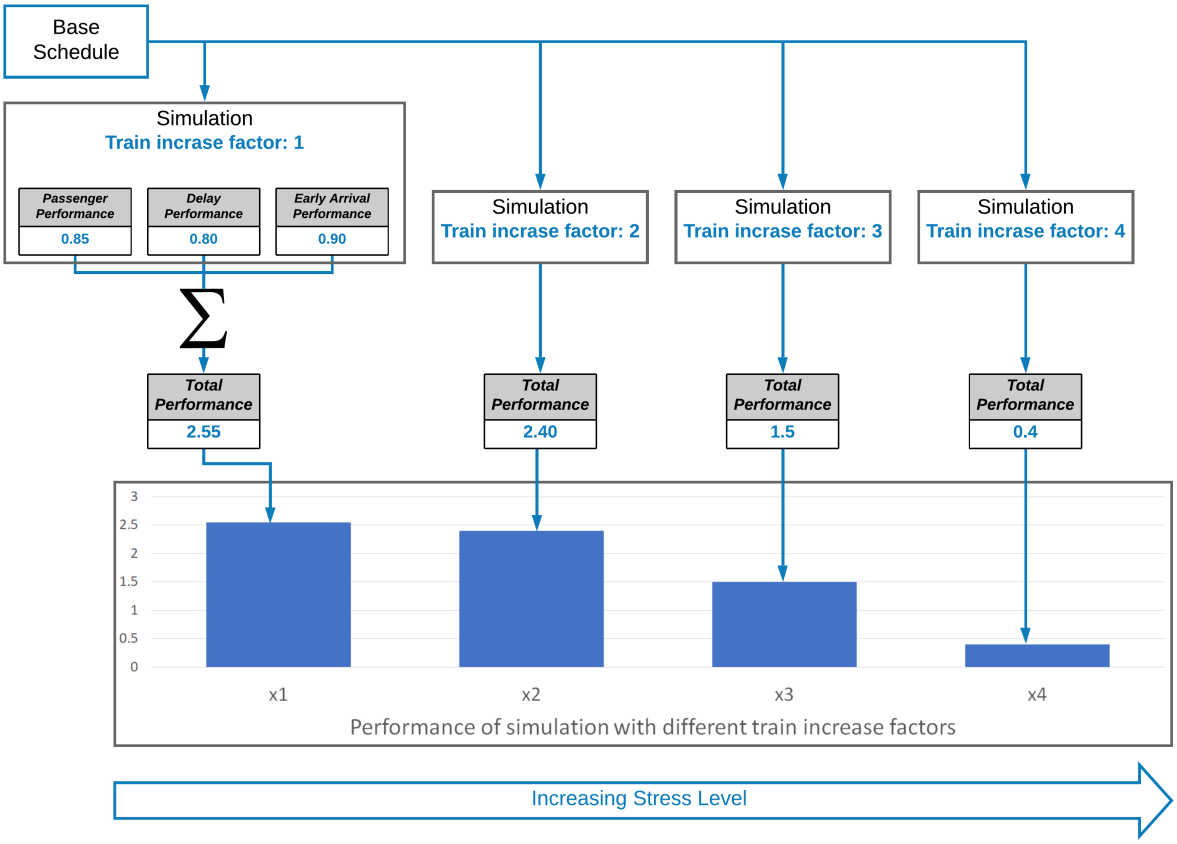


Figure 4.22.: Visualization of the performance measurement for different network conditions

The resulting performance measurements now contain information about the performance under normal network conditions (first bar) as well as under increasing stress. The re-

ward could then be calculated by summing all performance measurements, but this would not give control over whether the optimizer should create efficient or robust schedules. To achieve this, the individual scenario performance measurements are each weighted with an normalized weight vector. This is shown in Figure 4.23, where the efficiency weight vector considers only the performance measurement under normal conditions, while the robustness weight vector weights all performance measurements equally.

The total reward for a population member is then calculated by summing up the weighted reward for all three scenarios (see Figure 4.24).

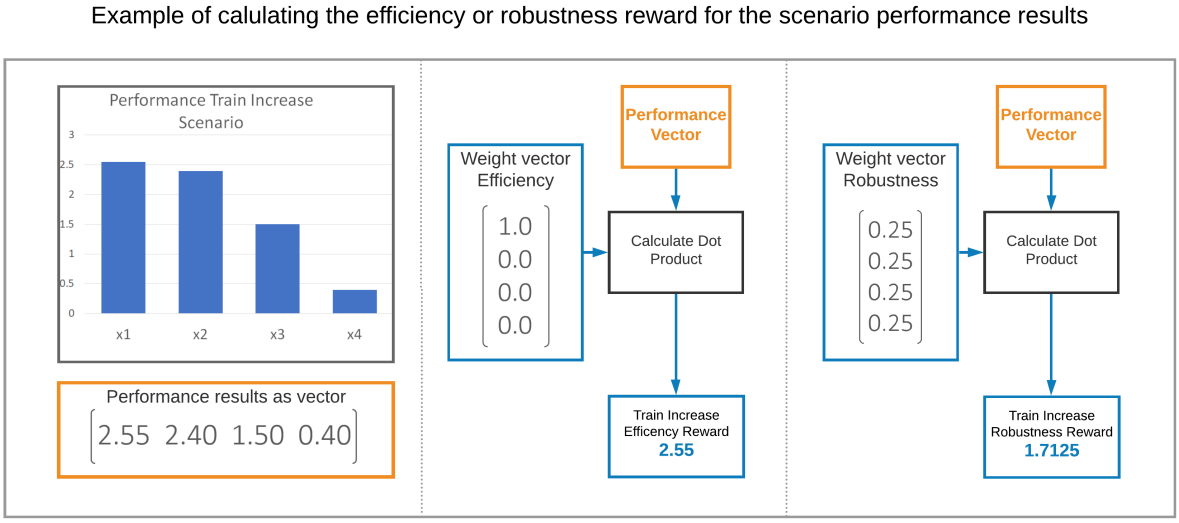


Figure 4.23.: Example for computing the efficiency and the robustness reward by calculating the dot product of the performance measurement and a weight vector

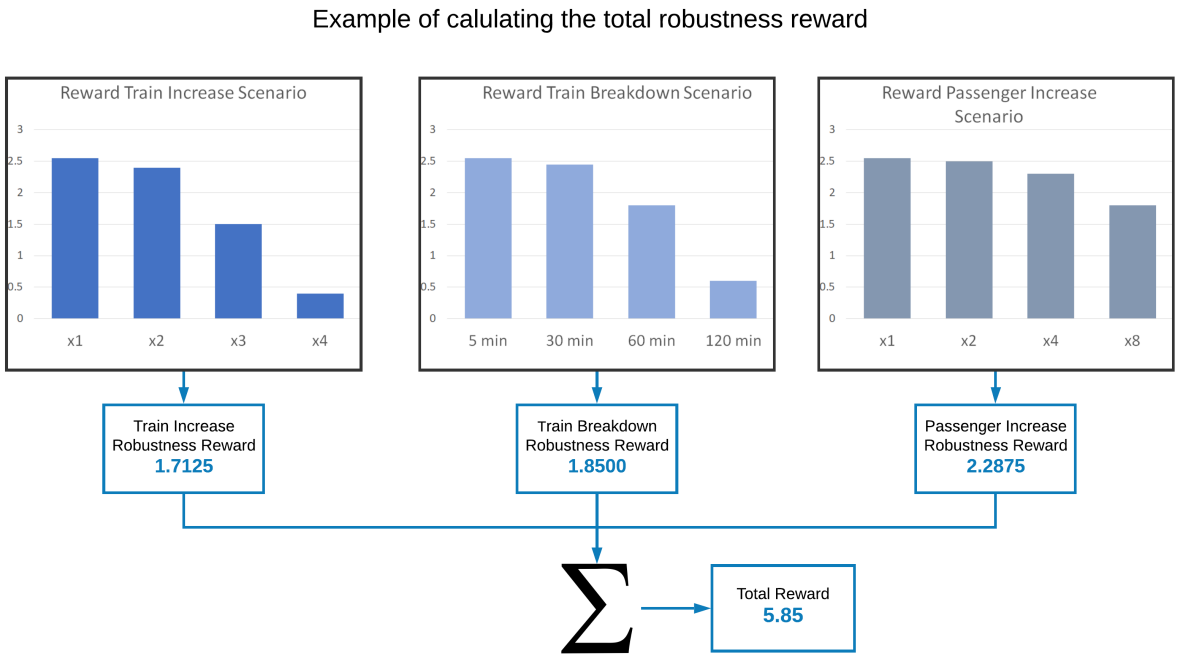


Figure 4.24.: Calculate the total reward for robustness by summing up the weighted reward for all three scenarios

4.2.4. Repopulating and genetic Operations

After the population is reduced to the 10% members with the highest reward (see Algorithm 1), it is repopulated by repeatedly selecting two random parents from the remaining members and combining their chromosomes to create new offspring (Figure 4.25).

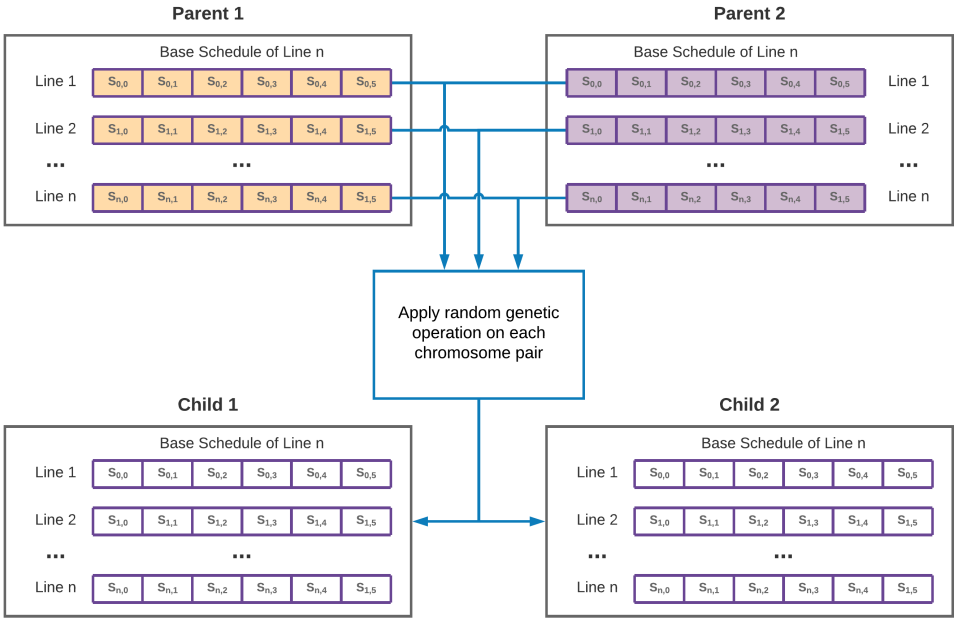


Figure 4.25.: Principle of creating offspring from two parents by applying genetic operations on each chromosome pair.

The process of the combining of two parents into offspring is performed by one of the following genetic operations on each pair of chromosomes:

- **Crossover:** A random part of the chromosome is swapped between the two parents.

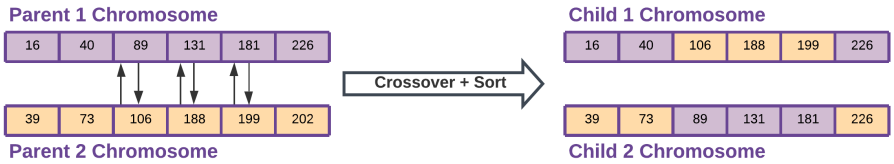


Figure 4.26.: Example of a crossover operation with any two parent chromosomes

- **Regeneration:** Both parent chromosomes are regenerated randomly.

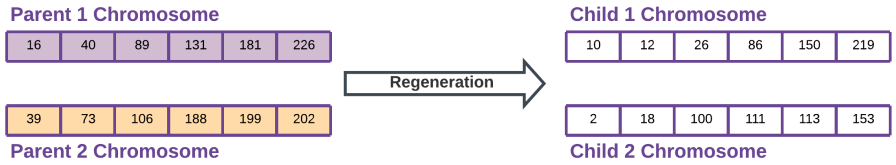


Figure 4.27.: Example of a regeneration operation with two arbitrary parent chromosomes

- **Swap:** The chromosomes of the parents are swapped as a whole. This operation should only be applied if the members have multiple chromosomes (e.g. network with multiple train lines), as with one chromosome, it would only create a copy of the parents.

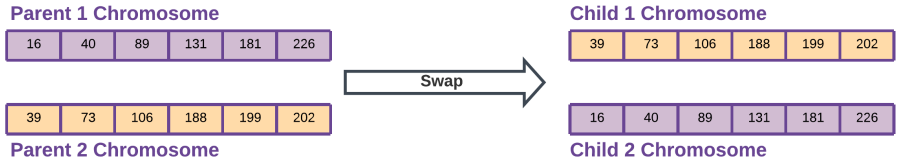


Figure 4.28.: Example of a swap operation with two arbitrary parent chromosomes

- **No Change:** Similarly to the Swap operation, this operation should only be applied when having members with multiple chromosomes. The result of the No Change operation is, that the schedule for a single line is left untouched while the schedules for the other lines are changed by different genetic operations.

Figure 4.29.: Example of a no change operation with two arbitrary parent chromosomes

- **Mutate:** Mutations change a random genome in the chromosome and afterwards sort the chromosome.

Figure 4.30.: Example of a mutate operation on an arbitrary parent chromosome

4.3. Experiment Results

To investigate and compare the behaviour of robust and efficient train schedules, a series of experiments on different sized networks where conducted using the simulator and optimizer described in Chapter 4. In each experiment, the optimizer was used on a network architecture to once create robust, and once create efficient schedules. This process was repeated with five different initial populations for both the robust and efficient optimization. By conducting multiple optimizations on the same networks, the deviation of the results can be calculated.

The detailed configuration of the used train networks can be found in the appendix, while the optimizer parameters used in all experiments are shown in Table 4.2. The analysis and discussion of the results takes place in Chapter 5.

Parameter	Value
Number of efficiency runs	5
Number of robustness runs	5
Generations per run	40
Mutation chance	0.1

Table 4.2.: Parameter for experiments

Legend Network Architecture Figure 4.31 shows the symbolism used to describe the physical architecture of the train networks used in the experiments.

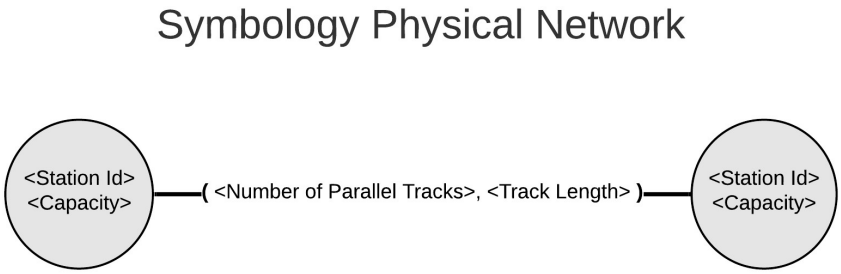


Figure 4.31.: Symbology used to describe the physical train network architecture

4.3.1. Experiment STAR-6S-2L

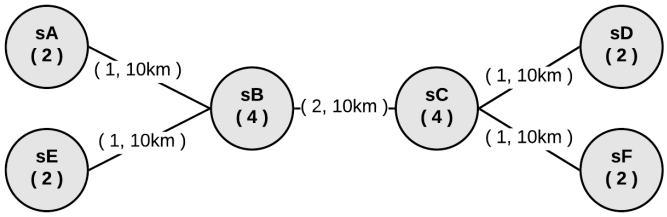


Figure 4.32.: STAR-6S-2L-01 network architecture

Figure 4.33.: Output of average reward for the ten best performing schedules in each generation

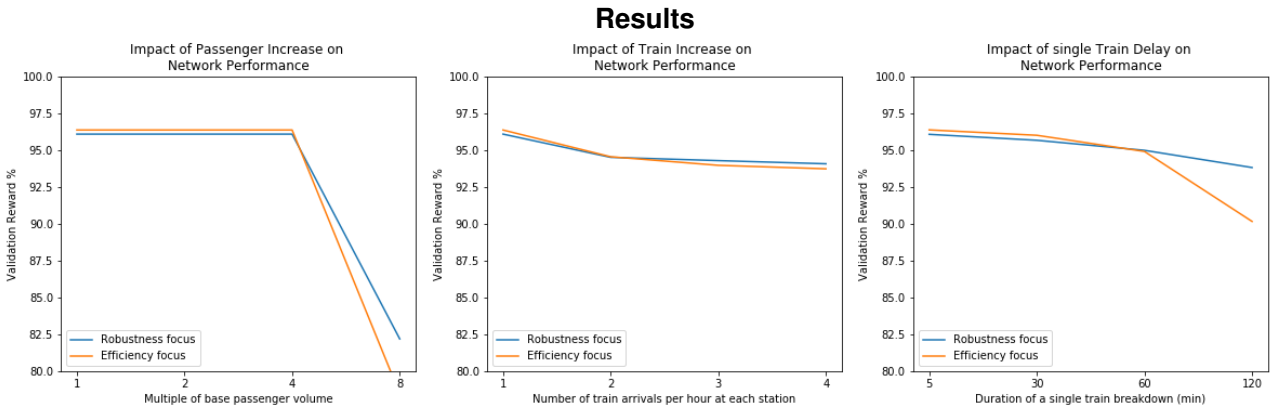


Figure 4.34.: Performance of schedules optimized for robustness and efficiency under different network conditions

4.3.2. Experiment STAR-9S-2L

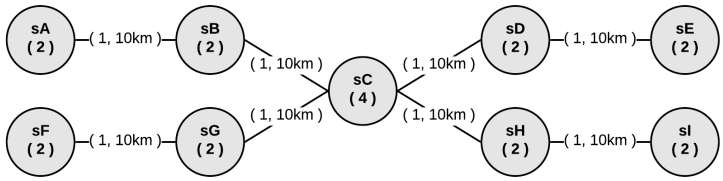


Figure 4.35.: STAR-9S-2L-01 network architecture

Figure 4.36.: Output of average reward for the ten best performing schedules in each generation

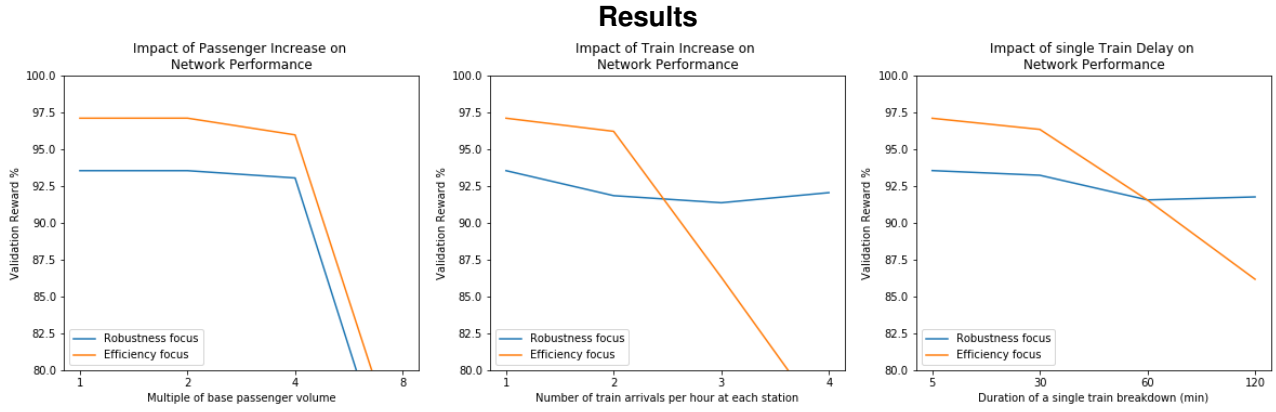


Figure 4.37.: Performance of schedules optimized for robustness and efficiency under different network conditions

4.3.3. Experiment STAR-10S-2L

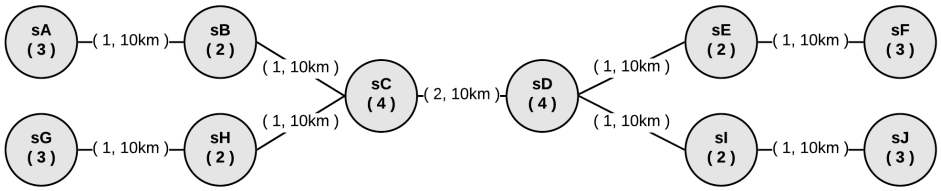


Figure 4.38.: STAR-10S-2L-01 network architecture

Figure 4.39.: Output of average reward for the ten best performing schedules in each generation

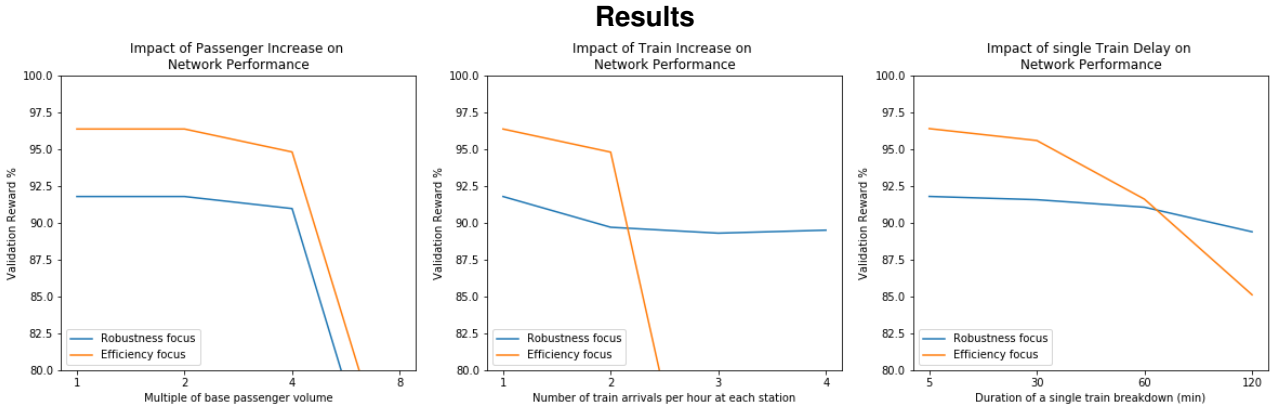


Figure 4.40.: Performance of schedules optimized for robustness and efficiency under different network conditions

4.3.4. Experiment STAR-14S-3L

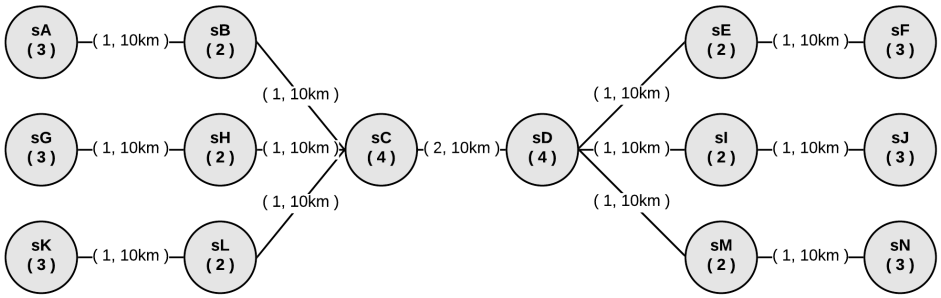


Figure 4.41.: STAR-14S-3L-01 network architecture

Figure 4.42.: Output of average reward for the ten best performing schedules in each generation

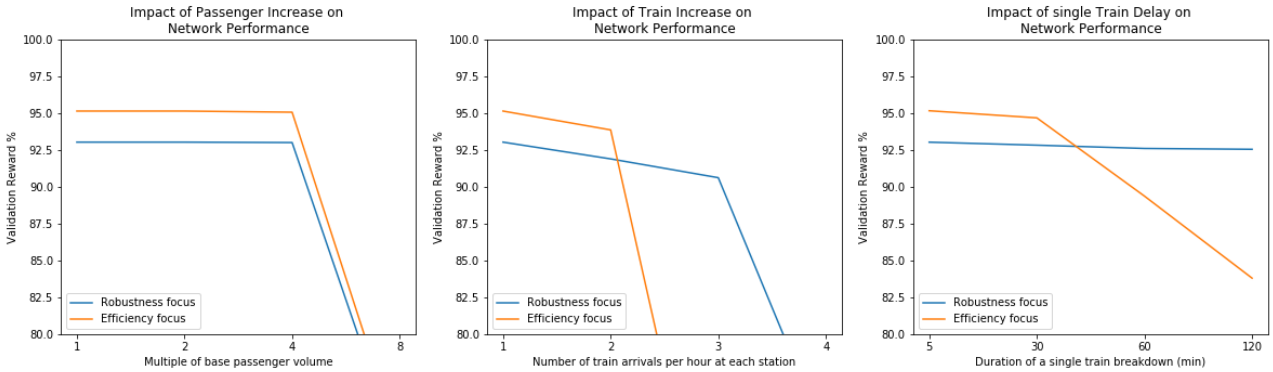


Figure 4.43.: Performance of schedules optimized for robustness and efficiency under different network conditions

5. Discussion of Experiment Results with focus on Trade-off between robust and efficient Timetables

The main objective of this project was to optimize train schedules for robustness and efficiency and then analyze the difference in behaviour between the robust and efficient schedules by measuring their performance under different network conditions. It was expected, that the efficient schedules would work better under normal network conditions but would have a rapid decline in performance under increasing network load. On the other hand, robust schedules would initially perform slightly worse than efficient schedules but would maintain their performance level for much longer and therefore outperform the efficient schedules on congested networks (see Section 3.2.3).

The experiment results documented in Section 4.3 showed, that for increasing amount of trains or delays, the expected behaviour could indeed be observed. In contrast, the increase in passenger numbers led to a similar rapid decline in performance for both robust and efficient train schedules. Under normal network conditions, the performance of efficient schedules was overall higher than that of robust schedules.

In the following sections, those results are analysed further by comparing the punctuality of timetables between different network sizes and by discussing the use of the findings for train schedule planning.

5.1. Comparison of punctuality between robust and efficient timetables for different sized Networks

To compare the punctuality between different network sizes, the best performing five robust and five efficient timetables from each experiment (see Section 4.3) were evaluated under normal and congested network conditions (3x the normal amount of connections per hour). Figure 5.1 shows the median punctuality for all sets of timetables under normal conditions while Figure 5.2 shows the median punctuality under congested conditions. The black indicators on top of each bar represents the 75 and 25 percentiles.

It can be seen, that under normal conditions, efficient timetables have a 0.84% to 1.81% higher median punctuality than their robust counterparts. Additionally, the variation in punctuality is lower for efficient timetables than for robust timetables. In contrast, Figure 5.2

shows that the punctuality of efficient schedules on the Star 14S and Star 10S networks falls below 60%, while robust schedules continue to achieve over 90% punctuality. This behaviour cannot be observed on the Star 6S and Star 9S networks, as the average punctuality only differs by 1.09% (Star 6S) and 0.21% (Star 9S) between efficient and robust timetables. This could indicate that larger congested networks benefit more from robust schedules than small congested networks. However, to confirm this assumption, further experiments on networks of different sizes would have to be carried out.

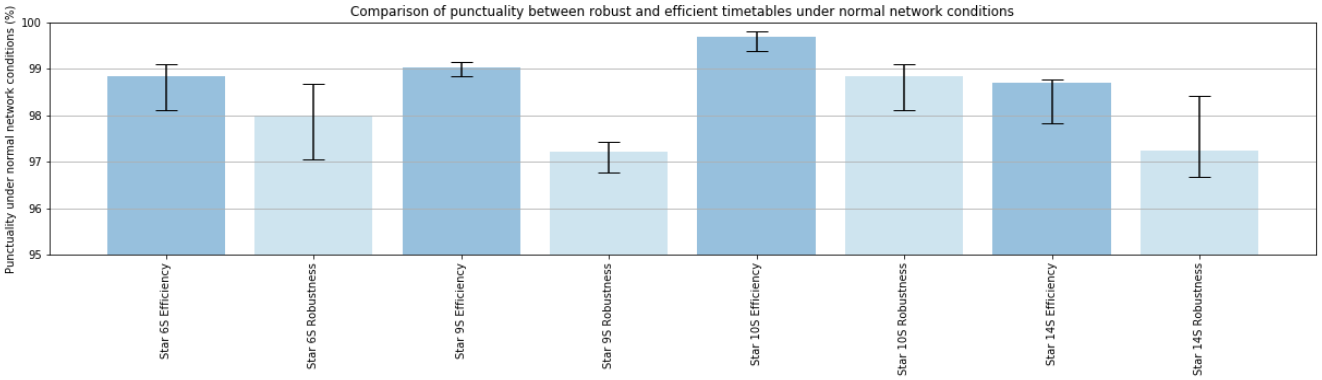


Figure 5.1.: Comparison of median punctuality of robust and efficient timetables for different star network sizes under normal conditions. The black indicator on top of each bar represents the 75 and 25 percentiles.

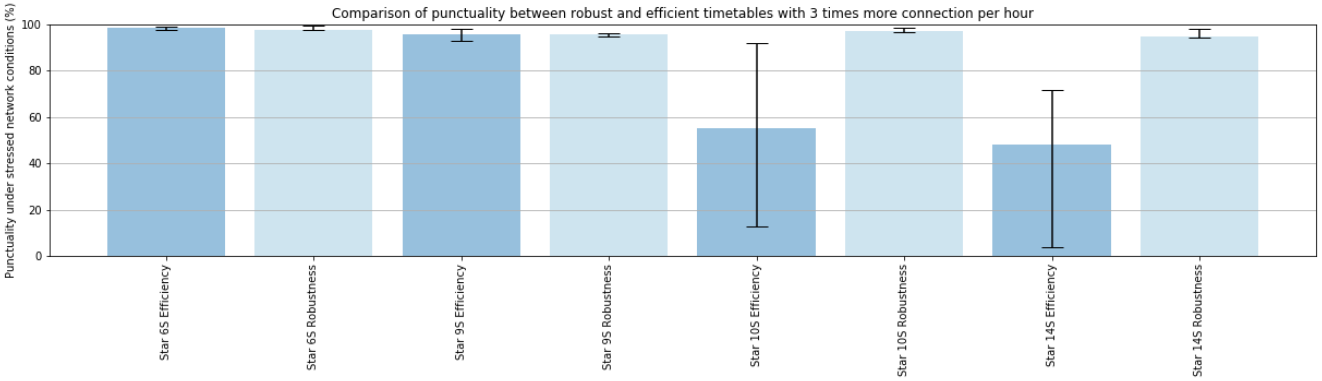


Figure 5.2.: Comparison of median punctuality of robust and efficient timetables for different star network sizes under stressed network conditions. The black indicator on top of each bar represents the 75 and 25 percentiles.

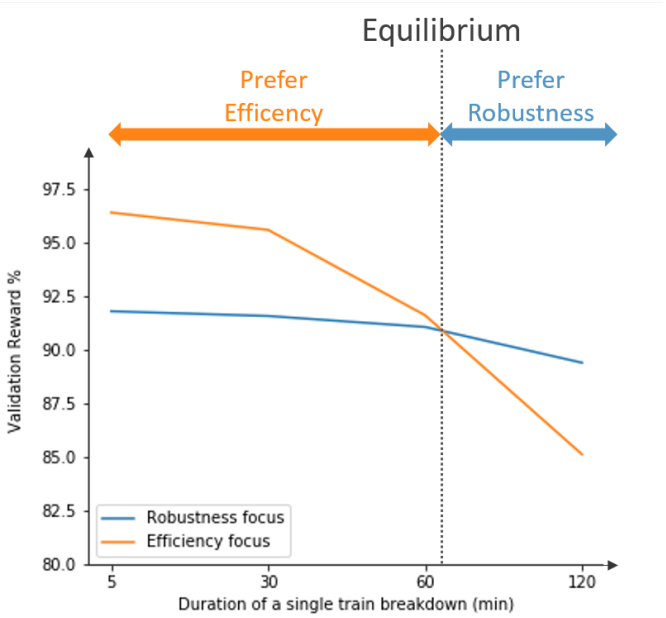


Figure 5.3.: Visualisation of the trade-off between robust and efficient schedules on the example of the STAR-10S-2L train network

5.2. Applicability of the results for schedule planning

An important finding of this project was, that under certain conditions (e.g. increasing amount of trains or delays) there exists a trade-off between robust and efficient schedules. An example for such a trade-off can be seen in Figure 5.3, where the schedule optimized for efficiency outperforms the robust one until the point where breakdown duration reaches more than 60 minutes. Therefore in this example, the schedules reach an equilibrium at 60 minutes breakdown time. By knowing this equilibrium a schedule planner can decide weather he should optimize for efficient or robust schedules. To give an example, if the schedule planner creates a schedule for a system which hardly ever experiences more than 30 minute breakdown times, he should aim for an efficient schedule. If, on the other hand, the system regularly experience long breakdown times (e.g. older rolling stock, old infrastructure, extreme weather conditions, etc.) a robust schedule would be more desirable.

Since the networks examined in this project were only small, artificial networks, it cannot be said whether the results also apply to their more complex, real-world counterparts. This would require more experimentation and research on more sophisticated simulations of real-world networks

5.3. Assessment of the chosen approach when implementing the simulator

The chosen simplifications for the train network simulator have shown to be effective in making the simulation time fast enough to generate optimized timetables in a feasible time frame. To illustrate, each of the four conducted experiments consisted of 432'000 simulated days and the average time it took to run an experiment was 3h 48 min[1]. Furthermore, modelling only the basic components of train networks made it possible to create networks without having extensive knowledge of real-world train network design.

In contrast, one of the disadvantages of the chosen simplifications was that the passenger management system did not allow the control and measurement of passenger movements at a low level. This is not an issue for the analysis of operational performance (e.g. punctuality), but it complicates the implementation of performance indicators related to the passenger experience (e.g. number of train changes between stations, waiting time at stations, etc.).

[1] Experiments were run on an Intel Core i7-9750H 6-Core CPU (2.60GHz)

6. Conclusion and Suggestion for further Research

In this thesis, we have implemented a simple train network simulator and combined it with genetic algorithms to optimize train schedules for robustness. By conducting experiments on different sized star networks, we showed that a simulation based approach with genetic optimization can yield robust schedules for small networks of up to fourteen stations. We furthermore compared the performance of efficient and robust schedules under different network conditions and presented the existing trade-off between robust and efficient train networks.

One of the main difficulty in the project was to create a simplified simulation of a complex real-world system, so that the computing time for the simulation is as short as possible, while at the same time modelling the most important behaviour of train networks. This required to model complex tasks such as traffic management, passenger movement, network design, etc. The complexity additionally led to many variables (e.g. passenger demand per hour per station, minimum dwell and turnaround times, train speeds, etc.) which have an influence on the performance of the network and therefore had to be fixed so that the experiments were comparable. Therefore, a lot of time went into the process of designing a network component, analysing the performance of the resulting simulation and then change the component to counteract unintended behaviour.

Since both the implemented genetic algorithm and the simulator can be used flexibly, they could find use in future projects to further investigate robustness in train scheduling. We propose the following project ideas to build on the results obtained in this work:

- By extending the simulator with a more sophisticated passenger management, additional performance indicators such as passenger satisfaction (e.g. travel time, number of train changes, etc.) could be utilized in the schedule optimization.
- Alternative optimisation methods from the field of artificial intelligence could be investigated. One method that shows potential and was considered in this project was reinforcement learning. Training a reinforcement agent on different network architectures can provide insights into whether such an agent can learn good timetable patterns for a given infrastructure and then apply them to new, unknown train networks.
- The genetic optimizer could be combined with a more sophisticated, commercial train network simulator to test its potential to deal with more realistic simulations of a real-world train network architecture. The resulting timetables could then be analysed together with a train timetable planner and compared with the timetable actually used.

Overall, the results of this project showed that simulation-based optimization methods have the potential to generate more robust timetables in overloaded train networks. In view of the challenges posed by increasing passenger demand in railway systems, it is worthwhile to conduct further research in this area.

List of Figures

List of Tables

Bibliography

Anderhub, G., Dorbritz, R., & Weidmann, U. (2008, December). *Leistungsfähigkeitsbestimmung öffentlicher Verkehrssysteme* (Report). ETH Zurich.

Arenas, D., Chevrier, R., Hanafi, S., & Rodriguez, J. (2015, March). Solving the train timetabling problem, a mathematical model and a genetic algorithm solution approach, In *6th international conference on railway operations modelling and analysis (RailTokyo2015)*, Tokyo, Japan.

BAV. (2014). *Dokumentation Planungsgrundlagen STEP Ausbauschritt 2030* (Sprachdienste BAV, Trans.; tech. rep.).

Beasley, J. (n.d.). *Integer programming*. Retrieved May 18, 2020, from http://people.brunel.ac.uk/~mastjjb/jeb/or/ip.html

Berkan, E. (2009, August 5). *Models for the train timetabling problem* (Diplomarbeit). Institut für Mathematik der Technischen Universität Berlin.

Borowiec, S. (2016). AlphaGo seals 4-1 victory over go grandmaster lee sedol. *The Guardian*. Retrieved May 17, 2020, from https://www.theguardian.com/technology/2016/mar/15/googles-alphago-seals-4-1-victory-over-grandmaster-lee-sedol

Bundesamt für Statistik, B. (2017). *Pendlermobilität* [Library Catalog: www.bfs.admin.ch]. Retrieved June 4, 2020, from https://www.bfs.admin.ch/bfs/de/home/statistiken/mobilitaet-verkehr/personenverkehr/pendlermobilitaet.html

Bussieck, M. (1998). Optimal Lines in Public Rail Transport. *Technische Universität Braunschweig*, 141.

Goossens, J.-W. (2004). Models and algorithms for railway line planning problems.

Goverde, R., & Hansen, I. (2013). Performance indicators for railway timetables, In *IEEE ICIRT 2013 - proceedings: IEEE international conference on intelligent rail transportation*.

Heidron Buttler. (2017). Wie entsteht der Fahrplan. Retrieved May 13, 2020, from https://www.gdi-adi.ch/fileadmin/user_upload/170301_Referat_GdI_-_wie_entsteht_der_Fahrplan.pdf

Infrastruktur, S. (2020). Linie mit Betriebspunkten [Library Catalog: data.sbb.ch]. Retrieved May 13, 2020, from https://data.sbb.ch/explore/dataset/linie-mit-betriebspunkten/

Khadilkar, H. (2019). A scalable reinforcement learning algorithm for scheduling railway lines. *IEEE Transactions on Intelligent Transportation Systems*, *20*(2), 727–736.

Koller, T., & Wullschleger, P. (2019, June 7). *Intelligenter Autopilot durch Reinforcement Learning* (Bachelorarbeit). Hochschule Luzern, Departement Informatik.

Lindner, T. (2000). Train Schedule Optimization in Public Rail Transport (W. Jäger & H.-J. Krebs, Eds.). In W. Jäger & H.-J. Krebs (Eds.), *Mathematics Key Technology for the Future: Joint Projects between Universities and Industry*. Berlin, Heidelberg, Springer.

Match 1 - google DeepMind challenge match: Lee sedol vs AlphaGo. (2016, March 8). Seoul. Retrieved May 16, 2020, from https://www.youtube.com/watch?v=vFr3K2DORc8&t=1h57m

Matti, T. (2019). Fahrplanentwicklung bei den SBB von 1929 bis 1985. Partizipation und Mobilitätsinteressen analysiert anhand von Fahrplanbegehren [Medium: application/pdf Publisher: Bern Open Publishing].

Rudolf Gäbertshahn. (1993). Der Integrale Taktfahrplan, In *Die Deutsche Bahn*.

SBB. (2020, May 27). *How punctual is SBB?* [Library Catalog: company.sbb.ch]. Retrieved June 1, 2020, from https://company.sbb.ch/en/the-company/responsibility-society-environment/customers/punctuality.html

SBB statistics [SBB statistics portal]. (2020, March 31). https://reporting.sbb.ch/

Serafini, P., & Ukovich, W. (1989). A mathematical for periodic scheduling problems [Place: USA Publisher: Society for Industrial and Applied Mathematics]. *SIAM J. Discret. Math.*, *2*(4), 550–581.

Strategisches Entwicklungsprogramm Eisenbahninfrastruktur. Ausbauschritt 2035 (2018, October 31).

Sutton, R. S., & Barto, A. G. (2015). Reinforcement Learning: An Introduction, 352.

Taha, H. A. (2014, May 10). *Integer programming: Theory, applications, and computations* [Google-Books-ID: DaSjBQAAQBAJ]. Academic Press.

A. Architecture of networks used in experiments

In the following sections, the architecture and the parameters for all networks used in the experiments are documented. The influence of the individual parameters on the network components can be read in chapter 4. The symbology used to describe the networks is explained in Figure A.1.

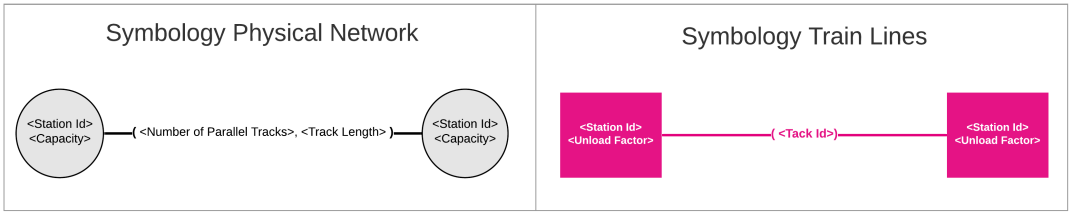


Figure A.1.: Symbology used to describe the network architecture and properties

A.1. Network STAR-6S-2L-01

A.1.1. Physical Network Architecture

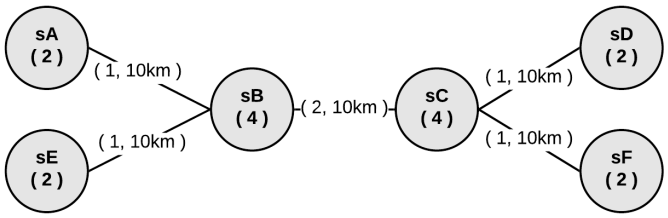


Figure A.2.: STAR-6S-2L-01 network architecture

A.1.2. Train Lines

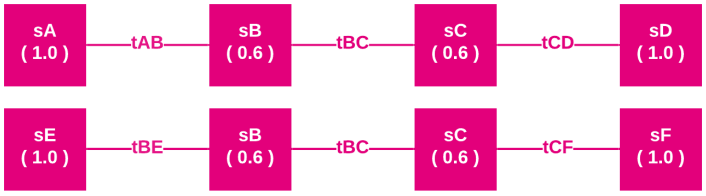


Figure A.3.: STAR-6S-2L-01 train lines

A.1.3. Additional Station Properties

	sA	sB	sC	sD	sE	sF
Destination Factor	1.0	0.8	0.8	1.0	1.0	1.0
Max passengers per hour	200	400	400	200	200	200

A.2. Network STAR-9S-2L-01

A.2.1. Physical Network Architecture

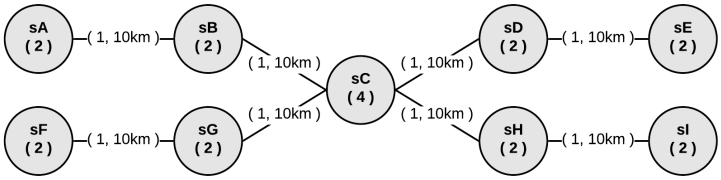


Figure A.4.: STAR-9S-2L-01 network architecture

A.2.2. Train Lines

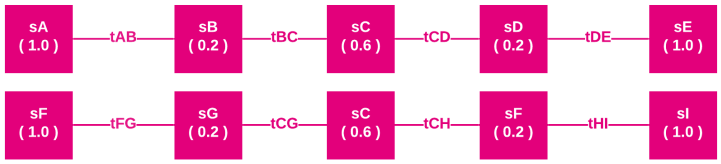


Figure A.5.: STAR-9S-2L-01 train lines

A.2.3. Additional Station Properties

	sA	sB	sC	sD	sE	sF	sG	sH	sI
Destination Factor	1.0	1.0	0.8	1.0	1.0	1.0	1.0	1.0	1.0
Max passengers per hour	200	200	400	200	200	200	200	200	200

A.3. Network STAR-10S-2L-01

A.3.1. Physical Network Architecture

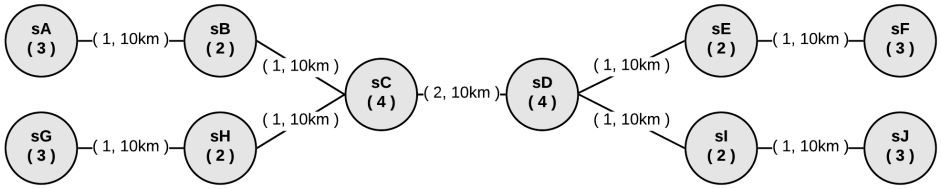


Figure A.6.: STAR-10S-2L-01 network architecture

A.3.2. Train Lines

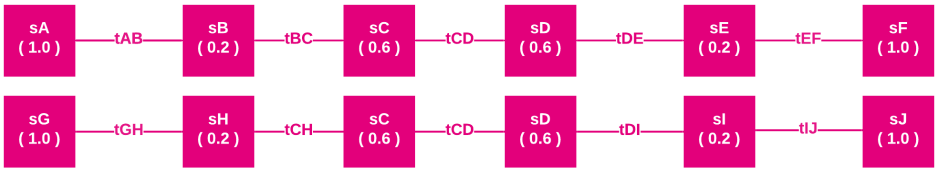


Figure A.7.: STAR-10S-2L-01 train lines

A.3.3. Additional Station Properties

	sA	sB	sC	sD	sE	sF	sG	sH	sI	sJ
Destination Factor	1.0	1.0	0.8	0.8	1.0	1.0	1.0	1.0	1.0	1.0
Max passengers per hour	200	200	400	400	200	200	200	200	200	200

A.4. Network STAR-14S-3L-01

A.4.1. Physical Network Architecture

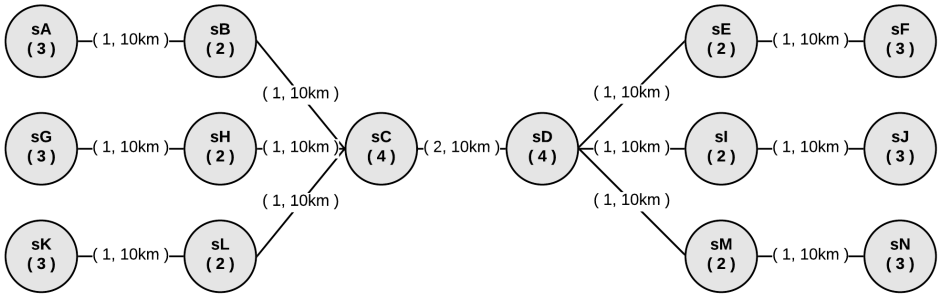


Figure A.8.: STAR-14S-3L-01 network architecture

A.4.2. Train Lines

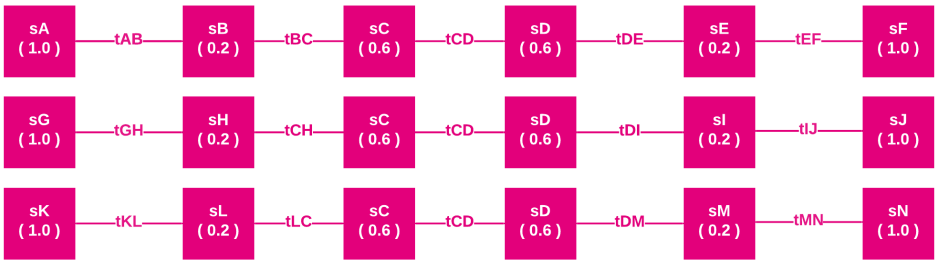


Figure A.9.: STAR-14S-3L-01 train lines

A.4.3. Additional Station Properties

	sA	sB	sC	sD	sE	sF	sG	sH	sI	sJ	sK	sL	sM	sN
Destination Factor	1.0	1.0	0.8	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Max passengers per hour	200	200	400	200	200	200	200	200	200	200	200	200	200	200

B. Project Management

B.1. Project Milestone Plan

The project milestone plan visualizes all milestones and phases of the project. Because the time frame for the given project was fixed, a waterfall model was chosen for the project planning. The plan incorporates the following 4 Phases:

- Initialization (MS01 Initialization)
- Simulator (MS02 Half-Time Presentation)
- Optimizer (MS03 Optimization)
- Completion (MS04 Submission)

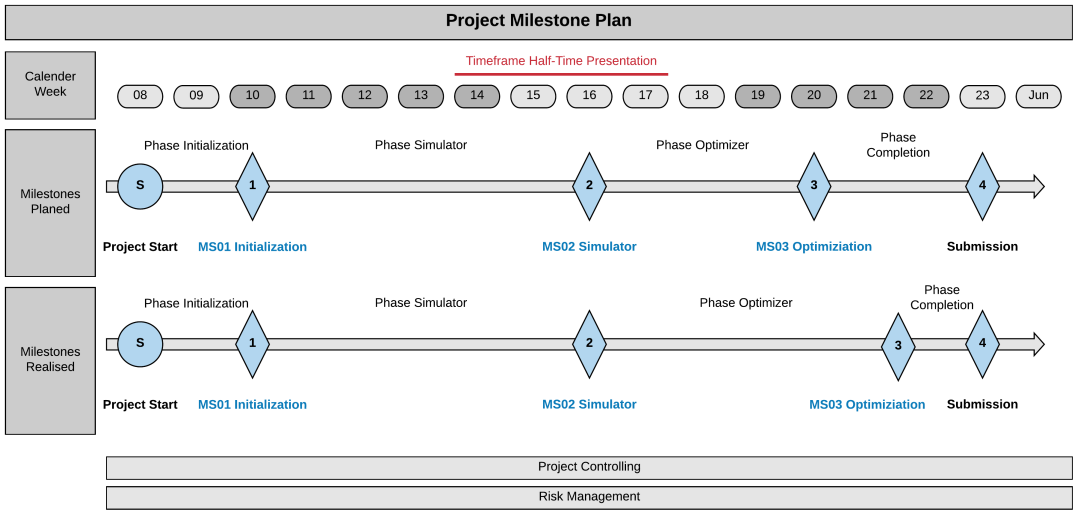


Figure B.1.: The project milestone plan visualizes the four phases of the project.

The original milestone plan had scheduled the completion of the optimizer in week 20. But because the original idea of a reinforcement learning agent was changed to a genetic optimizer, two additional weeks were added to the optimization phase in consultation with the supervisor. To compensate this, the documentation overhaul, which was planed in the completion phase, was also started two weeks earlier (see Appendix B.2).

B.1.1. Milestones

Milestone 1 - Initialization

Milestone 1 is considered done if the following deliverables are done and checked:

- Project reference plan
- Project schedule
- Milestone definitions
- Documentation structure (LaTex)

Date: 27.02.2020 | Calendar week 9

Deliverable:

- Project reference plan
- Project schedule
- Documentation template

Milestone 2 - Simulator

Milestone 2 is considered done if the following tasks are done and checked:

- Model/Abstraction for simulation is defined and implemented
- Simulation is implemented with the defined model
- Performance indicators for train networks are defined
- Results of simulation without optimization are done
- Halftime presentation is prepared and scheduled

Date: 16.04.2020 | Calendar week 16

Deliverable:

- Halftime presentation done
- Implementation of simulation environment

Milestone 3 - Optimization

Milestone 2 is considered done if the following tasks are done and checked:

- Optimizer is implemented and tested
- Results of optimization are visualized
- Results are analysed, interpreted and compared to results of other optimization methods

Date: 26.05.2020 | Calendar week 22

Deliverable:

- Simulation environment with optimizer implemented
- Results of experiments

Milestone 4 - Submission

Milestone 2 is considered done if the following tasks are done and checked:

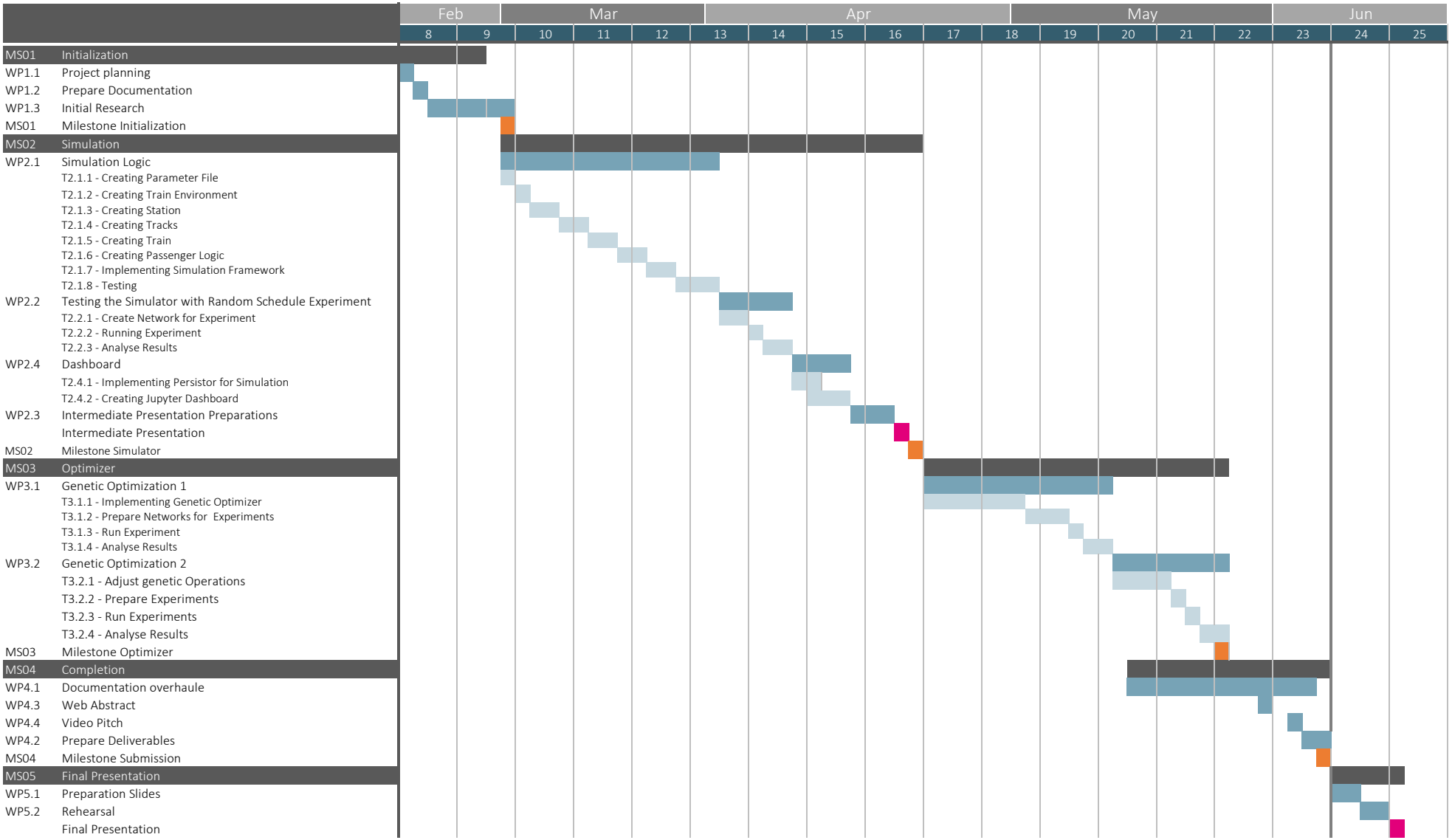
- Project documentation is finished and corrected
- Pitching Video is edited
- Code is cleaned and commented
- All deliverables are uploaded to the portfolio database

Date: 05.06.2020 | Calendar week 23

Deliverable:

- Project documentation (Portfolio Database + Expert)
- Pitching video
- Web abstract
- Source code / Raw data

B.2. Project Management Plan


Feb
Mar
Apr
May
Jun
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
MS01 Initialization
WP1.1 Project planning
WP1.2 Prepare Documentation
WP1.3 Initial Research
MS01 Milestone Initialization
MS02 Simulation
WP2.1 Simulation Logic
T2.1.1 - Creating Parameter File
T2.1.2 - Creating Train Environment
T2.1.3 - Creating Station
T2.1.4 - Creating Tracks
T2.1.5 - Creating Train
T2.1.6 - Creating Passenger Logic
T2.1.7 - Implementing Simulation Framework
T2.1.8 - Testing
WP2.2 Testing the Simulator with Random Schedule Experiment
T2.2.1 - Create Network for Experiment
T2.2.2 - Running Experiment
T2.2.3 - Analyse Results
WP2.4 Dashboard
T2.4.1 - Implementing Persistor for Simulation
T2.4.2 - Creating Jupyter Dashboard
WP2.3 Intermediate Presentation Preparations
Intermediate Presentation
MS02 Milestone Simulator
MS03 Optimizer
WP3.1 Genetic Optimization 1
T3.1.1 - Implementing Genetic Optimizer
T3.1.2 - Prepare Networks for Experiments
T3.1.3 - Run Experiment
T3.1.4 - Analyse Results
WP3.2 Genetic Optimization 2
T3.2.1 - Adjust genetic Operations
T3.2.2 - Prepare Experiments
T3.2.3 - Run Experiments
T3.2.4 - Analyse Results
MS03 Milestone Optimizer
MS04 Completion
WP4.1 Documentation overhaule
WP4.3 Web Abstract
WP4.4 Video Pitch
WP4.2 Prepare Deliverables
MS04 Milestone Submission
MS05 Final Presentation
WP5.1 Preparation Slides
WP5.2 Rehearsal
Final Presentation

C. Original Task Definition

Task Definition
Definitive registration to the Business Project/Bachelor thesis

1. **Start date:**
   Latest possible start date: autumn semester CW 38; spring semester CW 8
   17.02.2020

2. **Deadline:**
   Duration of a BAA: max. 15/16 calendar weeks (In order to be admitted to the regular graduation in summer, the submission must be made by Friday, one week after the end of the semester, at the latest.)
   Duration of a WIPRO: max. 14/15 calendar weeks
   05.06.2020

3. **Students:**

	Student 1:	Student 2:
Surname, first name:	Reust, Kevin	
Major:	Artificial Intelligence & Visual Computing	
Mobile:	+41 79 599 59 27	
E-Mail:	kevin.reust@stud.hslu.ch	
Project with employer (part-time students)	☐ Yes ☒ No	☐ Yes ☐ No

4. **Client:**

Company:	HSLU-I
Contact person:	Daly, Donnacha
Function:	
Street:	Suurstoffi 1
Post code / city:	CH-6343
Telephone:	
Email:	donnacha.daly@hslu.ch
Website:	https://www.hslu.ch/de-ch/informatik/

5. **Supervisor:**

Daly, Donnacha

6. Task Definition

Title:	AI based design of resilient scheduling and routing algorithms for train networks
Initial situation and problem definition:	Time-tabling of railway networks is an old problem, which is extremely well studied and understood. Switzerland is clearly a great example, having a railway network which functions very well in delivering punctual, densely connected and frequent train services across the nation. Nevertheless, there are two pressing issues which confound scheduling planners and need good solutions for future time-tabling: A) the number of passengers is increasing at a fast rate which requires more trains running more frequently on the same network and as a corollary B) the impact of delays in a more highly utilized network is more significant, with cascading effects causing rolling delays in less resilient (over congested) parts of the network. In this project the student will explore methods in artificial intelligence for scheduling and routing in congested networks, with a view to addressing these two problems, in a simulated setting.
Aim of the work and expected results:	The aim of the work is in three parts: 1) To build a realistic, scalable model which is flexible in applicability for different network types and traffic usage. 2). To optimize the network under different settings, such that implementable solutions for time-tabling can be found. And 3) to define and measure resilience and robustness metrics for the network which can also be used in the cost function for network design such that the resulting timetables are not the most efficient, but have a suitable trade-off between efficiency and for example rolling delays in the case of a disturbing event.
Methods/procedure:	The student is expected to have an interest in network planning, but no experience is necessary. However, the student should have a good grasp of optimization and artificial intelligence. The methods and procedure will focus on 1) Network Modelling 2) Network Performance Metrics and 3) Network Optimization using AI methods. The supervisor is agnostic to programming language but would recommend Matlab or Python (Numpy/Scipy) for the implementation.
Creativity, variations, innovation*	The models and network simulation will be created from scratch and is not driven by products and tools. This is a well-studied problem, and so it is unlikely that the student will find solutions which are unknown to the railway planning community. Nevertheless, it is hoped that resilience and robustness testing will throw some new lights on corner cases which arise in stressed networks
Keywords:	Scheduling, routing, railways, trains, networks, planning, robustness, resilience, congestion
Business project or bachelor thesis:	☐ Business project: 180 hours per student ☒ Bachelor thesis: 360 hours

*Please emphasize the extent to which your project idea has creative scope. The following criteria are relevant: The idea allows the students to develop their own ideas and variants, is located outside of the daily business, contains innovation and is not driven by products & tools.

Please tick one of the project types and the relevant focal points:

Project types:

- ☐ Use of standard software and services
- ☐ Software and product development
- ☒ Innovation projects (projects with knowledge gains, research projects)
- ☐ IT infrastructure development
- ☐ Structured analysis and conception of systems and processes

Focal Points:

- ☒ Artificial intelligence & machine learning
- ☐ Business process modelling
- ☐ Data engineering
- ☐ Hardware-oriented software development
- ☐ Human computer interaction design
- ☐ ICT business solutions
- ☐ ICT infrastructures
- ☐ Internet of things
- ☐ Mobile systems
- ☐ Security/Privacy
- ☐ Software development
- ☐ Visual computing (graphic, image processing, vision, VR, AR)
- ☐ other: Applications in Transport

7. Time management

Proposal for time division per person

WIPRO:		**BAA:**	
Per week:	ca. 12h	Per week:	ca. 20h
For the exam:	ca. 10h	Final week:	ca. 50h
Total:	180 h	For the exam:	ca. 10h
		Total:	360 h

8. Legal bases and regulations

The following legal bases and regulations are authoritative for business projects and bachelor theses at the Lucerne University of Applied Sciences and Arts – Information Technology:

- Studienordnung für die Ausbildung an der Hochschule Luzern, FH Zentralschweiz (Link)
- Studienreglement für die Bachelor-Ausbildung an der Hochschule Luzern - Informatik (Link)

9. Confirmation

By taking note of the task definition, the student and the client confirm that

- You agree with the task definition
- The client agrees that the Lucerne University of Applied Sciences and Arts – Information Technology charge him/her a fee of CHF 1000.00 (incl. VAT) per student for the organization of a bachelor thesis. This does not apply to work done by part-time students in conjunction with their employer and to HSLU internal clients. For the business projects, no cost contribution will be charged.
- Supervisors and experts gain unrestricted insight into the work. The work can also be shown to the public for presentations and marketing activities. A summary of the work will be published in any case. If the work has to be treated confidentially a corresponding confidentiality agreement must be enclosed with this form.

Date: _____

Please send the definitive task definition (pdf format) by e-mail to the transfer services and in copy to all parties involved.

Contact point for all information in connection with student works as well as for acceptance of project ideas and task definitions:

Hochschule Luzern - Informatik
Transfer Services
Suurstoffi 41b
6343 Rotkreuz
T: 041 228 24 66
E: transfer.informatik@hslu.ch