Disruptions in Public Transport: From Dispatching Assistance to Improved Robustness

Dissertation
zur Erlangung des Doktorgrades der Naturwissenschaften (Dr. rer. nat.)

der

Naturwissenschaftlichen Fakultät III Agrar- und Ernährungswissenschaften, Geowissenschaften und Informatik der Martin-Luther-Universität Halle-Wittenberg

vorgelegt von

Herrn Ralf Rückert
Geb. am 27.01.1987 in Merseburg

1. Reviewer	Prof. Dr. Matthias Müller-Hannemann Martin-Luther-Universität Halle-Wittenberg
2. Reviewer	Prof. Dr. Marc Goerigk Universität Siegen

Verteidigungsdatum: 09.02.23

Zusammenfassung

Der öffentliche Personenverkehr (ÖPV) ist eine wichtiger Teil der modernen Mobilität. Aus operativer Sicht besteht der ÖPV auf vielen einzelnen Komponenten, welche ein komplexes zeitabhängiges System von Beförderungsleistungen ergeben, die auf einem Fahrplan, einer Wagenumlauf- und einer Personaleinsatzplanung beruhen. Die meisten dieser Komponenten inklusive der Infrastruktur sind anfällig für Störungen, welche alle anderen beteiligten Komponenten beeinflussen. Für Reisende stellen Störungen ein großes Problem dar, welches im schlimmsten Fall dazu führen kann, dass Personen vom ÖPV zu privaten Alternativen wechseln.

In dieser Arbeit geht es um die gezielte Bekämpfung der Effekte von Störungen. Wenn diese auftreten, versuchen Disponenten durch schnelles Eingreifen dafür zu sorgen, dass der Schaden in Form von Verspätungen minimiert wird. Diese Arbeit unterstützt Disponenten auf zwei Wegen. Sie liefert erstens Methoden zur Optimierung der Entscheidungsfindung bei der Anwendung von Maßnahmen und versucht zweitens die Robustheit von Fahrplänen so zu verbessern, dass bei gleichartigen Verspätungen von Fahrzeugen geringere Verspätungen von Reisenden am Ziel entstehen.

In der Arbeit wird Werkzeug zur reisendenorierter Anschlussunterstützung für das Halten von Anschlüssen (PANDA) entwickelt. Diese simuliert verschiedene Alternativen und leitet daraus eine Empfehlung ab, ob ein Fahrzeug auf Anschlussreisende aus einem verspäteten Zubringer warten soll. Diese Entscheidungsunterstützung wird erstmals als Online-Problem implementiert, was gleichzeitig eine Quantifizierung der Verspätungen am Ziel und eine Visualisierung der Auswirkungen für Disponierten liefert. Wir gehen im weiteren Verlauf dieser Arbeit auf Aspekte wie Verlässlichkeit der Empfehlung sowie das Koppeln mit benachbarten Konflikten ein. Einflussfaktoren wie die Festlegung, wann eine Entscheidung getroffen wird, werden ebenfalls analysiert. Die Software zur Entscheidungsunterstützung wurde umfangreich mit Fahrplänen und Störungsdaten der Deutschen Bahn erfolgreich getestet.

Um im einem zweiten Hauptteil der Arbeit die Robustheit von Fahrplänen zu optimieren, erstellen wir eine Reihe von empirischen Robustheitstests, welche klassische Arten von Störungen simulieren. Mit Hilfe einer Metrik und systematischen Auswertungen kann nun die Robustheit von unterschiedlichen Fahrplänen für dasselbe Netzwerk verglichen werden. Die Resultate der Vergleichbarkeit bezüglich Robustheit werden anschließend genutzt, um bereits während der Optimierung bei der Erstellung von Fahrplänen mit Hilfe von maschinellem Lernen robustere Lösungen zu generieren. Auf verschiedenen Benchmark Testinstanzen konnten bestehende Fahrpläne signifikant verbessert werden.

Abstract

Public transport is an important part of modern mobility. From an operational point of view, public transport consists of many individual components, which result in a complex time-dependent system of transport services based on a timetable, vehicle circulation and staff scheduling. Most of these components, including the infrastructure, are susceptible to disruptions, which affect all other components involved. For travelers, disruptions are a major problem, which in the worst case can lead to people switching from public transport to private alternatives.

This thesis is about targeting the effects of disruptions. When they occur, dispatchers try to ensure that the damage in terms of delays is minimized through quick intervention. This thesis supports dispatchers in two ways. First, it provides methods for optimizing decision making in the application of interventions, and second, it attempts to improve the robustness of schedules such that similar delays of vehicles result in fewer delays of travelers at their destinations.

In the thesis, a tool for passenger-aware dispatching assistance (PANDA) for maintaining connections is developed. This simulates different alternatives and derives a recommendation whether a vehicle should wait for connecting travelers from a delayed feeder. This decision support is implemented for the first time as an online problem, which simultaneously provides a quantification of delays at the destination and a visualization of the impact of decisions on travelers for dispatchers. We address aspects such as the reliability of the recommendation as well as coupling with neighboring conflicts in the remainder of this thesis. Influencing factors such as determining when to make a decision are also analyzed. The decision support software has been extensively and successfully tested with timetables and disruption data of Deutsche Bahn.

In order to optimize the robustness of timetables in a second main part of the thesis, we create a set of empirical robustness tests simulating classical types of disruptions. Using a metric and systematic evaluations, the robustness of different timetables for the same network can now be compared. The results of the comparability in terms of robustness are then used to generate more robust solutions during timetable optimization by adding machine learning methods. On different benchmark test instances, existing timetables could be significantly improved.

Acknowledgement

First I have to thank all co-authors involved in the many publications that comprise most parts of this work.

Christoph Blendinger
Markus Friedrich
Martin Lemnian
Steffen Schüler
Alexander Schiewe
Sebastian Schmidt
Anita Schöbel

Most of all, I would like to thank Prof. Dr. Matthias Müller-Hanneman, who made this research possible. You connected (with) researchers and Deutsche Bahn officials. You asked the right questions, had great ideas, respected every input and concern, and finalized our efforts to complete papers in time.

I want to thank the rest of all FOR2083 members for their excellent collaboration and input on our work.

This research was only possible through funding by Martin Luther University, the DFG, and the Deutsche Bahn.

I am also grateful to thank Frank Berger for improving the real-world usability of our software to an unexpected extent.

Finally, and most importantly, huge thanks to my wife Sarah and my family for their support.

Contents

Introduction 1

The only way of catching a train I have ever discovered is to miss the train before.

— **G.K. Chesterton [Che09]**

1.1 Motivation

Whenever we need to travel any distance further than a few hundred meters, we ask ourselves: "How do I get there?" The process of answering such questions is a very complex one. The difficulty of answering them rises with the number of possibilities available. Habit, time, comfort, price and reliability may dominate our choices, either consciously or unconsciously. For a significant part of trips, public transport is an option. In Germany alone, 11 billion journeys per year are made using public transport [Dsb15].

The usage of public transportation has many benefits compared to car use. Buying and maintaining a car can be a substantial and unnecessary investment of money and time. The environmental impact of public transport, even when powered by fossil fuels is significantly lower than using an automobile [Lin10]. Public transportation that uses separate infrastructure is often faster and cheaper, especially in densely populated areas, as well as for traveling between major cities. Despite these advantages, private cars dominate the mobility of German citizens. Only 38% of the working population use public transport for commuting. While comfort and habits influence the mode choice of commuters, a study [Ste18] identifies two reasons that discourage passengers from using public transportation. These two reasons are the availability of fast connections and the reliability of services. Another result from this poll conducted in 2018 confirmed that 52% of car users would consider using public transportation if the trips were faster. Moreover, only 62% of passengers are satisfied with their current duration of travel [Ste18].

The reliability of the existing services is another matter. While regional public transport companies often have a good reputation, long-distance connections suffer from more substantial problems. In 2018 and 2022 one in four German long-distance trains was more than five minutes late [Rö19; Ver22]. This situation leads to passengers often missing their connecting trains. In this case, it is the job of dispatchers of transport providers to decide, whether connecting trains should depart or wait for changing passengers. Some of those situations could be improved if dispatchers would make decisions optimized for passenger satisfaction. In other cases, problems arise from poorly designed schedules containing connections with little robustness. This dissertation will introduce improvements on both fronts.

Funds and data for the research in this dissertation came from two major projects. The first is a cooperation between the Martin Luther University Halle-Wittenberg and the Deutsche Bahn

Fernverkehr AG. This cooperation started in 2011 with the main focus on improving dispatching decisions concerning the effects of delayed trains. The second important project is the FOR2083, an interdisciplinary research unit, funded by the Deutsche Forschungsgemeinschaft (DFG). The group started in 2015 and contains researchers working on different topics. The project focuses on integrated planning for public transportation.

Before diving into the structure of this dissertation, the goal and contribution of this work to research and practice in public transport will be outlined.

1.2 Goals and Contribution

The work done throughout the research covered in this dissertation focuses on two primary goals: The first goal is to enhance the process of passenger-oriented train dispatching in case of disruption. As a result, we provide dispatchers with a novel framework for making informed decisions. The main beneficiaries of such a framework and improved dispatching are the passengers. Using large scale simulations, the benefits to the passengers become quantifiable.

The second goal is to decrease the effects of disturbances in general by improving the process of creating more robust schedules. Work concerning the optimized creation of line plans and timetables often is under scrutiny from practitioners. The reason for being skeptical of plans outperforming their manually curated schedules is because an over-optimization can lead to situations where small disruptions have severe consequences. While the robustness concerning the recovery of the original schedule has been investigated [Lie+09], measuring the robustness of passenger journeys in the context of possible alternative schedules/line plans will be improved by the work introduced.

To achieve our goals, we made several contributions trying to advance current practices in public transport. The first achievement and basis for later work was our creation of a software-prototype helping connection dispatchers improve their work. The second contribution consists of several papers, often combining advancements in algorithms and usage of big data to enhance our understanding of problems in public transport. This dissertation contains structured insights into these papers as well as additional material. Those papers presented were accepted in peer-reviewed conferences [Lem+14; Rüc+15; LMHR16; Fri+17b; Fri+18; MHRS19; Mül+21b] and journals [Rü+17; MHR17; MH+22]. The combined content of these publications introduce new methods in public transportation.

- We improve passenger-oriented dispatching by building a prototype software for dispatchers.
  - It can make recommendations for online dispatching decisions.
  - It gives advice for rerouting during the dispatching.
  - It can evaluate whether small changes in passenger flows may change a recommendation.

- We show that it is possible to make waiting recommendations that include the cascade of resulting conflicts. We reveal how fast combined waiting recommendations become unlikely to yield a benefit.

- We present a novel approach to evaluate the robustness of multiple variants of a schedule, making it possible to select those handling defined common delay scenarios superiorly. The metrics utilized rely on passenger utility rather than vehicle delays.

- We use our approach on increased robustness and feed previously gained knowledge to a machine learning model. Once the model is created, it becomes possible to improve robustness during optimization with a minimal increase in running time. This method produces schedules that are competitive in terms of average travel time but superior in robustness.

1.3 The Author's Contribution

Before giving the structure of this dissertation this section will give a detailed account of the contribution of the author of this dissertation to the scientific papers covered here. For all of these papers, the author contributed to the design of research questions and hypotheses. He also was responsible for work on implementation and evaluation of necessary experiments. The amount of work on the implementation differs from paper to paper but Table 1.1 shows a detailed summary on this account.

paper	theory	implementation and evaluation
[Lem+14]	share of design	minor implementation and evaluation
[Rüc+15; Rü+17; MHR17]	share of design	majority of front-end implementation, experiments and evaluation
[LMHR16]	share of design	implementations of ILP, experiments and evaluation
[Fri+17b; Fri+18]	share of design	implementations, experiments and evaluation of tests
[Mül+21b; MH+22]	share of design	evaluation of tests, experiments and creation of ML-oracle

Tab. 1.1: Contributions of the author to the papers combined in this dissertation.

1.4 Structure of this Dissertation

This dissertation consists of three parts. The first part will concentrate on fundamental models (Chapter 2) and data (Chapter 3) on public transport. It contains details about commonly used models. Many concepts are state-of-the-art and not needed by readers familiar with public transport planning and modeling. Readers unfamiliar with some of those aspects will receive the necessary information for understanding basic concepts with explanations and illustrations. This section will also contain the specifications about data available to researches and the general public.

The second part will focus on multiple problems arising during delay management. One central challenge here is the decision, whether a train shall wait for delayed passengers coming from another train or depart on time. We present a framework assisting dispatchers of the Deutsche Bahn undertaking these decisions (Chapter 4). By creating this framework, we generated multiple new possibilities for answering open questions about improving dispatching strategies/policies. We investigate finding the best time when to decide and reroute passengers (Chapter 5), studying sensitivity to noise in the passenger flow (Chapter 6) as well as handling additional conflicts resulting from dispatching (Chapter 7). For each topic, we create methods investigating and evaluating a detailed set of large scale instances. One example of those instances is the whole network of the *DB Fernverkehr AG* and *DB Regio AG*. Schedules and delays used in most experiments are based on real-world data.

The third part of this dissertation is about making line plans and schedules more robust (Chapter 8), by presenting new methods for the evaluation of robustness. For this reason, the data sets used consist of smaller networks available to a broad spectrum of researchers. Data and methods in this part are not restricted to long-distance train traffic but also applicable to bus and tram networks. To further improve the realistic modeling of passengers, a more refined model of passenger behavior is presented. The latest work (Chapter 9) will use methods from the robustness evaluation in a framework that can optimize the robustness of the timetable of a public transport plan.

Part I

Fundamentals

Modelling Public Transport 2

Don't oversimplify and don't overcomplicate.

— **Joshua Fields Millburn and Ryan Nicodemus** [MN18]
(The Minimalists)

Providing people with public transport is a complicated business that creates several challenges of a different nature. This chapter will give an introduction to a complex web of tasks, that is, public transport planning and executing.

2.1 Fields of Research in Public Transport

The field of public transport research has many specialized disciplines. There are many ways to subdivide the structure and tasks involved in this field. [Bor+18] recommends starting with the classical stages for planning in the railway industry: strategic, tactical and operational planning. Furthermore there are five main phases of the railway network planning process. Figure 2.1 shows these phases and highlights to which of them this work is contributing. Network assessment and infrastructure planning are the first phases during the strategical planning step. Tasks in this phase analyze demand as well planning the design of the network. Building new infrastructure is typically very expensive, and in the case of railway infrastructure, only huge companies and governments can afford to create them. Such projects usually have a very long time-horizon before and execution stage. It is a lengthy process that becomes possible through the detailed understanding of demand and cost created by a particular project.

When a working infrastructure has been built, the planning process for timetables, vehicles and crew schedules begins. The final phase belonging to the operational cluster is traffic management, dealing with delay management, rescheduling and informing affected passengers. Here the set of tasks is quite different and includes online problems. These problems need to be solved within a short timeframe.

While the main focus of the work presented here focuses on disturbance management as well as the evaluation of planned services, it is of importance to understand the surrounding topics as well. For this reason, the next two sections contain important concepts of planning and routing.

2.2 Planning

The timetable is not the only aspect of public transport that has to be planned. There is a lengthy process of creating several other important plans like the crew- and vehicle schedule. This process is typically divided into several steps until passengers can use any service. The

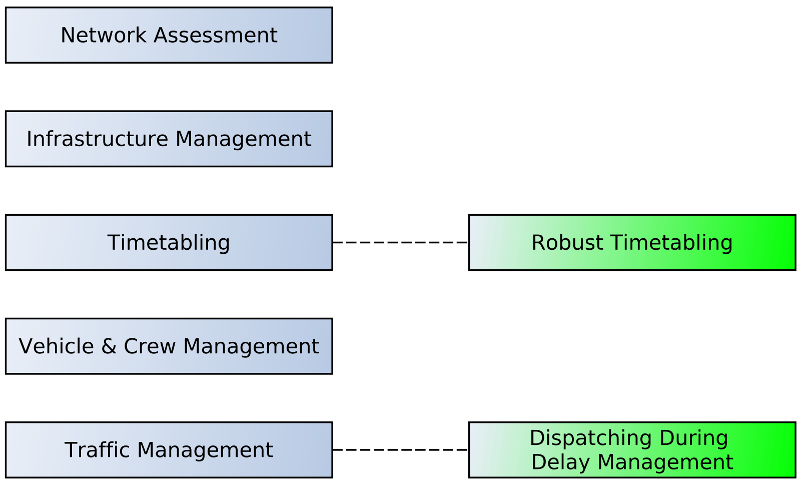

Network Assessment
Infrastructure Management
Timetabling
Robust Timetabling
Vehicle & Crew Management
Traffic Management
Dispatching During Delay Management

Fig. 2.1: Main phases of the railway network planning [Bor+18]. Tasks studied in our work in green.

first step is *network design*. Building streets for cars and buses or tracks for trains require a massive amount of resources. Typically, the governments are involved in planning, approving and funding when the project's main purpose is public transportation. Because of those facts, *network design* has a slightly isolated place in the planning process. Other steps of transport planning have entirely different dimensions in price and planning time. Therefore, in research concerned with planning for public transport, the infrastructure network is often considered as given.

Based on this infrastructure, the first step towards creating services is line planning. Lines are designed to make it easy for the passenger to find and remember an entity associated with a direction of travel. Public transport companies also prefer the structure of services as a line operating periodically.

Based on a set of lines, trips for each vehicle of lines have to be specified. This specification contains information on when vehicles leave and enter a specific stop. The result of all these specifications is called a timetable or schedule. Additionally, there has to be a plan defining what physical vehicle is used to execute the trips specified in the timetable. Planners (and sometimes dispatchers) need to determine which vehicle has to perform a particular trip. This specification is called the vehicle schedule. Analogous to the vehicle schedule, the crew schedule designates the shifts and locations of employees operating the vehicles. Vehicle schedules and crew schedule dominate the operational costs of the public transport company.

Figure 2.2 (left) shows this process described in the last paragraphs as a gradual process. However, if a specific timetable has bad operational costs (compared to the next best solution), a step-by-step optimization leads to sub-optimal solutions. Therefore, researchers try to combine multiple steps into one integrated problem. This, however, results in theoretically hard and practically challenging computational problems. Often their solutions rely on heuristics in several steps [GSS13]. In several parts of this process, especially the evaluation, one needs to ask the

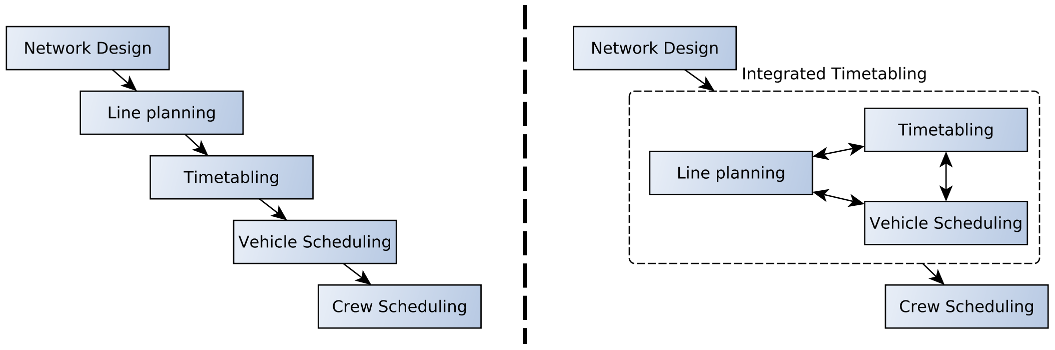


Fig. 2.2: Classical planning steps (left); integrated approach (right)

question of how fast the passengers can reach their desired destination. Any method to achieve this means requires a fast calculation of routes. The next section is focused solely on this topic.

2.3 Routing

Calculation of routes from a starting point to a target station at a specific time is a fundamental problem in many applications concerning public transport. In early street-navigation, the departure time was considered irrelevant to routing. In the current state of the art routing for cars and taxis, the effects on congested roads affect the duration spent on every edge of the network. Calculation of a journey containing bus/train services works differently. From an algorithmic view, the infrastructure becomes irrelevant. Only the schedule defines possible operations that have to be modeled using specific data structures. The survey [Bas+16] contains the most efficient approaches for this task. Dijkstra algorithm can be used on the event-activity network (see the next section) without preprocessing. However, many alternative algorithms outperform the classical approach. When the schedule is aperiodic, goal-directed techniques, contraction hierarchies,the connection scan algorithm [Dib+13], and labeling algorithms are preferable. When the schedule is periodic, RAPTOR (Round-bAsed Public Transit Optimized Router)[DPW12] is currently one of the best algorithms.

Another difference of street routing is the result expected by users. When a route is requested during navigation, the driver is interested in the route with the earliest time of arrival and often one alternative with better fuel efficiency. Passengers using public transport have a different expectation when requesting a connection. Duration of travel, the number of interchanges and the price are three criteria that dominate most passenger's choice of route. This need causes the routing problem to change from a single-criterion routing problem to a bi- or multi-criteria routing problem. While the single-criterion routing problem is easy from a computational point of view, the bi-objective shortest path problem already is NP-hard [Ser87]. The presence of preference and flexibility of passengers transformer this problem into one where a set of solutions

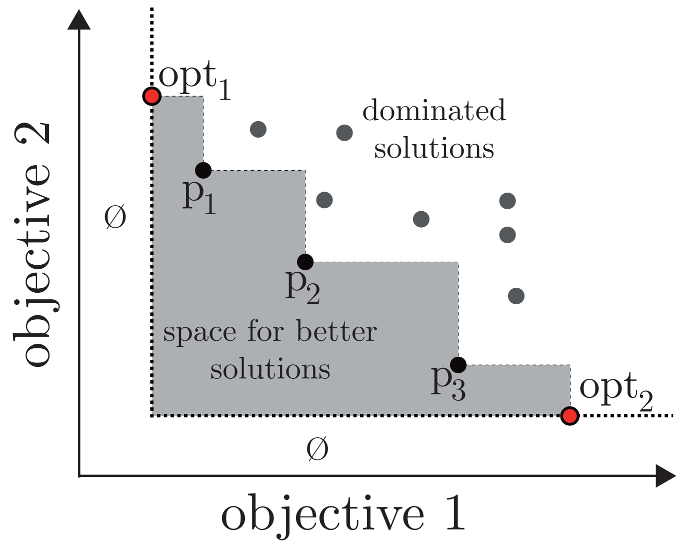


Fig. 2.3: The solution space for two objective functions, supposing not every solution has been found. $opt_{1/2}$ are the optimal solution for one of the two objectives. $opt_{1/2}$ are Pareto-optimal and p_1, p_2 and p_3 are the currently discovered non-dominated solutions. New Pareto-optimal solutions can only be discovered in the grey area.

called Pareto-optimal solutions presents valid options. Figure 2.3 illustrates this principle with two dimensions.

A solution is called dominated, if there is some other solution which is better or equal in all components and strictly better in at least one. Every solution that is non-dominated belongs to the Pareto set. The main approaches for bi-objective shortest path problems are enumerative approaches such as label correcting [BSS89; GM01] and label setting [TC92] or ranking methods [Mar84; RE09]. These methods are essential whenever passengers need to be informed about appropriate routes to reach their destination.

The generalization of this setting is the multicriteria problem. Even for the two-criteria case, the set of Pareto-optimal solutions can be exponentially large, but since common optimization criteria are positively correlated [MHW01] many algorithms can compute solutions fast. The Multicriteria Label-Setting algorithm [Pyr+08] is an optimized extension of the bi-criteria search in a time-expanded model to incorporate minimum change times [MHS07]. However, extended versions of the Connection Scan Algorithm (CSA)[Dib+13] and the multi-criteria version of RAPTOR [DPW12] are faster and require no preprocessing [Bas+16].

In fields of research concerned with simulating passenger flows, having multiple choices for a group may not be necessary. Every passenger decides on one route that maximizes his/her utility expressed as a weighted sum of criteria. In the next section, we will explain how the modeling of a public transport system works on different levels. Creating an abstract model of processes in public transport is essential. Programs/algorithms can only solve problems that are adequately formalized.

2.4 Network Models

Working with challenges produced by disruptions creates a need for modeling the processes in public transport in a way efficient algorithms can use. While in its basic form, a schedule is only a table containing data, there are several ways to represent this data as a graph structure. These graph structures have the advantage of representing dependencies between events and locations the raw data does not provide. This section contains several formal definitions, which we use throughout this work.

There are different ways for modeling processes in public transport using different levels of detail. As an example, we will present several ways of modeling a timetable-based network for trains in this section. The first and simplest model (A) represents stations connected by trains. Given a set of stations S a graph $G = (S, E)$ with $E \subset S \times S$ can represent connections between stations. This level of detail is not sufficient for planners of delay management. Information about a network edge should at least contain some information about distance or minimal travel time t_{min} to be of any practical use. These considerations lead to a second model (B). For most of the work done during our papers, the definition of a *network edge* is:

$$e = (s_{source}, s_{destination}, t_{min}) \quad , s_{source}, s_{destination} \in S, t_{min} \in \mathbb{Q}^+ \tag{2.1}$$

Two things about this definition make a significant difference. The minimal travel time makes a delay propagation possible. Another quality of this definition is that there can be multiple edges between stations. This design reflects the underlying infrastructure more closely. We use this model for calculating lower bounds for connections as well as delay distributions for specific segments between stations.

For planners and infrastructure managers, this level of detail is not enough. For work in those fields, every track needs additional information. Typically their set of nodes is a combination of points representing tracks (with associated platforms) and points representing network switches. Information about the tracks as edges between these points contain the maximum driving speeds on these segments. This microscopic modeling (C) is required for an accurate track allocation that satisfies all security concerns. This level of detail, however, is not used by route planning or other simpler tasks. Figure 2.4 illustrates the three levels of detail covered in this section.

In the next section the core component for modelling a timetable in practice is presented: the **E**vent **A**ctivity **N**etwork (EAN).

2.4.1 Event Activity Networks

The last approach of modeling a network (D) now containing data about the actual schedule is the EAN. It is commonly used for schedule-driven traffic in public transport and often formalized

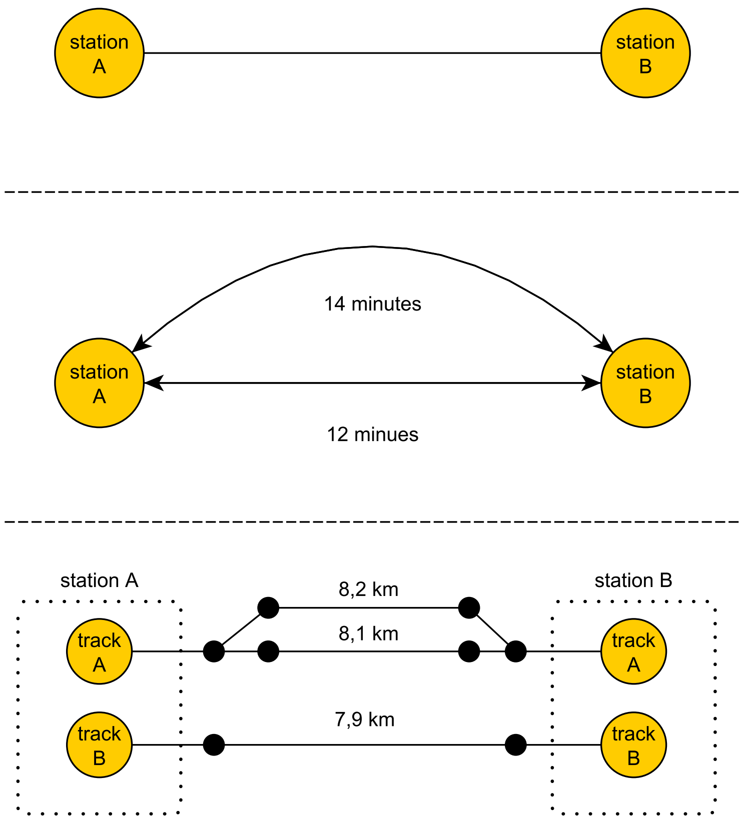


Fig. 2.4: Levels of detail of modeling infrastructure networks. (Top) An undirected network connecting stations (A). (Center) Directed network mirroring minimal travel times on infrastructure. (Bottom) Specialized model necessary for track assignment.

as $N = (V, A)$. V is a set of event vertices, and A is a set of (directed) activity arcs. This network is sometimes also called a time-expanded network [Pyr+08].

Event vertices $v \in V$ usually are either of type arrival or departure event. Events and activities are typically associated with vehicles and infrastructure. Edges can have multiple types and are associated with either train- or passenger activities. The activities associated with trains are driving and dwelling. There can also be additional arcs for dependencies modeling the shunting of a train to its next departure node. Train activities and dependency arcs are crucial for delay propagation/prediction in online systems. Other types of activities are associated with passengers. The most important of these arcs are the transfer arcs. It describes the possibility of an interchange between two trains. When dispatchers or planners specify that an interchange has such a high priority that the connecting train has to wait for passengers or staff it can also be called a dependency (arc). Figure 2.5 shows an event event activity network with four trains. There are also several additions to this model. The most important of those additions is the representation of a special kind of transfer arcs connecting different physical stops. These arcs are often called footpaths. In contrast to transfer arcs connecting arrival and departure events

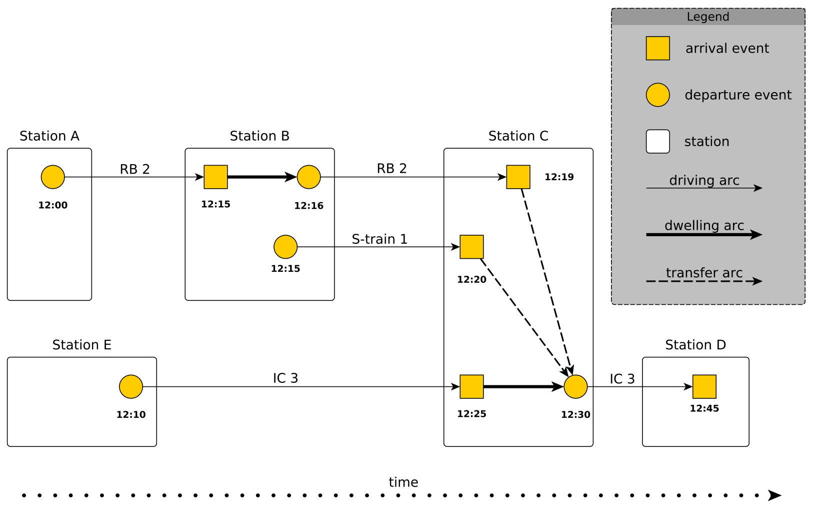


Fig. 2.5: Illustration of an event-activity-network (D) with three trains and five stations.

directly, those arcs are time-independent, creating additional problems during routing. Several paper are focused on the effects of those arcs and practical benefits of restricting or allowing footpaths of defined lengths [Bau+19].

Depending on the available data, events and activities have several attributes. The basic attributes for an event are its type, its associated station, and the planned time from the schedule. Activities do not have to contain many attributes except their type. However, for delay propagation, the minimal time of execution of this activity is commonly present. Event-activity-networks, as a class of graphs, feature certain properties. Activities are only allowed between events of a different type. Transfer arcs, for example, are only allowed to connect an arrival with a departure. While these rules do not forbid the existence of cycles in the network, in practice, any non-dwelling activity has a minimal execution time greater than zero. A a result, event-activity networks belong to the class of directed acyclic graphs in graph theory.

As mentioned earlier, this network is also called a time-expanded network. An alternative to this model is the time-dependent model, where nodes represent stations. In this context, driving-arcs

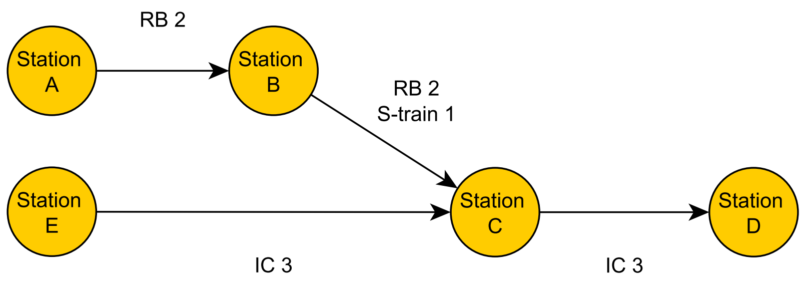


Fig. 2.6: Illustration of time-dependent graph with three trains and five stations.

are called *time-dependent link delays* describing the delay of an entity wanting to use this arc at a certain time [OR90]. This model is illustrated in Figure 2.6. It has several advantages over the time-expended model. The graph has significantly fewer vertices and arcs grouping events into functions where they also can be accessed in a cache-friendly manner. Algorithms can make use of this fact and improve request times for s,t-queries [Pyr+08]. The drawback of this model is the more complicated handling of data in an online setting. Each arc comes with a table specifying the sevices of the activity. It becomes difficult to detect invalid routes used by passengers during delay propagation. Because our work serves precisely this purpose, we use the time-expended model (EAN). Populating these networks with passengers is another challenge. The next section will focus on modeling passengers and their integration into the network.

2.5 Modeling Passengers

In this section, we include passengers in the model. There are three major topics to be covered concerning passengers. The first aspect is the modeling of passenger demand for public transport. After this part, we present ways to model passengers as part of the software framework. The final section will focus on passenger behavior.

2.5.1 Modeling Passenger Demand

The passenger demand is one of the most critical aspects of public transport planning. In the planning of periodic timetables, the expected passenger demand is crucial for choosing frequencies and capacities of used vehicles. There are two common ways to specify demand for planning or routing purposes. The first one is the origin-destination matrix, which is typical for planners. An origin-destination matrix (OD-matrix) is an $n \times n$ matrix for a network of n stations. An element a_{ij} denotes the number of passengers demanding to get from i to j. This matrix is only supposed to represent the demand for a certain frame of time. An example of this is the demand for one year, or every Friday from 6:00 to 6:30. Time is an essential factor for demand. A different network can have different daily load curve for one day. Figure 2.7 illustrates the difference in utilization. In this figure, the long-distance network of the Deutsche Bahn has an entirely different utilization compared to the expected utilization of a bus network used by planners from Stuttgart [Fri+17a].

OD-matrices are usually created using a combination of aggregated sold ticket data and estimations. Depending on the digital footprint of the customers, the quality of this data can vary. Airlines, for example, know the exact number of passengers, while rail and bus companies selling annual passes know less. Planners often interpret historical demand as the demand of passengers willing to use their services during those past conditions. There can also be additional demand from people currently using other forms of transportation.

While the origin-destination matrix is a common way of modeling demand, more detailed representations exist. One way is to store every single sold ticket or connection request made by passengers. A request/ticket typically contains the number of passengers, origin, destination,

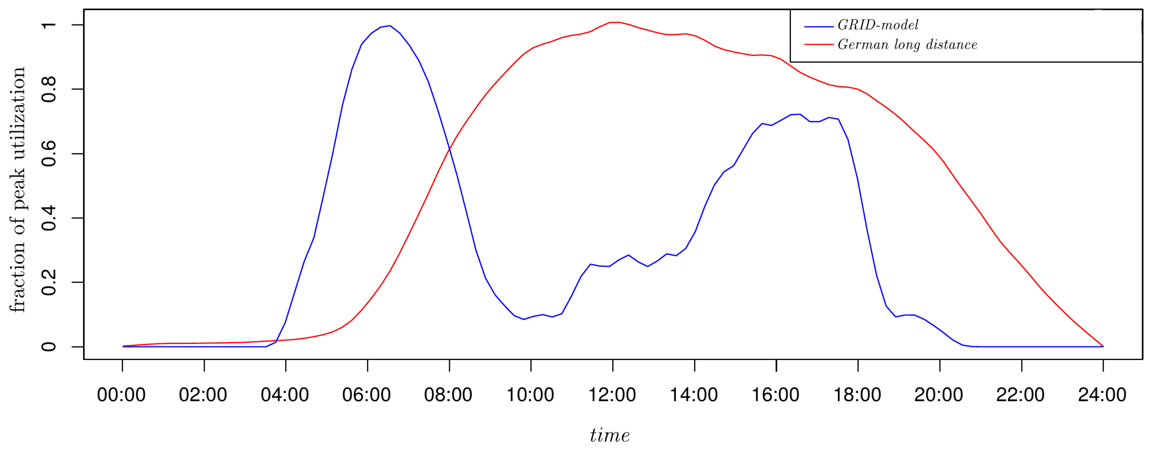


Fig. 2.7: The graph shows the fraction of the peak utilization compared to the daily maximum of two completely different networks. The blue curve represents the utilization using the passenger-demand of the GRID-model network [Fri+17a]. The red curve is the utilization in German long distance railway traffic provided by the Deutsche Bahn.

and the requested departure time. In some cases, this data can even be more detailed than this containing information about preferences, like the preference for more extended stays during interchanges, the need for wheelchair-accessible platforms or trains for transporting bicycles.

During the simulation of the public transport system, we need specific connection requests. Therefore, connection requests are generated from an origin-destination matrix using a heuristic. In our work [Lem+14], for example, Deutsche Bahn provided us with data where passengers were already routed. They generated these connections from their knowledge about yearly demand. An example from this dataset specified that 0.28 passengers want to use a specific combination of trains to get from Halle(Saale) at 07:09 and arriving in Bremerhaven at 12:14. This fractional nature of estimated demand will be the next crucial point in this section about the modeling of passengers.

2.5.2 Passenger Representation

There are multiple ways to model passengers in public transport. In this section, the most common of these ways will be explained, highlighting the strengths and weaknesses of the models.

Fractions and Groups

While, in reality, the number of passengers traveling from origin to their destination is an integer, scientists often model passengers as fractional numbers. Even though this seems an unintuitive choice, there are two good reasons for this practice. The first reason is that input using an origin-destination matrix can sometimes lead to fractional numbers. This can happen when yearly demand is scaled down to daily demand. To avoid this practice, one can round the number of passengers to integers [Ber+11a]. We are using randomization schemes to achieve this purpose. The other reason for using a fractional amount of passengers is the usage of a model based on probabilities. If it is equally likely for a passenger to take one of two routes, we can assign half a passenger to either path. This practice can be useful because of specific computational properties. Solving problems via linear programming, for example, is generally much easier than integer linear programming. To the best of our knowledge, however, connection dispatchers do not work with fractional passengers.

Another important aspect concerning the number of passengers arises when dealing with groups of multiple passengers. A typical example from practice is a family of four people traveling together. We collect a set of passengers with the same origin, destination, and time of planned departure as a group. This practice can have two main reasons. The first reason is the computational benefit, having no extra work dealing with identical groups. The other reason is due to the realistic modeling of social units. The family mentioned in the example will never split up to have an optimal traveling time. Groups aggregated for computational benefits alone, however, can split up if, for example, the capacity is only allowing a fraction of them to board.

Aggregation and Storage of Groups

There are many advantages and disadvantages of the explicit storage of each passenger on his route. One disadvantage is the memory needed to store those passengers. We can examine this using the following constants:

c_{base}	The cost for storing basic information like source, target, size and time.
$c_{structure}$	The overhead of storing a list, or array.
$c_{pointer}$	The cost of storing a pointer. Usually 64 bit.
$\overline{n_{edges}}$	The average number of activities used by a passenger.
$\overline{n_{vehicles}}$	The average number of vehicles used by a passenger (often ≤ 2.0).

If there are k groups of passengers, at least kc_{base} storage is needed. At the start of each simulation, every group then has to decide on a specific route. After making this decision, the question arises of how to store the specific route. There are two ways of achieving this.

The first approach (A) only stores minimal information about each passenger route. Only events where there are boarding and de-boarding are needed. The number of these events are typically small, with an average of fewer than four events in most networks. Therefore, the

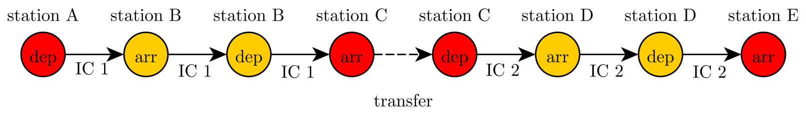


Fig. 2.8: A route of a group modeled in an event activity network. The first approach (A) only stores the red-colored events. The second approach (B) needs to store all 7 activity-edges shown.

basic storage of routes is relatively cheap. When implementing this, it is only necessary to store a list of pointers referencing each event. We achieve a total memory consumption of only $k \cdot (c_{base} + c_{structure} + 2c_{pointer} \cdot \overline{n_{vehicles}})$. This can be simplified as $k \cdot (c_{base} + c_{structure} + 256bit)$. These types of routes are useful for answering basic questions during a simulation. We can derive the duration of travel, duration of waiting, driving times, and modes of travel from this representation. Figure 2.8 illustrates a route where the first approach uses a minimal number of events.

The second approach (B) is storing a pointer of every activity a passenger is using. This approach will increase memory consumption significantly. We now need to store $k \cdot (c_{base} + c_{structure} + c_{pointer} \cdot \overline{n_{edges}})$. In rare cases, when passengers only need one or two edges, this method is competitive in terms of memory consumption. Depending on the type of public transport, this method often may need 2 to 20 times more memory. At first glance, there are not many advantages to the second approach. When the experienced quality of a journey is evaluated based on the occupation of each driving arc, the evaluation is slightly faster. The second approach has benefits in online simulation and passenger information.

Linking of Groups to the Event-Activty-Network

Combining aspects of delay management and passenger simulation results in requests coming from both directions. Passengers want routing information and train status updates for better transparency. Operators want to know where passengers in a specific train wish to go, making sure they succeed. Both worlds need to be linked to guarantee fast response times for those types of requests.

To illustrate this need, let us suppose there are k groups of passengers with an average number of $\overline{n_{vehicles}}$ vehicles per group. If a dispatcher wants to know about the planned destinations of passengers traveling on a specific train, we need to scan all groups and their routes. Let us suppose we applied the approach (A) from the last subsection. This scan will take $k \cdot \overline{n_{vehicles}}$ steps to produce the requested groups we will call $k_{ontrain}$. We only have to look at the stored routes. Storing references between graph and groups can reduce the number of steps to the size of the output ($k_{ontrain}$). This section will explain how we can efficiently achieve this.

One approach that drastically reduces the complexity for answering this question is to store the id of the boarding/de-boarding group at the respective event. For the whole graph this costs only $k \cdot (c_{structure} + 2c_{pointer}\overline{n_{vehicles}})$. This results in a doubling of the total cost of applying

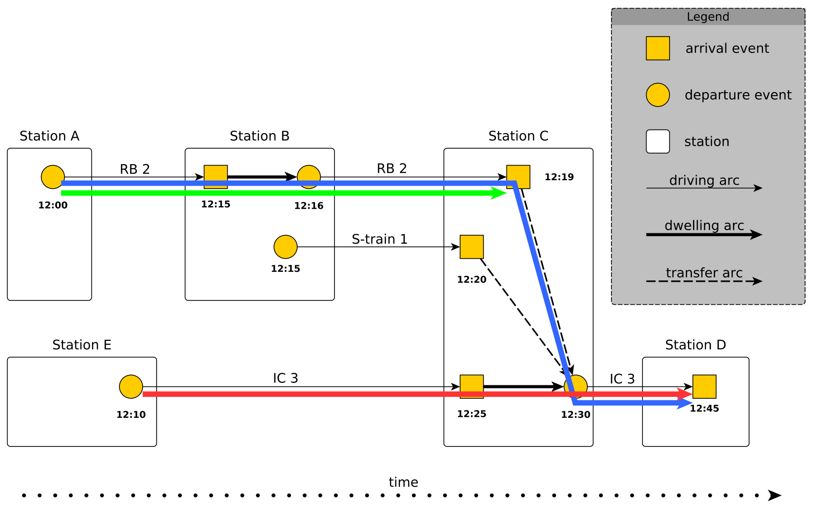


Fig. 2.9: Event activity network with three passenger groups as colored paths [MHR17].

approach (A). This method reduces the complexity of information about one train to a scan of its previous nodes. This practice is mimicking method (A) from the group side to the network side.

A second approach is to have a container on every activity-edge, storing a pointer to every group using it. Here we are also mimicking the technique (B) onto the network. This method also has double the memory consumption of (B) in total but produces the $k_{ontrain}$ groups in $k_{ontrain}$ steps. (This only counts as one atomic operation returning the pointer of the list that already exists). If we use one of both techniques, the graph is an event-activity network enriched with a passenger-flow (see Figure 2.9). In our practical work (Chapters 4-8), we always use approach (B).

2.5.3 Influences on Route Selection

Passengers are individuals by nature and act according to their personal needs and utilities when using public transport. The level of detail modeling passenger behavior used in planning and simulation can vary by the size and nature of the network. Typically the more opportunities available to an individual, the harder the choice for a route. Giving not too many options is also essential, so individuals do not experience stress through opportunity overload [CBG15]. In networks where there are only two lines from the same operator, only a few factors will influence the decision of passengers. In long-distance railway, however, many essential aspects influence the route selection.

Figure 2.10 illustrates the most significant influences on the route selection. Reliability and punctuality can have a significant impact on decisions [Kou+14]. If the service has a reputation of not being reliable, passengers avoid routes with several interchanges.
Measuring the reputation of a public transport company as well as the quality of service is difficult. While unhappy passengers will ask for compensation or write a bad review on social

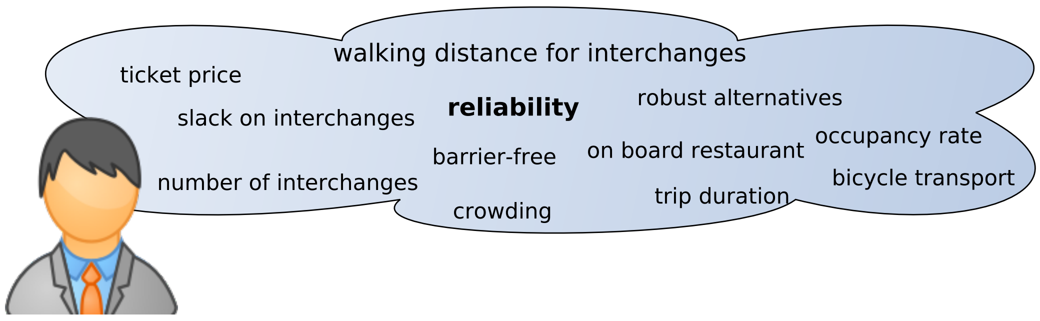


Fig. 2.10: Influences on route selection.

media, positive feedback is quite rare. Companies like Deutsche Bahn use **k**ey **p**erformance **i**ndicators (KPIs) to evaluate their quality of service during daily operation. While punctuality of trains is an important KPI for Deutsche Bahn, experienced delay at the end of each trip becomes more and more important.

The journey of passengers needs to be tracked and updated when delays occur to measure experienced changes and delays correctly. The next section will give an introduction to live-information about passengers and vehicles.

2.6 Including Live-Information

In the information age, many processes like logistics become traceable through technology. Objects and goods often have a digital counterpart informing about their status and whereabouts. The online tracking of trains is essential for apparent reasons. The monitoring of individual passengers is a more delicate topic. Being tracked as an individual can be seen as a violation of one's privacy. From an operator's point of view having access to specific data can, however, result in multiple benefits for the passenger. These benefits will be discussed in Part II after we outlined basic concepts about live information in this section. We begin by presenting information about passengers.

2.6.1 Availability of Live-Information about Passengers

The degree of information public transport companies have is different in many ways. Some public transport companies have many details about their passengers while others do not. Airlines or the FlixBus company know 100% about the number of passengers traveling in their live-system because boarding without confirmed payment for a specific trip is not possible. These companies have full information on their used capacities.

Other public transport service providers offering monthly subscriptions and paper tickets often have to estimate real-time capacities. Some companies equip a certain number of their vehicles with devices that count boarding and de-boarding passengers. This data is then used to update their estimation of passenger-demand of certain connections.

Another technology widely adopted is the use of contactless smart cards. The city of Seoul introduced this way of payment in 1996 [RFID18]. Customers can charge money to a card containing a Radio Frequency Identification (RFID) chip. When boarding a vehicle or entering a station, customers tap their card to an RFID-reader to register, that they are using a service. Many companies require the passenger to tap a second register when leaving to calculate the fare for exact distances. Using information from those cards, providers of public transport usually have close to perfect information about traveling-demand. Ridership prediction of vehicles can be done even when smartcard readers are located inside of stations [van+15]. Modern smartphones make digital payment of public transport more convenient. Commonly used methods let user either buy a digital ticket (and display a verification as QR-code) or let the smartphone use **n**ear-**f**ield **c**ommunication (NFC) to communicate with a smartcard reader. Recent studies show how to generate OD matrices from this data [KAA21]. Efforts to use deep learning approaches for passenger flow prediction in bus transit systems [LC17; Luo+21] have also produced the first results.

2.6.2 Vehicles

Live-information on vehicles is essential for two reasons. On the one hand, the public transport company may want feedback about problems that occur during operation. On the other hand, passengers require information if their journey is not going as they have planned. If a scheduled transfer from one vehicle to another will not be possible, the passenger wants to have a good alternative as fast as possible. Modern companies measure delays either with generated messages about arrival or departure or GPS-units. Some companies have drivers call a delay manager and report their delays via radio signal or cell phone. Vehicles now are communicating their status and can send additional data like when and how long doors have been opened [Sch16].

The Deutsche Bahn uses two ways of digitally monitoring their vehicles. The older approach is measuring whenever a train passes an entry or exiting point of a platform or railway switch. This technique is cost-effective but sometimes produces false information when trains pass well outside their scheduled time. The Deutschen Bahn also created an online monitor where trains can be monitored with their GPS coordinates in real-time [Zei13]. The result of this project was an integration into the DB Navigator app [Chi18].

2.6.3 Impact of Live-Information on Passengers

Smartphone technology has a significant impact on passenger behavior. In the past, passengers had to rely on the information received from staff members. Now there is little to no gap between information known to the passenger and to staff on trains. Even though this is a positive trend, this causes additional problems.

When the information given by train staff is contradictory to what, for example, the app provided by the same company, passengers are confused. Another problem is the management of crowding in trains. When official staff members were the only source of information about the recommendation of route-choice during rush hour crowding could be moderated. Now passengers demand to board a specific train on the edge of its capacity limit. This trend may lead to a situation where train staff and sometimes even the police force every passenger not having a seat reservation to leave the train [FAZ19].

This concludes details about the modeling and accessibility to data in public transport today. The next chapter will go into detail about this data, informing about the current practices in storing and exchanging it.

Data Sources and Common File Formats

3

It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.

— **Sir Arthur Conan Doyle, Sherlock Holmes [Doy87]**

A crucial component for implementing models covered in the last chapter relies on data available are specific file formats. Those formats need to contain all the necessary features for the task. For undertaking the task of simulating a particular problem, data has to go through the process of being imported from some specified file format.

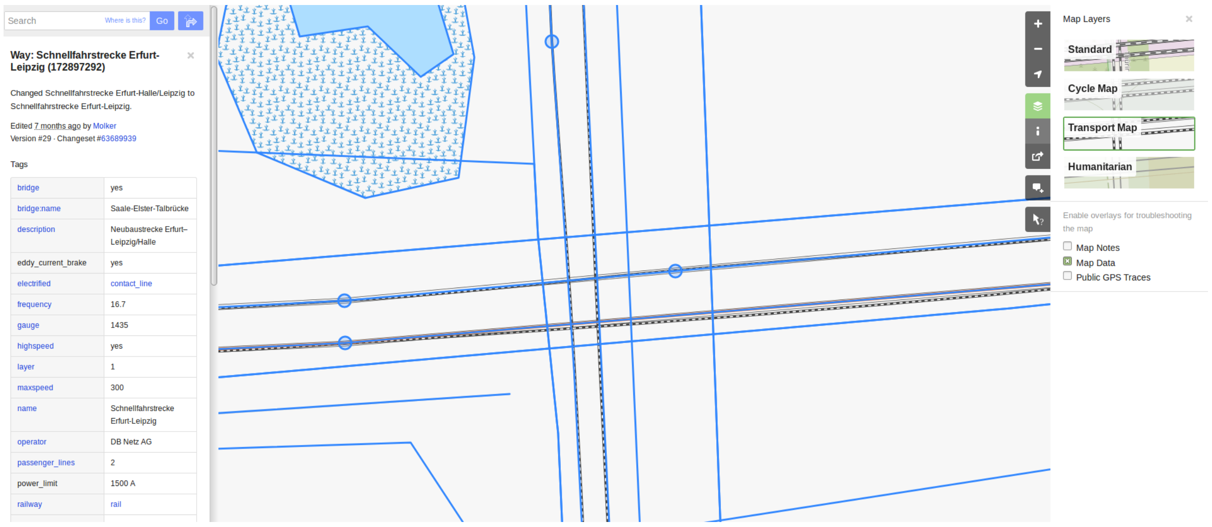
This section will focus on data necessary for modeling and implementing software for solving problems introduced in part II and III. This chapter presents the different types of data. Standard data formats used by researchers and practitioners will be shown in detail.

3.1 Infrastructure

One essential element of public transport data is the underlying infrastructure. The infrastructure is the critical component in planning and has a significant impact on delay management. Every step in public transport planning has to respect the properties of the infrastructure. Decisions in delay management often can only be executed if the underlying infrastructure has enough capacities to support them.

For bus networks, the underlying infrastructure is every street that buses are allowed to use according to traffic laws. Therefore, any of these streets can theoretically be used. In practice, this is more complicated because of several reasons. Many cities have special lanes or gates not accessible to the general public. Due to their size, some busses cannot use narrow roads, sharp corners, bridges, or tunnels with restrictions in size or weight. Planners consider only a small subset of routes for daily operation. They use such a subset of the data containing only relevant information about the infrastructure network. One typical feature of network edges is the duration of travel during regular traffic and peak hours. Overtaking is also not always possible and has to be an extra feature for each road.

In railway networks, the infrastructure network has to contain more detailed information. These networks typically have several tracks connecting platforms of stations or stops. There are railway switches connecting tracks in a directed fashion. They create sections where only one vehicle at a time is allowed to enter due to safety reasons. Sectors also have different speed-limits varying from 10km/h to 300 km/h. These and other characteristics make schedule planning for



Way: Schnellfahrstrecke Erfurt-Leipzig (172897292)

Changed Schnellfahrstrecke Erfurt-Halle/Leipzig to Schnellfahrstrecke Erfurt-Leipzig.

Edited 7 months ago by Molker
Version #29 · Changeset #63689939

Tags

bridge	yes
bridge:name	Saale-Elster-Talbrücke
description	Neubaustrecke Erfurt–Leipzig/Halle
eddy_current_brake	yes
electrified	contact_line
frequency	16.7
gauge	1435
highspeed	yes
layer	1
maxspeed	300
name	Schnellfahrstrecke Erfurt-Leipzig
operator	DB Netz AG
passenger_lines	2
power_limit	1500 A
railway	rail

Map Layers

Standard
Cycle Map
[✓ Transport Map]
Humanitarian

Enable overlays for troubleshooting the map

☐ Map Notes
☒ Map Data
☐ Public GPS Traces

Fig. 3.1: Screenshot(©OpenStreetMap contributors) from the OSM in map data mode. We can identify single tracks and find useful metadata. This data states that the tracks cross via a bridge with an ICE track of the DB Netz AG with a max speed of 300 km/h. Image [Ope19]

railway networks notoriously difficult from an algorithmic point of view. Footpaths are another essential component of the infrastructure. They can either be a base for calculation of transfer times between physical platforms belonging to the same station/stop or longer paths connecting different stations. The first type of footpath is vital for planning in bus networks. The second type is usually not considered in timetabling but used in disruption simulation and multimodal passenger routing.

While schedules have several common-file formats used and exchanged by companies and scientists, there is no consensus concerning infrastructural data. Companies often manage their infrastructural data inside their systems. Outside of their frameworks, the *OpenStreetMap* is one of the best sources for infrastructural data.

3.1.1 OpenStreetMap

The OSM (***Open**Street**Map***) [Ope19] is a well-known project providing free map data to millions of people. The project was created in 2004 and is collaborative. In most areas, the *OpenStreetMap* is more detailed than other maps featuring small hiking paths as well as multiple train tracks.

Practitioners and scientists used the map in numerous applications concerning public transport. In the context of this work, we want to highlight two basic features and usages of *OpenStreetMap*. The first is a detailed annotation of roads or railway tracks of the infrastructure. Figure 3.1 shows many tags about just one railway track.

New lines/services can be planned on the bases of the paths. Planning of new services, however, has its limits. If for example, a new bus route needs to be planned and checked for obstructions. The map can not provide data about what type of bus can bend around what sharp corner or crossing.

Contributors launched a second useful feature in April of 2011. They updated the *OpenStreetMap* with a new set of public transport features. Those features include stops and routes for any regular services in public transport. Clicking on any part of motorway in Germany, for example, will produce up to 20 different long-distance bus routes while railway tracks usually associate with one to eight different services.

Information about the infrastructure is more or less static. Changes are not frequent. Schedules, however, need to be updated at regular intervals.

3.2 Schedules

One of the most fundamental aspects of public transport is that vehicles operate on a schedule. This schedule is often periodic for many practical reasons. Periodic timetables are simpler to operate. Passengers can remember departure times more easily. Public transport companies are sometimes obligated by contract with the local authority to provide a certain number of connections every hour. Before details about different common file-formats are covered in Section 3.2, we state our used definitions:

trip A trip is a service transporting passengers associated with a particular line of a schedule. A trip is executed by a single vehicle.

line A sequence of stops that will be served by trips several times a day. Lines can easily be associated with particular directions and final stops.

vehicle Any vessel for the transportation of passengers specified by the vehicle schedules. One vehicle usually will be deployed for multiple trips a day.

The following subsections presents scheduling data formats used in Part II and III.

3.2.1 HAFAS

Many European railway companies are using the *HaCon Fahrplan-Auskunfts-System* [Hac] as their software for timetable information, most importantly it is used by the Deutsche Bahn AG and other smaller German companies. Raw data of the schedule of German railway traffic is therefore mainly available in the so-called HAFAS format. One dataset in the proprietary HAFAS format [Haf] often provides a yearly schedule of trains under the same administration. The format specifies information about services into units containing a unique stopping pattern at a specific time of day for one line. A reference to a bit-field implies on which day to execute this service. The referenced bit-field has a length of 365 digits. The example below in Figure 3.2 shows a tiny part of the German railway schedule 2017.

Unfortunately, this format has several drawbacks. The most significant portion of those files is white spaces. There is a lot of redundancy concerning the written names of stations. The format contains no information about the infrastructure network at all. There is also no connection between tracks of the station and interchange times. There is no clear divide between a real station and one single track.

```
*Z 10500 AM____                                         % 10500 AM____ (001)
*G BSV 8010205 8010207                                  % 10500 AM____ (002)
*A VE 8010205 8010207 003545                            % 10500 AM____ (003)
*A SV 8010205 8010207                                   % 10500 AM____ (004)
*I FZ 8010205 8010207       2870679                     % 10500 AM____ (005)
*L SEV15 8010205 8010207                                % 10500 AM____ (006)
*R   8010207 8010205 8010207                            % 10500 AM____ (007)
8010205 Leipzig Hbf                 00533               % 10500 AM____ (008)
8012188 Leipzig-Gohlis        00539  00539              % 10500 AM____ (009)
8080840 Leipzig Coppiplatz    00541  00541              % 10500 AM____ (010)
8012193 Leipzig-Moeckern      00543  00543              % 10500 AM____ (011)
8010207 Leipzig-Leutzsch      00552                     % 10500 AM____ (012)

bitfield.101:
003545 <start>0000[...]0000000000000000000000000002040800000000000000000000000<end>
```

Fig. 3.2: Example of a trip in HAFAS format. This example shows the stopping pattern of a regional train that always operates from Leipzig Hbf at 05:33 and ends at Leipzig-Leutzsch at 05:52. The line at the bottom is the corresponding line from the bit-field file written in the hexadecimal numeral system. The trip is only executed three times a year.

3.2.2 General Transit Feed Specification

The General Transit Feed Specification (GTFS) is a standard format for public transport schedules associated with geographic information and was introduced by Google in 2006. The format was initially created by Google to serve the purpose of integrating transit data into *Google Maps*. The format initially named *Google Transit Feed Specification* gained popularity in the United States where there was no standard format for public transit timetables. Developers of transit-related software started using this format immediately. The name of the format was changed replacing the *Google* with *General* in 2009.

```
route_id,service_id,trip_id,trip_headsign,direction_id,shape_id,wheelchair_accessible,bikes_allowed
1001,1001_20181005_20181118_Ke,1001_20181005_Ke_1_0540,"Käpylä",0,1001_20170814_1,1,2
1001,1001_20181005_20181118_Ke,1001_20181005_Ke_1_0550,"Käpylä",0,1001_20170814_1,1,2
1001,1001_20181005_20181118_Ke,1001_20181005_Ke_1_0600,"Käpylä",0,1001_20170814_1,1,2
1001,1001_20181005_20181118_Ke,1001_20181005_Ke_1_0610,"Käpylä",0,1001_20170814_1,1,2
1001,1001_20181005_20181118_Ke,1001_20181005_Ke_1_0620,"Käpylä",0,1001_20170814_1,1,2
1001,1001_20181005_20181118_Ke,1001_20181005_Ke_1_0629,"Käpylä",0,1001_20170814_1,1,2
1001,1001_20181005_20181118_Ke,1001_20181005_Ke_1_0639,"Käpylä",0,1001_20170814_1,1,2
1001,1001_20181005_20181118_Ke,1001_20181005_Ke_1_0649,"Käpylä",0,1001_20170814_1,1,2
1001,1001_20181005_20181118_Ke,1001_20181005_Ke_1_0658,"Käpylä",0,1001_20170814_1,1,2
```

Fig. 3.3: Example of a trip.txt file from a GTFS schedule

In contrast to the HAFAS format, GTFS uses .csv-files and keeps redundancy to a minimum. While the HAFAS format always contains one year, a GTFS-trip operates between a specific start day and an end day with a weekly pattern. Stops are usually entities where passengers board or disembark from a vehicle. The design uses an optional parameter $location_{typ}$ that marks a stop as a station/group containing one or more physical stops of the default type. With this distinction, it is possible to define separate footpaths between tracks of one station and from one station to another. The GTFS format does not contain any information about the

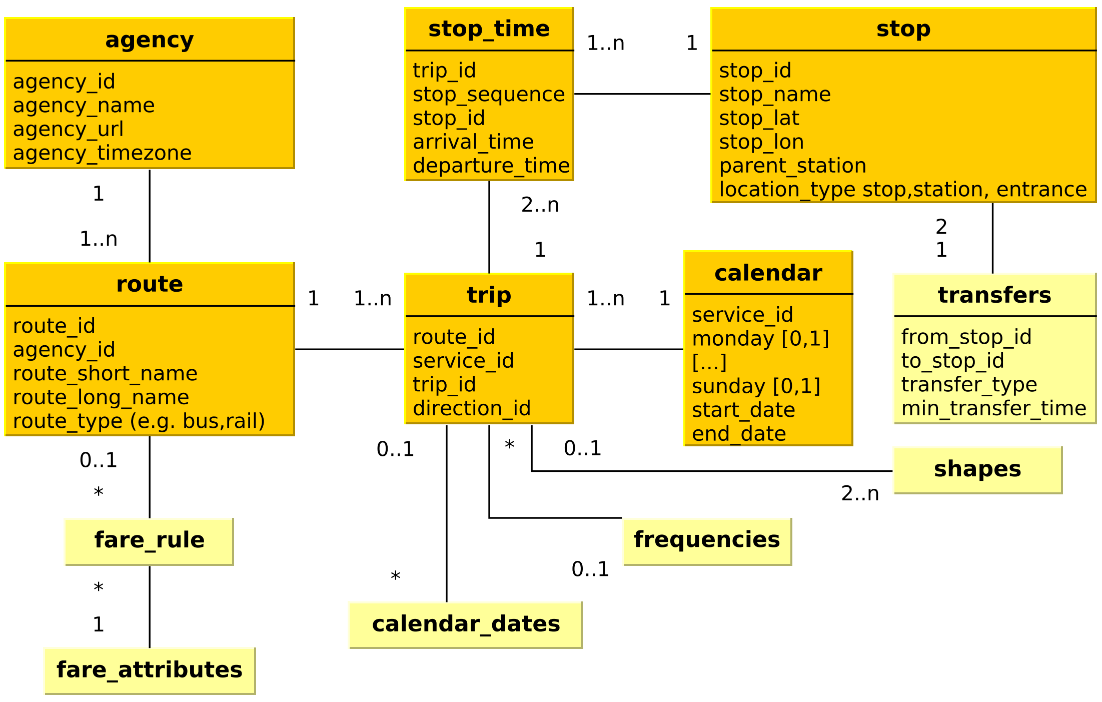


Fig. 3.4: UML of GTFS format. The figure contains only a subset of all attributes. Optional classes are drawn with lighter color

infrastructure. Nevertheless, using the optional *shapes.txt* file providing the geographical location of routes may be used. This fact creates at least a small advantage over the HAFAS-format where there is no such information available. Figure 3.4 shows an UML diagram of entities covered by the GTFS format. In contrast to the proprietary HAFAS format, the GTFS format is still gaining popularity.

3.2.3 LinTim and its Data Format

While the purpose of GTFS and HAFAS is only to encode a schedule, formats belonging to the field of planning have different requirements. Researches from Göttingen created the software toolbox LinTim("Line planning and Timetabling") to calculate each of these steps using mathematical models [GSS13; Lin]. In [SG13] passenger paths were introduced. Using a defined pool of lines, LinTim can output multiple timetables satisfying various constraints considering the number of passengers and included buffer times.

Serving the task of creating a line plan, timetable, and vehicle schedule LinTim produces several outputs filed in the .csv format. The format and structure of those schedules are very similar to GTFS. For our work in Part III, the provided data contained only the schedule for one day. Another difference to GTFS is that edges contain an attribute stating their minimum travel time which is useful when vehicles need to make up lost time.

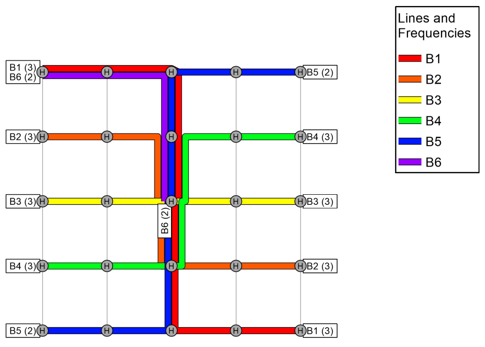
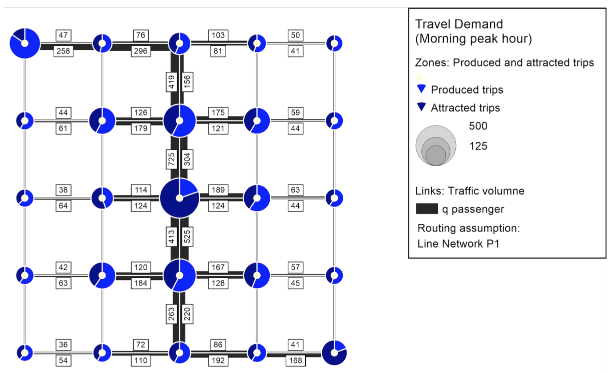


Fig. 3.5: Example of a line plan for the grid network [Fri+17b]



Travel Demand (Morning peak hour)
Zones: Produced and attracted trips
Produced trips
Attracted trips
500
125
Links: Traffic volumne
q passenger
Routing assumption: Line Network P1

Fig. 3.6: Visualization of the demand of the grid network used in Part III [Fri+17b]

One of several artificial networks has been the special focus of the FOR2083 research group. This network is a grid of 25 stations and a specially curated origin-destination matrix for passenger demand. Several line networks were created and analyzed according to speed and robustness with given schedules in Section 8.4. In Figure 3.5 an example for a set of lines is given. The optimization of a line plan and schedule is only possible if the input data contains passenger demand. For this network, a sophisticated demand model was created, as shown in Figure 3.6.

Having covered infrastructure and scheduling data, the only relevant part missing for the simulation of an undisturbed day of traffic in public transport is passenger data.

3.3 Passenger Demand

Exchanging passenger demand data in a standardized format is not a common occurrence. Private companies do most public transport and are therefore not willing to share data that is of the highest value to possible competitors. The scientific community often exchanges passenger demand in form of OD-matrices (see Section 2.5.1) stored in **c**omma **s**eparated **v**alues(.csv) files. Figure 3.7 is an example of such a file.

```
origin-stop-id;target-stop-id;customers
1; 1; 0
1; 2; 21
1; 3; 15
1; 4; 10
1; 5; 8
1; 6; 5
[...]
```

Fig. 3.7: Example of a OD-matrix as .csv

Storing a single connection using comma-separated values is also common. A group can be represented either by one line per group or one line per booked train. The latter can be seen in

```
ID;OriginStopID;DestinationStopID;NumPassengers;Parts;Part;TrainID;Date;BordingTime;DeBoardingTime
2001;8000001;8000832;1;2;1;13;02.05.17 00:00;02.05.17 09:39;02.05.17 10:15
2001;8000001;8000832;1;2;2;;02.05.17 00:00;02.05.17 10:37;02.05.17 10:53
2002;8000001;8003361;1;2;1;15;02.05.17 00:00;02.05.17 11:39;02.05.17 12:15
2002;8000001;8003361;1;2;2;;02.05.17 00:00;02.05.17 12:32;02.05.17 12:37
2003;8000002;8000114;2;2;1;2069;02.05.17 00:00;02.05.17 14:57;02.05.17 16:18
2003;8000002;8000114;2;2;2;;02.05.17 00:00;02.05.17 16:25;02.05.17 16:31
2004;8000002;8000114;2;2;1;2161;02.05.17 00:00;02.05.17 16:57;02.05.17 18:18
2004;8000002;8000114;2;2;2;;02.05.17 00:00;02.05.17 18:39;02.05.17 18:46
2005;8000002;8000231;2;2;1;2063;02.05.17 00:00;02.05.17 08:59;02.05.17 09:25
2005;8000002;8000231;2;2;2;;02.05.17 00:00;02.05.17 09:31;02.05.17 11:21
```

Fig. 3.8: Example of connections as .csv-file

Figure 3.8. Even though the format is called comma-separated, the separator can often be a semicolon, especially in countries which use the comma as decimal separator.

The input data introduced up to this point are sufficient to simulate undisturbed public traffic and to calculate metrics about the passenger's intended journey. Going from this point to realistic simulations of delays requires real-world delay data or an already trained model for generating random delays.

3.4 Delay Data

Most data mentioned up to this point do not suggest that problems and data in public transport belong to the field of big data. This fact changes as soon as delay data is involved. While a periodic schedule for a month may only require a few megabytes in size, the corresponding delay data, including the storage of prediction messages generate gigabytes of data. Deutsche Bahn, for example, produces 29 million individual delay or prediction messages a day on average.

Because of the sizes of those data sets, they are usually not exported or interchanged. Those data often reside in large databases or are exchanged using packages containing small chunks of data. In Figure 3.9 the content of an XML file can be seen. The document contains information about a delayed vehicle (tram).

Concluding the essential data types for the simulation of public transport, there is still the problem of availability. Without individual contracts to public transport companies getting data for future research can be difficult. Fortunately, many online resources are helping with this task. The next section will present some of those resources.

3.5 Availability as Open Data

Open data is the concept that data should be freely available to everyone to use. The scientific community prefers open-source software for several reasons. Public transport used to be a field where except printouts of a schedule, no data was publicly available. Even if specific data has some online-availability, usage and republishing of this data is restricted. There are many arguments on behalf of open data. Scientific and economic improvement flourishes when people

```
<?xml version="1.0"?>
<Paket Version="1.2" SpezVer="1" TOut="20160218235954945" KNr="838468510">
<IstProg>
<Service Id="240328749160" IdZGattung="STR" IdBfEvaNr="732688" IdZeit="20160219005800">
  <ListZug>
<Zug Nr="57303" Gattung="STR" GattungInt="STR" Linie="M13" Name="STR 57303">
  <ListZE>
    <ZE Typ="Start">
      <Bf EvaNr="732688" Name="Revaler Str., Berlin"/>
      <Zeit Soll="20160219005800" Prog="20160219005800" Dispo="NEIN"/>
    </ZE>
    <ZE Typ="An">
      <Bf EvaNr="732712" Name="Libauer Str., Berlin"/>
      <Zeit Soll="20160219010000" Prog="20160219010000" Dispo="NEIN"/>

[...]
```

Fig. 3.9: Delay Data Message Packet used by Deutsche Bahn

have access to the concerned data. On the other hand, passenger specific data can not be distributed freely due to several reasons, such as privacy and company beneficial secrets.

3.5.1 Deutsche Bahn API

The German railway provider Deutsche Bahn created an API for accessing schedule data in 2015. For the use of this API, an API-key has to be requested by the Deutsche Bahn. The API only provides schedule data from long-distance trains. Data gained by this API can easily be converted to GTFS using scripts. In 2016 a GitHub project was created doing exactly this. Anybody having access to an API can create a GTFS feed or defined duration of time using this tool [Db16].

3.5.2 Publicly Available Sources for GTFS

In contrast to the HAFAS format, the GTFS format is still gaining in popularity within the open data community. In September 2021 there were 850 providers of GTFS-Feeds on *OpenMobilityData* [IO21], 22 of which were from Germany. Moreover, since 1.1.2020 the website *gtfs.de* [Bro] provides schedules for most German providers separated into 3 feeds: local transport, regional trains and long-distance trains. The site provides this format for the next seven days of traffic. A website focused on research and providing data for start-up companies is Open Data ÖPNV [DEL]. The site also gives access to a live feed of delay messages.

This availability is a step towards an improvement in scientific research concerning public transport. Third parties can also use this information to improve good sources of information for passengers.

This part presented the necessary modeling techniques and data to work with software in public transportation. The following two parts contain chapters focusing on specific problems. From this point forward, we address research problems in the following form in each chapter.

We give motivation, methods, plan and conduct experiments and evaluate the results. The next chapter will focus on our endeavors to create the PANDA framework.

Part II

Train Disposition

The second part focuses on disruption management, with a strong focus on train dispatching. Delays and train cancellation are a common occurrence in daily operation. The effects of those disruptive forces cause several problems on many levels. The two most prominent issues are that passengers experience delays, and operatives strive to restore operations to the planned schedule. These problems are merely the tip of an iceberg of problems:

- passengers miss their connections
- operators have to decide if a train should wait for delayed passengers
- passengers have to be informed about decisions
- passengers have to be informed about alternatives
- operators have to handle track allocations, which may collide with freight trains
- service employees may enter overtime or even a state where they are no longer allowed to continue working
- operators have to restore the vehicle schedule to prevent problems from interfering into the next day

The work we present seeks to create improvements in the first four of those points. Passengers and operators both may suffer from incomplete or insufficient information about current events or passengers. We explore this part of the problem in great detail in Chapter 6. We condense many of those problems to be solved into one software designed to help both passengers and dispatchers. This software is called PANDA and was officially introduced by us in [Rüc+15]. We begin by explaining the current work of dispatchers in detail. Then we expand on the **P**assenger **A**ware **N**ovel **D**ispatcing **A**ssistance (PANDA; Chapter 4) for supporting decisions of the dispatcher. After establishing methods for simulation and evaluation of scenarios occurring during usage of this framework, we investigate several research questions. These questions include the crucial timing for a dispatching decision and its communication to all parties involved (Chapter 5). As mentioned earlier in this section, we challenged the recommendations made with the help of the framework in terms of their reliability, given inaccurate input data. We achieve this task by performing an extensive evaluation of cases and a detailed analysis of their sensitivity (Chapter 6). The final part of our research into this topic will break the traditional paradigm of deciding one connection at a time and perform coupled decisions (Chapter 7). The added value of this approach will be evaluated using real-world data provided by Deutsche Bahn.

PANDA 4

Often injustice lies in what you aren't doing, not only in what you are doing.

— **Marcus Aurelius [Aur80]**

This part will focus on the job of dispatchers working with the management of connections of passengers. The PANDA software should enable dispatchers to perform better-informed decisions. After a brief introduction explaining the job of dispatcher handling delay scenarios we will introduce our software assisting their work. The PANDA project combines many challenges from real-world software engineering. Here we deal with a large-scale network that has to be dynamically adapted to live data. Computing scenarios to assist dispatchers has to happen online and in real-time. In this project we focus on German long-distance trains.

4.1 Introduction to the Work of Dispatchers

There are different types of dispatchers working on railway administration. Some of them exclusively have to manage the allocation of infrastructure for trains. There are several other important duties for dispatchers described in [Rot+99]. Some of those duties are to answer requests for train movement and track usage. Another task is the communication of reasons for delays as well as estimations for future departures during disruptions [Mur+10]. In this part of this work, we will only focus on dispatchers managing the maintenance of connections between trains. There are two duties of those dispatchers managing delayed vehicles. The first one is deciding if the connecting trains shall delay their departures so passengers in danger of missing it still can board. We refer to this particular decision as to the *waiting decision*. The second duty is communicating their decisions to other members of the staff involved and the passengers. The workplace of a dispatcher typically features several monitors, see Figure 4.1. In most cases, several of those monitors are dedicated to a detailed visualization of tracks and their utilization, as shown in Figure 4.1. Time–distance diagrams are also a commonly used tool for visualization during dispatching. Dispatchers managing connections typically use a visual matrix of arriving and departing trains displaying information about transferring passengers when this information is available. The numbers for those changing passengers are collected by the train staff, resulting in fluctuating rates of accuracy. At the time when we started this project, information for the dispatchers were limited. In addition to digital inputs, a telephone, and the collected information about passengers are the only set of tools a dispatcher managing connections has for his decisions [Ros+11].

A fundamental tool that is missing here is one giving any indication and quantification on the effect of a made decision. One further problem is that information about alternatives is often

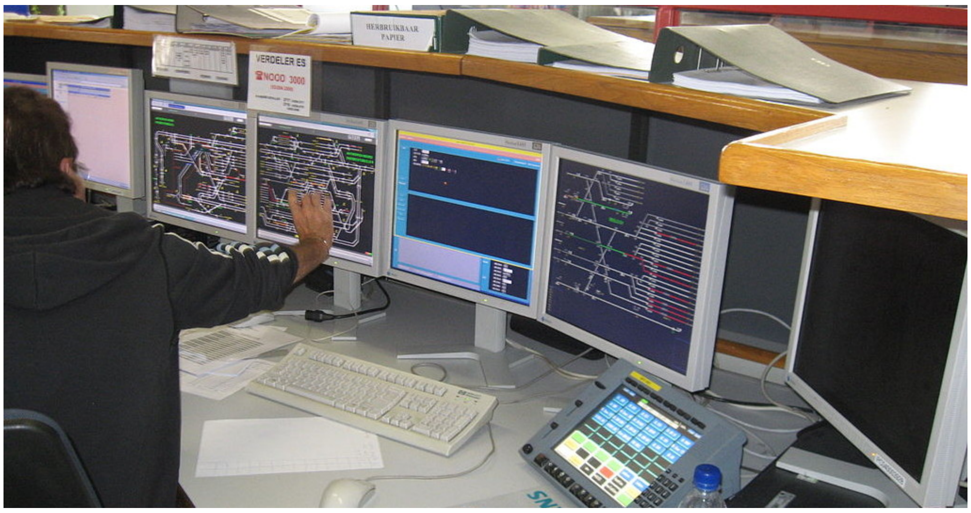


Fig. 4.1: Workplace of dispatcher[Com07]. The hand of the dispatcher points to a monitor displaying track allocation.

direction-specific when they should be destination-specific for any given group of passengers. These missing capabilities of the available tools were the main reason for the development of the PANDA framework, introduced in 2015. Now we give a detailed definition of the problems the framework is working on. Delay management in public transport is about any short-term adaptation of a timetable as a result of delays. As a special case of this category we focus on the:

waiting decisions The decision whether to introduce an intentional departure delay of a vehicle so delayed passengers can still board.

We call a new delay introduced by dispatchers or disruptions *primary delay*. Non-primary delays are the simple continuation of primary delays that persist until the delay is consumed by buffer or the trip ends. The final type of delay is the *knock-on* delay. This delay is created when dependencies of a delayed vehicle force another vehicle to be delayed. A *knock-on* delay can have technical reasons. When waiting time rules are involved *knock-on* can also be caused by strategic reasons.

4.2 Related Work

Section 2.1 describes where in the many fields and steps of public transport planning delay management is located. Usually, the main goal of delay management is to restore the vehicle schedule, or track occupation in a way that services can resume their duties after the original plan. Complex rescheduling may be involved to reach this goal. Our work, however, focuses on minor changes in an already broken schedule to optimize for the earliest arrival of passengers with broken transfers and not fixing the schedule like in most other publications belonging to this topic. The *Handbook of Optimization in the Railway Industry* [Bor+18] has detailed insights into delay propagation and delay management [Dol+18]. There the PANDA framework is also mentioned and its methodology compared with different approaches.

The problem of delay management has been studied using several approaches and analyzing various aspects of the solutions. Researchers in this field often use event-activity networks as a model for the representation of the schedule. [Sch01] converts the decision problem into a mixed-integer program minimizing the total delay of passengers. Waiting decisions of a network with more than 1000 vehicles are optimized within seconds successfully by this method. These techniques, however, rely on periodic services and the basic assumption that all primary delays are known. In practice, ILP formulations suffer from a large decrease in performance in complex networks where the set of possible routes is huge. Continuing this line of work, [Sch07] showed experimental results, reducing delays by up to 17% in a network of similar size. Those results depend on the same conditions for routes and use only five initially known delays. This work also shows the problem is solvable in linear time when no paths of two delayed customers will meet. The general formulation of the delay management however is NP-hard even under simplifying assumptions [Gat+07].

Some formulations of the delay management even take track capacities into account [SS10]. Unfortunately, Deutsche Bahn could not provide our project with detailed infrastructure data, due to their data policy restrictions. [DH11; Dol+10; Sch13] investigated more realistic models using passenger routing on non-periodical networks. Their common assumption is that all future delays become know at some point in time. There are also works focused on the online version of the problem, more similar to the methods used here. Among those are [BS14b; BBOK13; Gat+04] using more realistic assumptions than previous works. Other authors focused on the improvement of modeling and evaluating passenger demands in delay scenarios [Bie06; BS07]. The latter is most similar to PANDA describing the microsimulation of both consequences as one of the 14 strategies suggested. They focus on the number of missed connections and time spent waiting rather than the arrival delay.

The DisKon project [Sch+05] also tried to create a realistic framework for train disposition, but has never been used in daily operation. The PANDA software, in contrast, was available to two disposition centers of the Deutsche Bahn. Before going into detail about the software framework, we need to specify certain terms used:

connection/journey A synonym for a route of a passenger rather than a specific interchange between trains.

interchange/transfer The act of changing from one train to another.

feeder train The train bringing passengers to some transfer.

connecting train The train passengers are changing into during an interchange.

waiting time rules A defined maximum number of minutes the connecting train is supposed to delay its departure, waiting for passengers performing a transfer.

minimal interchange/transfer time The time necessary to physically perform an interchange.

time supplement/buffer The excess time in a (vehicle) schedule that can be used to compensate for delays.

slack The excess (buffer) time available for passengers to perform an interchange.

4.3 Evolution of the Software Framework for this Research

There are different requirements for useful software used by dispatchers. Managing the basic representation of a schedule is the most basic requirement. This means either just representing a static timetable or an adaptable representation like an EAN (see Section 2.4.1). Another fundamental requirement for passenger-oriented software is the ability to find connections from a source to a destination starting at a given point in time. Müller-Hannemann and Schnee presented the MOTIS (**m**ulti-**o**bjective **t**ravel **i**nformation **s**ystem) software at the ATMOS conference in 2004. In 2007 they published an improved version [MHS07] having not only the ability to generate shortest paths but doing a multi-criteria Pareto search (see Section 2.3) returning multiple routes. Adding the ability to insert delay and delay propagation in this software was the basis for dealing with online-problems reaching from an online-connection search to the detection of broken transfers [MHS09]. At this point, the software was able to send messages informing users about their connections. A similar feature was later integrated into the DB-Navigator app and is used by many passengers since.

Up until this point, it was sufficient for the MOTIS software to consist solely of a graph optimized to perform search requests. Delay propagation made it necessary to include a graph representing dependencies between events.

The next big step for this software was the introduction of passengers in the form of an abstract flow representation [Ber+11a]. This addition made it possible to calculate new attributes of a schedule designed for a specific passenger-flow. At this point, the software could calculate the total travel time for all passengers for planned journeys and during real-world delays.

In 2014 marked a new stage in the development of MOTIS and the work with dispatchers. Under the guidance of Matthias Müller-Hannemann from Martin Luther University Halle-Wittenberg and Christoph Blendinger of Deutsche Bahn new goals were set to improve the work of dispatchers. These goals included a new usable prototype capable of a live calculation of decisions and a new GUI for platform-independent usage. Martin Lemnian and the author did the majority of the implementation of the software and GUI. After three years, we reached our set goals and Martin Lemnian left this project. In this period of time we published several papers [Lem+14; Rüc+15; LMHR16] with additional inputs by Dr. Steffen Schüler who has been especially involved in interchange classification. We give more detailed evaluations as well as additional information connecting these papers in the forthcoming chapters.

The changes in the framework being able to achieve the set goals were the following. The passenger-flow representation inside MOTIS was replaced by an explicit one making it possible to observe every passenger [Lem+14]. More about this work will be presented in Chapter 5. We introduced the biggest set of features in 2015 with the development of PANDA, which we explain

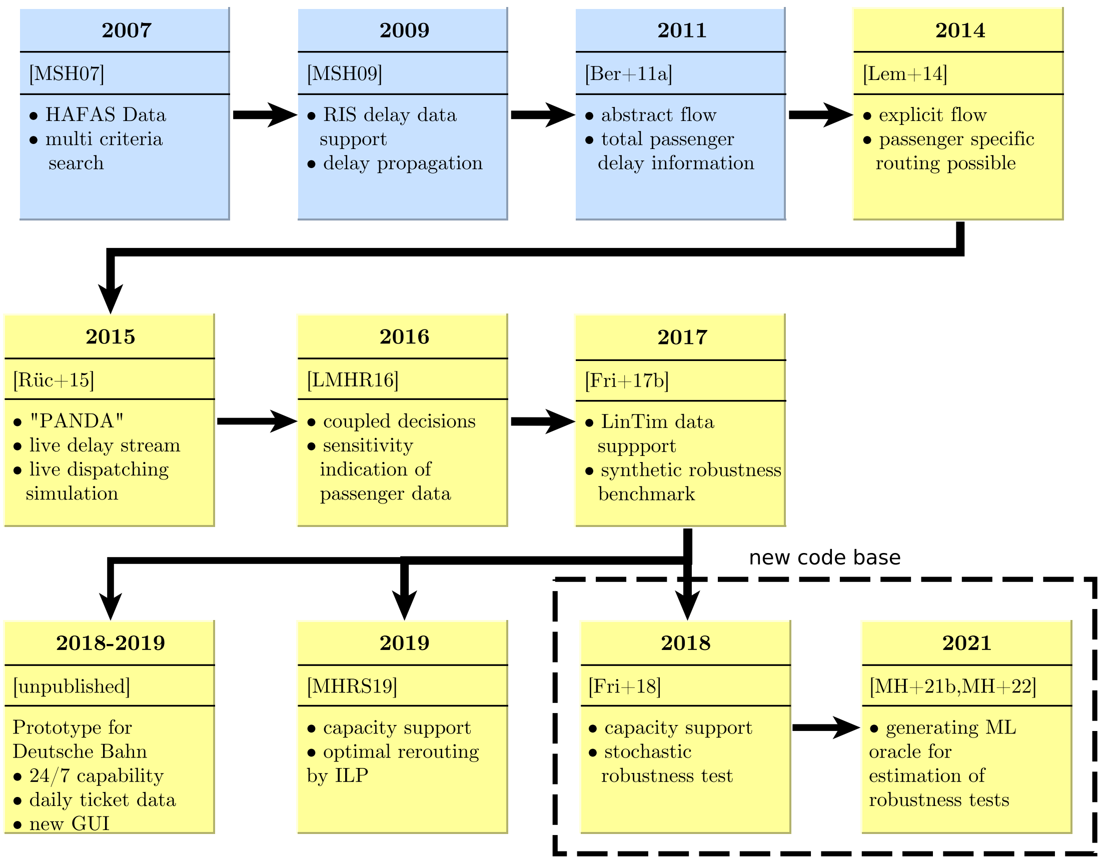


Fig. 4.2: Development and publications using MOTIS and its successor PANDA. Yellow publications feature work with main contributions by the author.

in detail in the next chapter. The features introduced in the development of PANDA were used for research into several questions concerning the quality and benefits of recommendation made by the software. Chapter 6 and 7 will cover these experiments in great detail.

The creation of the DFG funded group FOR2083 [FOR2083] marked the beginning of a new chapter in the research into passenger-friendly optimization in disruption management in public transport. This cluster of seven research groups focused on several different topics in public transport. This collaboration brought new data sets as well as new challenges. Part III contains the work done during this collaboration. In this period most implementations except the creation of timetables was done by the author.

The new challenges made use of the capabilities of the software framework developed up to this point. Changes like the import of data formats used by a wider scientific community were necessary. In 2018 due to performance issues core functionality of the MOTIS framework was transformed into a new code-base optimized for other forms of passenger demands as well as simulations using capacities. Figure 4.2 is a summary of the development of the software back-end as well as the publications.

The next section will cover a detailed analysis of the requirements for software usable for dispatchers.

4.4 Requirements and Software Architecture

Before explaining the architecture of the PANDA framework, it is essential to analyze the requirements of the software. A detailed description of these requirements, as well as some newer features of PANDA, were introduced in [MHR17]. We grouped these requirements into specific categories. The first one is data handling of several different sets of data. This data is a mix of weekly schedules and live data:

- an abstract description of the *general infrastructure* (for example, a table containing all stops, including location, names, minimum transfer times, walking times for footpaths between nearby stops, and the like);
- the train *schedule*, which includes a specification of all trips, their operating days and further attributes of each trip. The *schedule* is typically valid for one year but gets daily updates;
- a list of available *resources* specifying which vehicle is used for which trip, updated daily; in this context, this is of importance for the calculation of free seat capacities;
- *passenger flow* data, containing for each passenger his/her intended travel route;
- *real-time* information about the current status of each trip (timestamps and predictions, and all types of deviations from the schedule like track changes or cancellations) on the current day of operation;
- a huge database of historical *process* data collected from real-time information over previous years; in particular, this database contains time-series of delay data for each event. These time-series can be aggregated to empirical delay distributions, which can be used to infer better predictions of estimated arrival and departure times.

The second group of requirements concerns the capabilities of the designed software to be applicable in practice:

- *Massive updates.* Transit operators provide a live-stream of messages containing real-time information about the current status of trains and vehicles. They provide realized arrival and departure times of reached stations as well as predicted times for future events. Cancellations, changed routes, and several kinds of service alerts have to be reported. This task implies that a timetable information system, as well as a dispatching system, must support efficient updates of its internal data structures. On a typical day of operation in Germany, about 29 million individual messages have to be parsed and handled. The bulk of these messages concerns updates of some timestamp, i.e., they tell that the predicted timestamp for some specific event changes by x minutes.

- *Scalability.* A timetable information system must be highly scalable to a large number of users. In peak hours, several thousand users send queries. For example, HaCon GmbH, the market leader in Europe for timetable information systems, reports that their apps and web-based solutions receive over 100 million requests from passengers each day [Kan+11]. A dispatching system must support multiple users as well since a few dozens of dispatchers want to run their simulations in parallel.

- *Real-time computation.* Nowadays, the user of a timetable information system expects to receive an answer to his or her timetable query instantly. Likewise, the simulation of a waiting/non-waiting decision should not exceed a few seconds. Since one such simulation may require as a subroutine many shortest path queries, this demand limits the time for one shortest path query to a few milliseconds.

- *24/7 availability.* In practice, timetable information systems, as well as dispatching systems, have to be available without interruption 24 hours per day, seven days per week.

There are also practical requirements specific to dispatchers for comparing information to their established environments. We will now give a detailed account of the changes made to the system since [Lem+14] to meet all requirements.

We made several changes concerning data import and interface to make live dispatching assistance possible. The MOTIS software before this work was not optimized for fast updates and live simulation. Schedule and passenger-information had to be received and repaired before the beginning of a day of operation from servers of Deutsche Bahn. The workflow for importing live delay messages from the "**R**eisenden-**I**nformations-**S**ystem" RIS-Server had to be improved. New messages arrive every 30s. The system pushes them to the designated folder via FTP-server. These messages contain files needing to be parsed by a script. After this is accomplished, they are written to a database before being introduced in the software. Figure 4.3 illustrates the architecture of the framework. We optimized the code to improve the latency when users make requests. The most challenging query (simulating a delay scenario) did not exceed 120 seconds. All other requests take only a few milliseconds, except when the software is updating the graph for a duration of three seconds every minute.

One of the biggest challenges was creating a new graphical user interface in the browser that can seamlessly communicate with the algorithmic core. We set up a *ruby on rails* server for this task. Rails is connected via Unix-socket to the algorithmic core and via HTTPS to the internet browser of dispatchers.

These details about the framework conclude out description of the architectural side of the PANDA software. The next section will explain the workflow and introducing the first version of the user interface.

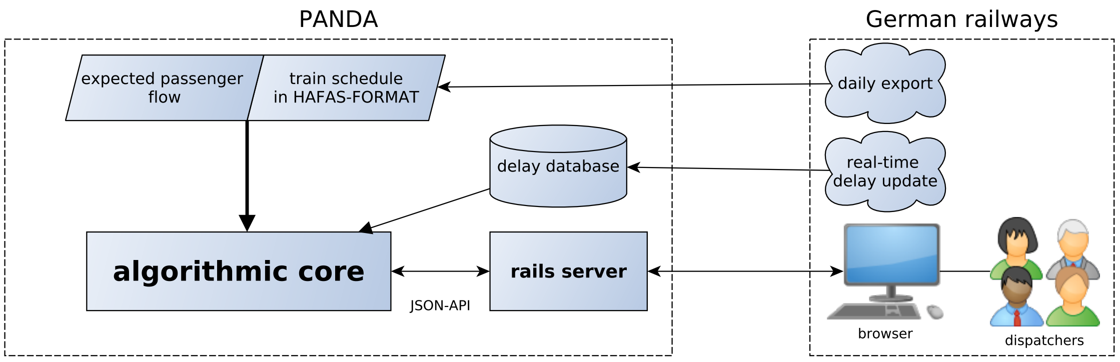


Fig. 4.3: Sketch of PANDA's system architecture and its interface

4.5 Workflow and Interface

We designed the interface in such a way that dispatchers can simulate decisions. This simulation should not make persistent changes to the internal graph representation. The process of how a dispatcher is supposed to work with the software is sketched in Figure 4.4. After receiving updates from Deutsche Bahn delay system, PANDA classifies the transfers into a specific category according to the demand for action. Some of those categories trigger a visual indicator for dispatchers, informing them about necessary decisions. The dispatcher then simulates the departure delay and communicates his/her decision into the systems hosted by Deutsche Bahn. After internal servers of Deutsche Bahn process a decision, their server will communicate delays of trains back to our database.

After presenting the first prototype of PANDA, the Deutsche Bahn decided to schedule a two-week field test with the dispatchers. Dispatchers should be able to use the software running on their usual hardware. Because of the high-security standards with external software, using a browser-based web application was the only feasible option for implementing PANDA. Several map-based views for the dispatcher were introduced using a color-scheme for ease of orientation on the criticality of the transfer. This scheme corresponds with the classification that will be explained in detail in Chapter 5.3. The red color indicates that an interchange can not be kept (or only be kept with an unrealistic waiting time). The orange color is for the *critical* cases where a decision about a potential delay of 4-10 minutes should be made. The yellow color indicates a transfer should be dispatched due to the waiting time rules. We later replaced several designs (for example Figure 4.5) with search based variants. Dispatchers preferred this view because they often know which train is worth a dispatching simulation.

One of the essential features of the early prototype of PANDA was the *Evaluation View*, a comprehensive summary of affected groups and key figures of a decision. This summary (center right-hand side Figure 4.6) showed a direct comparison between the most important effects of a choice. These effects include the number of passengers with minimal delay and the number of passengers being more than one hour late at their destination. The table at the bottom right

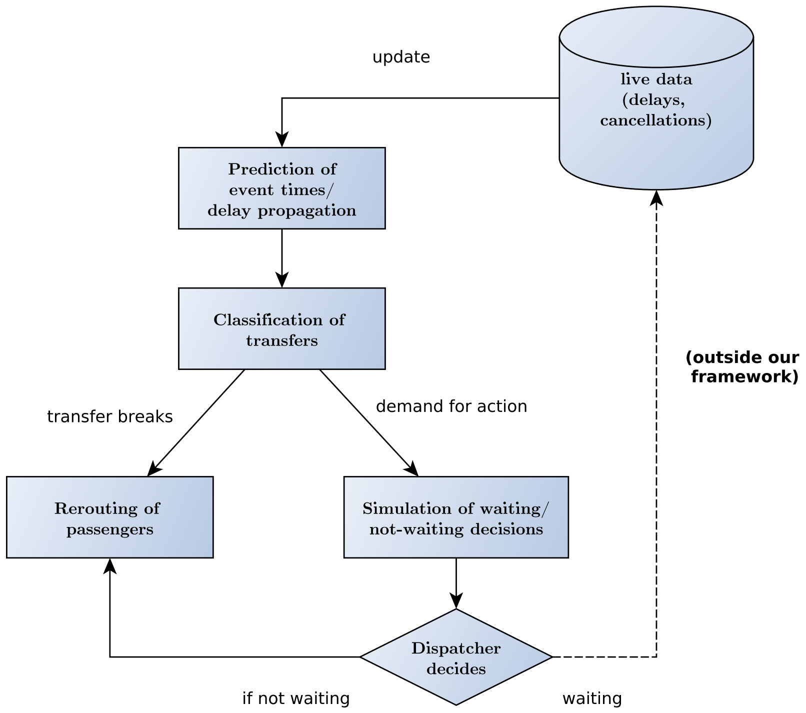

update
live data (delays, cancellations)
Prediction of event times/ delay propagation
Classification of transfers
(outside our framework)
transfer breaks
demand for action
Rerouting of passengers
Simulation of waiting/ not-waiting decisions
Dispatcher decides
if not waiting
waiting

Fig. 4.4: Workflow of a dispatcher using PANDA from delay to decision [MHR17]

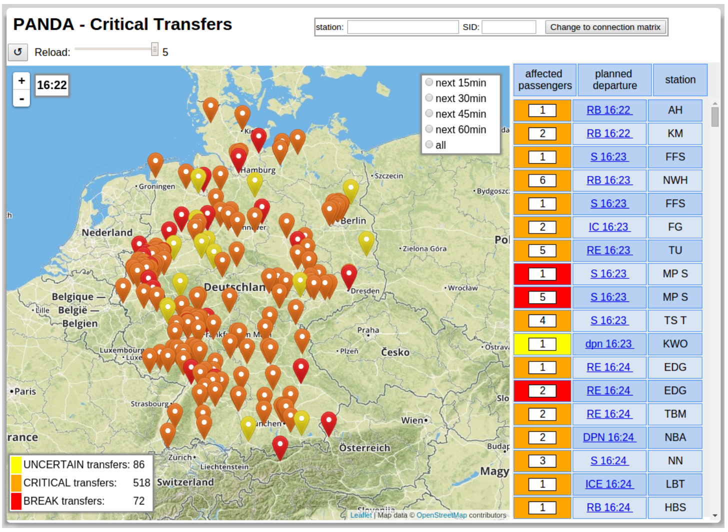
corner included every passenger grouped by common destination and route. The color of the groups matches their colored path on the map.

Hovering with the mouse over the produced delay minutes of a specific group resulted in displayed information about the used route in the form of used trains and their delays. This feature was received very positively by dispatchers, interested in the quick reference to the best alternatives for passengers at this point [Rüc+15].

4.6 Algorithmic Core

This section will explain the fundamental sequence of processes used by every task given to PANDA. The flow-chart Figure 4.7 is a visualization of those fundamental processes.

First, the program builds an EAN from the schedule. At the end of this process, the EAN is built in the same fashion as explained in Section 2.4.1 and visualized in Figure 2.5.This EAN can have additional information provided by the schedule. If there is a concept for lines and a vehicle schedule linking trips, this is also featured. Of special importance is the minimum execution time of driving and travel arcs and the presence of dependency arcs. These arcs can be the result of the turnaround edges of the vehicle schedule or the specification of waiting time rules. For example, this could mean that the schedule has a rule that every departing vehicle of type "IC" has to wait for delayed vehicles of type "ICE" for a maximum of 5 minutes if passengers are using this connection. The presence of such a rule will trigger the explicit construction of



PANDA - Critical Transfers

station: SID: Change to connection matrix

Reload: 5

next 15min
next 30min
next 45min
next 60min
all

affected passengers	planned departure	station
🟧 1	RB 16:22	AH
🟧 2	RB 16:22	KM
🟧 1	S 16:23	FFS
🟧 6	RB 16:23	NWH
🟧 1	S 16:23	FFS
🟧 2	IC 16:23	FG
🟧 5	RE 16:23	TU
🟥 1	S 16:23	MP S
🟥 5	S 16:23	MP S
🟧 4	S 16:23	TS T
🟨 1	dpn 16:23	KWO
🟧 1	RE 16:24	EDG
🟥 2	RE 16:24	EDG
🟧 2	RE 16:24	TBM
🟧 2	DPN 16:24	NBA
🟧 3	S 16:24	NN
🟧 1	ICE 16:24	LBT
🟧 1	RB 16:24	HBS

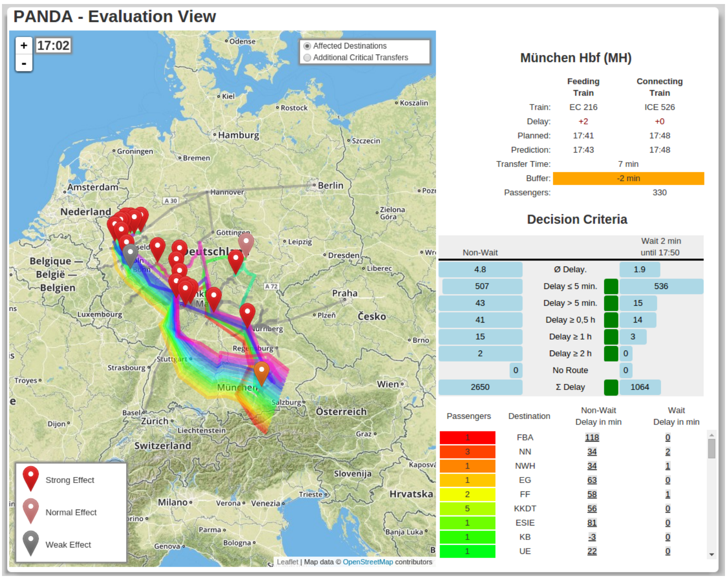
🟨	UNCERTAIN transfers:	86
🟧	CRITICAL transfers:	518
🟥	BREAK transfers:	72

Fig. 4.5: Basic view with summary of critical transfers of german trains [Rüc+15]

dependency edges. Dependency edges created by the vehicle schedule specify for instance that the last arrival of the vehicle "ICE 1001" connects to the first departure of the vehicle "ICE 1002" with a minimal turnaround time of 20 minutes.

Building the passenger flow can be done in one of two ways. If the input consists of ticket data, the software only has to find the corresponding path in the EAN. Often there is only a general demand in form of an origin-destination matrix (see Section 2.5.1). In this case for every passenger demanding to get from origin to destination, the software will find an optimal route according to the stated preferences. For example, the model could state that all passengers prefer an early arrival without transfers but will award a path with one transfer equal to a route without transfers that is 20 minutes shorter. Once the demand is settled we end up with the EAN enriched with explicit passenger routes as seen in Figure 2.9. Before any operations are performed the *current_time* of the graph has to be set. This is important for update requests and the validity of routes. Now the startup is finished and the program enters a state waiting for instructions.

The shutdown command will simply trigger that information about all passengers will be stored and the program is terminated. This information includes the initial travel time for each group as well as the realized travel time, number of reroutings, and transfers. The update command will trigger that persistent and temporary changes are made. Updates can contain external realized event times and external predictions for future events. Measured event times are persistant, while prediction only stays as long as the realized measure time is received. After events receive external updates all outgoing activity and dependency edges trigger an internal update to ensure that the graph is always consistent when the update is finished. There is also the possibility to cancel a train. After an update command, there is always an update of the *current_time* and classification of transfers associated with changed events. Any passengers that have invalid transfer edges entering a departure event that is earlier than the *current_time*


PANDA - Evaluation View

17:02
Affected Destinations
Additional Critical Transfers
Strong Effect
Normal Effect
Weak Effect
Leaflet | Map data © OpenStreetMap contributors

München Hbf (MH)

	Feeding Train	Connecting Train
Train:	EC 216	ICE 526
Delay:	+2	+0
Planned:	17:41	17:48
Prediction:	17:43	17:48
Transfer Time:	7 min	
Buffer:	-2 min	
Passengers:		330

Decision Criteria

Non-Wait			Wait 2 min until 17:50
4.8	Ø Delay.		1.9
507	Delay ≤ 5 min.	🟩	536
43	Delay > 5 min.	🟩	15
41	Delay ≥ 0,5 h	🟩	14
15	Delay ≥ 1 h	🟩	3
2	Delay ≥ 2 h	🟩	0
0	No Route		0
2650	Σ Delay	🟩	1064

Passengers	Destination	Non-Wait Delay in min	Wait Delay in min
1	FBA	118	0
3	NN	34	2
1	NWH	34	1
1	EG	63	0
2	FF	58	1
5	KKDT	56	0
1	ESIE	81	0
1	KB	-3	0
1	UE	22	0

Fig. 4.6: Screenshot of the evaluation view of the PANDA prototype [Rüc+15]

have to be rerouted. Invalid transfers where the departure is later or equal to the current time are neglected. This is because this event may still receive a delay message if, for example, a dispatcher wants to maintain the transfer by adding a delay. It is also possible that a late update of the arrival event of the invalid transfer is processed and the transfer becomes valid because the vehicle could reduce its delay.

The simulation routine is the core functionality used by dispatchers to create recommendations. Here we process the request for a delay (that will later be undone). In this routine, it is important to identify passengers that are affected by this simulation. Other passengers should later be ignored if the dispatcher only wants to see the local consequences of his simulation and not all passengers with pending conflicts. Now all affected passengers having an invalid route (even if it is after the *current_time*) will receive an alternative route. This alternative is stored with all metrics of interest to the dispatcher. To prepare the full information necessary for an evaluation the simulation method has to be called twice. The first time with the delay for the connecting train and the second time without a delay but with the affected passenger of the first run to compare alternative routings. This concludes all relevant routines of the PANDA framework necessary for most tasks.

4.7 Summary and Impact

The PANDA-framework provided dispatchers with several new possibilities for potentially increasing their quality of service. Having a compact summary of the impact of a decision as well as access to detailed journey information of groups was received positively by dispatchers using it in a field test in Leipzig in June 2015 (see Figure 4.8). Although dispatchers recognized the benefits of the software, they were reluctant to trust software from outside their usual

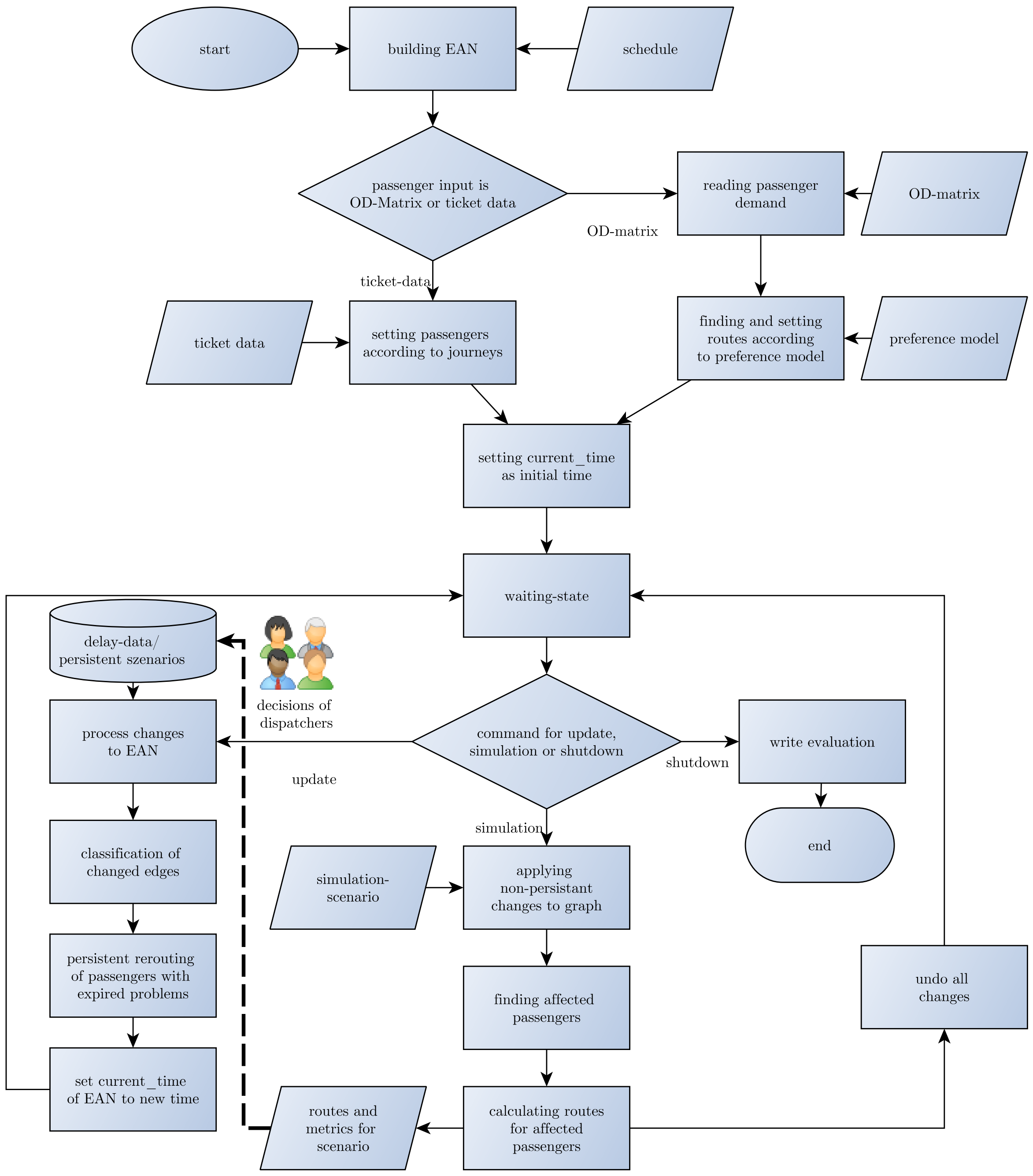

start
building EAN
schedule
passenger input is OD-Matrix or ticket data
OD-matrix
reading passenger demand
OD-matrix
ticket-data
ticket data
setting passengers according to journeys
finding and setting routes according to preference model
preference model
setting current_time as initial time
waiting-state
delay-data/ persistent szenarios
decisions of dispatchers
process changes to EAN
command for update, simulation or shutdown
write evaluation
update
shutdown
end
simulation
classification of changed edges
simulation-scenario
applying non-persistant changes to graph
persistent rerouting of passengers with expired problems
finding affected passengers
undo all changes
set current_time of EAN to new time
routes and metrics for scenario
calculating routes for affected passengers

Fig. 4.7: Flow-chart of processes and data at the core of every task done by the PANDA framework.

Fig. 4.8: Field test using a first version of the PANDA-framework with dispatcher in Leipzig.

workflow created by the Deutsche Bahn. While the topic of the PANDA project raised some interest [Fuh2019], it took two years to improve the software to a point where it was stable and ready for 24/7-use by more dispatchers. The Deutsche Bahn continued funding the PANDA project and specified significant changes in the design of the interface and performance of requests. These following changes were implemented by Frank Berger. After dispatchers and coordinators were satisfied with the changes, PANDA was tested and used for several months in two of seven dispatching centers in Germany. The fact that dispatchers used an academic prototype in practice is an extraordinary success of the endeavor. The official development of PANDA as a prototype tool for dispatchers ended in Mai 2019. Developers from DB Systel include core concepts and functionalities in the KIRA-project announced in late 2018 [Ver].

Timing of Train Disposition 5

Time is the longest distance between two places.

— **Tennessee Williams, The Glass Menagerie [WD44]**

The previous Chapter 4 solely focuses on the binary waiting-decision itself. In a real-life application, however, this is an online problem with several constraints. The online problem of waiting decisions revolves around the scheduled departure time of the connecting vehicle. Once this vehicle is departed, the interchange is no longer possible. In a real-life application, the decision usually has to be made earlier to be still executed. In practice, the last possible moment for an executable decision is 15 minutes before. This time is needed to verify that the infrastructure manager approves the change in track usage. In addition to this process, several involved staff members have to be informed about the decision. For algorithmic purposes, these 15 minutes are the default parameter for the latest simulation of a decision. In this chapter, we will focus on situations in German long-distance railway, until this point.

5.1 Timing of a Decision

Making the decision as late as possible has several benefits. Every second can introduce new changes in delays of feeders or connecting trains. If some feeder train is recovering parts of its delay, the need for waiting could disappear, making a waiting-decision pointless. If the connecting train gains a delay higher than the feeder train due to other reasons, the problem is no longer a waiting-decision. It becomes a problem for the infrastructure manager who has to confirm if both vehicles still can use their designated tracks. The longer the dispatcher is waiting with his/her decision, the better the information about current events. For this reason decisions from dispatchers of Deutsche Bahn are usually made 5-15 minutes before the latest possible moment.

This strategy, however, has two significant drawbacks. The first one is customer discomfort. Individuals without knowledge about their planned connection can experience higher levels of stress than those who already know that their interchange is not possible. The second reason is the decreasing number of possibilities in alternative routes for passengers. Early communication of a non-waiting decision can open up opportunities for an earlier arrival for the passenger. We created the following experiments to find evidence for this hypothesis made in [LMHR16].

5.2 Related Work

Section 4.2 already gives a detailed introduction to important work on this topic. Of all papers already mentioned, [BS14b] comes closest to the work done in this chapter. Bauer and Schöbel focus on the online version of the wait-depart decision which is also the core of our work. We look at the implications of the timing and the communication of alternative routes to passengers when the optimal solution is likely to be the non-waiting case. This is done during a point in time of the online problem when it is not certain that reaching the desired connection is impossible.

What sets our work apart is also the quantification of the benefits passengers could have by using this method. The data and cases used here closely resemble the passenger flow present in the real world long-distance network in Germany.

5.3 Terms and Classification

This section will explain expressions used in the following experiments and define classification-states for interchanges. We already introduced the basic terminology used by dispatchers at the beginning of Chapter 4.

An important concept in the context of dispatching is that of waiting time rules (WTR). For a specific transfer, WTR define a duration of time, the connecting vehicle is supposed to wait until passengers from the feeder vehicle have boarded. The Deutsche Bahn has WTR for every pair of long-distance trains having frequent interchanges. During this time, the infrastructure is usually available for the departing train. For the dispatcher of the Deutsche Bahn, the WTR are only a guideline and are not automatically executed.

Because dispatchers wanted a clear indicator of the demand for action, we created a classification for the status of an interchange taking place in the future. We classify into one of the following four categories [LMHR16]:

SAFE the transfer seems to be safe. A transfer is regarded as safe if the slack time (see Section 4.2) for the transfer is positive, i.e., the transfer will hold unless an unexpected delay of the feeder train occurs.

UNCERTAIN the transfer is uncertain but likely to hold. In this case, the dispatcher should delay the feeder train, by applying the standard waiting time rule. This action maintains this transfer.

CRITICAL the transfer is likely to break unless a dispatcher performs an explicit waiting decision. Here the delay of the feeder train is so large that the desired transfer can only be maintained if the dispatcher decides to delay the connecting train (beyond the standard WTR).

BREAK the transfer will break. Even under the best conditions, the delayed feeder train will arrive too late to reach the connecting train. The delay is so large that waiting for the connecting train is no feasible option. The feasibility distinction depends on whether passengers have reasonable alternatives, in particular in the late evening.

For a future transfer, the delays of the feeder- and connecting train determine the classification of a transfer. To illustrate the dynamical change in delays and classification, Fig. 5.1 gives an example where the status of some specific interchange is monitored for two hours.

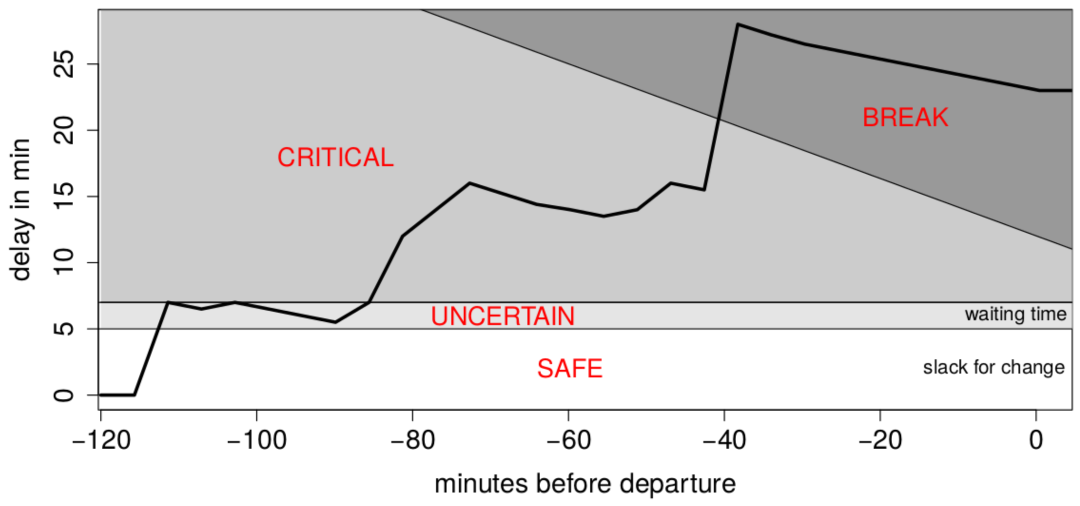


Fig. 5.1: The figure sketches exemplarily the development of the delay in minutes of a single feeder vehicle, two hours before the departure of a connecting vehicle of a planned interchange. Depending on the size of the delay and the potential to regain, PANDA classifies the transfer as **SAFE**, **UNCERTAIN**, **CRITICAL** or **BREAK** [Lem+14].

In this case, 40 minutes before the departure of the connecting train, an additional delay cause the transfer to be classified as **BREAK**. After this delay waiting is no longer an option; dispatchers should communicate new routes for affected passengers. The next step for working with the defined classifications for transfers is their specific implementation. We created three classification schemes [LMHR16] which will be explained in the next section.

5.3.1 Classification Schemes

The EAN modeling the real-world network receives information about realized arrival and departure times every minute. Whenever new information about the realized event times is available, we propagate it through the network. Received information about event times arrive as one of three types.

delay messages messages about realized past events

automated predictions messages about future events generated by algorithms of the Deutsche Bahn

manual predictions messages about future departure times from human dispatchers in case of decisions, this type has priority over automated predictions

Not every event of the route of a train will receive an external prediction. Information about events during a trip have to be consistent for later usage in the simulation. Generating missing timestamps as well as the timestamps used in delay propagation during simulation is difficult. For this task, data about the acceleration, and driving speeds is often useful. When there is no information about infrastructural data, speeds, and acceleration of trains, two types of approaches for predictions are common. The first one utilizes a statistical model using large sets of data [KL17]. We apply a second type of prediction utilizing heuristical propagation rules. We propagate the current delay along the activity edges of a train. If there are defined buffer timess along an activity edge, we subtract those from the propagated delay. In the simulations made in the following experiments, we used the latter method.

Included in this heuristic propagation is the application of waiting time rules. Given a number $wt_a \in \mathbb{N}_0$ for a transfer activity $a \in A_{transfer}$, it shall be maintained if the connecting train has to wait at most wt_a minutes compared to its original schedule. We also consider the duration of the interchange itself, which we denote with l_a. By $w.t^{maxwait} := max\{w.t^{sched} + wt_a | a = (v, w) \in A_{transfer}\}$ the maximum waiting time induced by any delayed feeder train is denoted.

Unfortunately, this heuristic can be too pessimistic on tracks where the driver can make up lost time. The classification needs to assess the situation correctly, even if there is no information on minimum traveling times. Therefore, every event is assigned a lower bound representing the theoretically earliest arrival using a general assumption on minimum traveling times. In practice, it is typically assumed that every train can make up 7% on every driving arc. Using this assumption during propagation for each arrival event $arr \in V$ we obtain a lower bound $arr.t^{lb}$ on its realization time.

The definitions up to this point suffice to create a simple classification scheme using bounds.

Classification by Lower Bounds

We formalize the first classifier in the following fashion. An interchange arc is denoted as $a = (arr, dep) \in A_{transfer}$. We classify it as BREAK if $arr.t^{lb} + l_a > max(dep.t^{pred}, dep.t^{sched} + wt_a + \delta)$. We look at the earliest possible arrival of the passenger at the departure platform $arr.t^{lb} + l_a$. If this is greater than $dep.t^{sched} + wt_a + \delta$, it is considered as BREAK. The parameter $\delta > 0$ specifies a "safety margin". An increase in this parameter makes the classification more conservative. In this method, we also check the predicted departure time of the connecting train $dep.t^{pred}$ and consider if an existing delay of the departure could make the transfer valid. An arc is classified as SAFE, if $arr.t^{pred} + l_a \leq dep.t^{pred}$. A transfer is classified as UNCERTAIN if it is not SAFE but $arr.t^{pred} + l_a \leq dep.t^{sched} + wt_a$. If a transfer could not fit in any of those distinctions, it is CRITICAL. This classification scheme will later be referred to as "STANDARD".

Classification by Transfer Probabilities

The second classification scheme makes use of the massive amount of data about past transfers. It is based on probabilities and has been suggested by the Deutsche Bahn. Dispatchers of Deutsche Bahn had access to a similar version of this scheme giving an indication about future transfers using only three classification states. The type of train has an impact on the probabilities if connections are maintained. Here dispatchers distinguish between InterCity Express (ICE), InterCity (IC), and regional trains.

For a train of type $trainType$ which is currently delayed by d minutes, and a time horizon of h minutes we have an empirical density function $f_{\Delta}^{trainType,h,d} : \mathbb{Z} \mapsto [0,1]$ for the probability that the current delay will change by x minutes in h minutes from now. With the help of this function we can derive the probability $f^{trainType,h,d}$ that future event v for this train will occur at time $t = v.t^{sched} + d + x$, where $d + x \geq 0$ holds. In the following notation Δ is not a parameter. It only marks, that the function is applied to a duration and not a timestamp. We define

$$f^{trainType,h,d}(t) : \mathbb{Z} \mapsto [0,1], \text{ with } f^{trainType,h,d}(t) = f_{\Delta}^{trainType,h,d}(x), x \in \mathbb{Z}.$$

Using this function, we can compute the distribution of departure and arrival times for all future events with respect to the current delay scenario. For a given transfer, we can calculate the joined probability p of cases where the time is sufficient to comply with the minimum transfer times. The underlying assumption is that the probability of delays is independent.

Next we explain the formula used for calculating the probability of a future transfer arc $a = (arr, dep) \in A$. We derive the probability distribution that the transfer will be maintained as follows. As before denote by l_a the minimum transfer time and by $dep.t^{maxwait}$ the maximum waiting time of the departing train. Thus, unless the departing train itself is delayed, it will wait at most until $t_{max} = dep.t^{maxwait}$. The transfer activity requires at least l_a minutes. The probability p that a transfer will be maintained if the feeder train arrives not later than $t_{max} - l_a$ is

$$p = \sum_{t=0}^{t_{max}-l_a} f^{arr.trainType,arr.h,arr.d}(t) \tag{5.1}$$

The probability that a transfer will be maintained after t_{max} depends on the distribution of the departing train as well. In this case

$$p = \sum_{t_1=t_{max}-l_a+1}^{\infty} \sum_{t_2=t_1+l_a}^{\infty} f^{arr.trainType,arr.h,arr.d}(t_1) \cdot f^{dep.trainType,dep.h,dep.d}(t_2) \tag{5.2}$$

where $arr.h$ and $dep.h$ denote the current time horizon for the arrival event arr and the departure event dep, and where $arr.d$ and $dep.d$ denote the current delays of the arriving and departing train, respectively. This classification scheme will late be referred to as STOCHASTIC. The last step needed for classification is a partition of the final probabilities into intervals for each state. These intervals are shown in Table 5.1.

Classification by Fuzzy Logic

This third classification scheme proposed in [LMHR16] utilizes fuzzy logic for classification. The concept of fuzzy logic differs from classical logic, where variables are either true or false. It is a form of many-valued logic, where the truth value of variables may be any real number between

class	**rule**
SAVE	$p \geq 0.96$
UNCERTAIN	$0.6 \leq p \leq 0.96$
CRITICAL	$0.05 \leq p \leq 0.6$
BREAK	$p \leq 0.05$

Tab. 5.1: Translation of probability values of a transfer into classification classes.

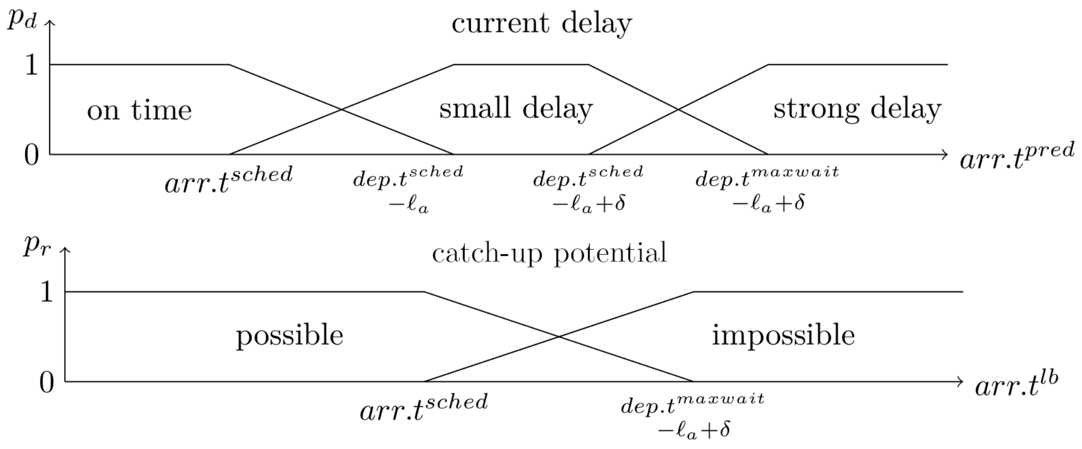


Fig. 5.2: Fuzzification of the two variables *current delay* and $catch-up\ potentia$l for a transfer arc $a = (arr, dep)$ [Lem+14].

zero and one. The reason for choosing this concept is the representation of the vagueness of information present in real-world scenarios. The basic attributes of this scheme are the same as those used by the STANDARD scheme. We will refer to this classification scheme as FUZZY. To classify a transfer concerning uncertainty, we use a classifier based on fuzzy logic. We consider three linguistic variables for the feeder's arrival event:

- the *current delay* with possible values *on time, small delay* and *strong delay*,
- the *catch-up* potential with values *possible* and *impossible*,
- the *state* of a transfer with values SAFE, UNCERTAIN, CRITICAL, and BREAK.

Figure 5.2 shows for an arrival event *arr* how the variables $arr.t^{lb}$ are fuzzified into linguistic variables *current delay* and *catch-up potential* with certainty p_d and p_r, respectively. We use the interference rules shown in Table 5.2 to determine the state of a transfer. The lines of the table are to be read as IF current delay is *current delay* AND regain potential is *regain potential* THEN the transfer is classified as *class*. We use the maximum of p_d and p_r to compute the certainty p_t of a transfer.

We now introduced three classification schemes producing different outcomes. In the next section, we measure their performance against one another and calculate the potential benefit using the best of the three schemes.

current delay	catch-up potential	class
on time		SAFE
small delay	possible	SAFE
small delay	impossible	UNCERTAIN
strong delay	possible	CRITICAL
strong delay	impossible	BREAK

Tab. 5.2: Fuzzy inference rules.

5.4 Experimental Evaluation of the Quality of Classification

There are preconditions to be met before early rerouting is usable in practice. The classification needs to reach a level of accuracy that is compatible with business practices. A BREAK classification, for example, has to be most accurate of all classification states, because it may cause a rerouting for a passenger. If, at a later point in time, a transfer classified as BREAK works or dispatchers maintain it, the rerouted passenger has ended up with a more substantial delay than the delay using his initial route. This scenario would massively hurt the passenger and damage the reputation of the system and company involved. Therefore, the falsely as BREAK classified transfers have to be minimal. However, we have to keep in mind that every prediction made about a future transfer can never be 100% accurate. A new delay to the feeder or the connecting train can happen at any time. Classification errors can also occur in the opposite direction. At the beginning of the day, every transfer is classified as SAFE by default. Through this definition, we classify every transfer that will eventually break wrong at this point. Here we try to give the best possible prediction based on the current information.

The first experiment will consist of looking at the number of classifications for each scheme. We investigate the rate of false-positive BREAK classifications first. After having established the quality of classification, we quantify the potential benefits of early rerouting in the second experiment.

The next section focuses on the composition of our test instances.

5.4.1 Test Instances

The data necessary for conducting these two experiments consist of schedule, delay data, passenger information, and data for the empirical density function $f_{\Delta}^{trainType,h,d}$ for the STOCHASTIC classification scheme. We used a schedule of all long-distance trains and regional trains of 2013. The schedule also included trains not belonging to the Deutsche Bahn. The schedule contains 36,772 trains, 8,592 stations, and the corresponding event activity network consists of about 2 million events. The kind of delay data used is the same as the one used by PANDA framework illustrated in Figure 4.3.

Deutsche Bahn AG provided realistic passenger data for several test days in 2013. This data consists of 2.9 million passengers and their travel connections per day; the passengers travel on roughly 400 000 different routes, with an average travel time of 119 minutes and .73 transfers on average. For the empirical density function $f_{\Delta}^{trainType,h,d}$, a dataset was also provided by the Deutsche Bahn. This dataset is identical to the one used for informing employees of the Deutsche Bahn in their live service in 2013.

5.4.2 Quality of Classification

Evaluating the performance of our classification is strongly time-dependent. The longer period until the transfer, the less accurate the performance. This is why Figures 5.3 and 5.4 all feature the time difference before the transfer on the x-axis. First, we want to compare the number of different classification states between the three schemes. Then we want to focus on the performance of the BREAK classification. We will be using the scheme with the best performing break classification during the early rerouting experiments in Section 5.5.

We recorded for each classified transfer its corresponding state from the first point in time when we classify it as non-safe (by default, we consider all planned transfers as SAFE) until its realization. We set the parameter δ used by STANDARD and FUZZY to the value 4.

Figure 5.3 shows the fractions of the four classes within the last three hours before realization. While all three schemes have a similar fraction of transfers, classified as BREAK, they strongly differ in all other classifications. The STANDARD, for example, is classifying 9% of all transfers as CRITICAL 60 minutes before the transfer, while the STOCHASTIC approach is classifying only 3% as CRITICAL. When focusing on the important BREAK classification, we can notice a big difference. The STOCHASTIC scheme is classifying many transfers as BREAK from 110 minutes to 70 minutes before departure of the connecting train. The other schemes only gradually raise their BREAK classification over time. All three approaches end up with 2% to 3% of transfers classified as BREAK 15 minutes before the departure of the connecting train. Figure 5.4 shows the quality of the three classification schemes concerning false positives. That means cases where the classification predicts BREAK, but the transfer is eventually maintained. The overall best classification rate is obtained by STANDARD, outperforming the two other methods by a small margin. The lack of route-specific probability distributions may explain

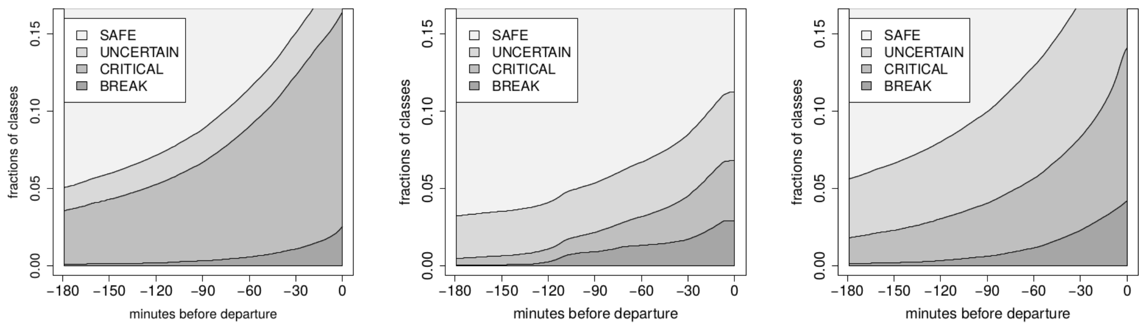


Fig. 5.3: Distribution of the four classes by the three different schemes. STANDARD (left), STOCHASTIC (center), FUZZY (right) [Lem+14].

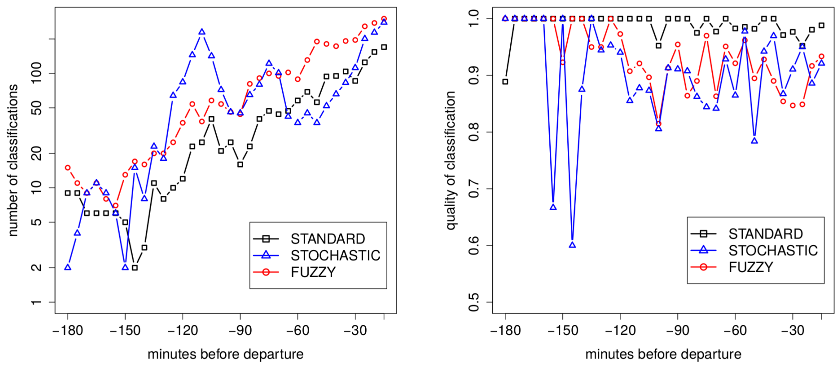


Fig. 5.4: Number of cases (note the logarithmic scale) and accuracy of the classification schemes for class BREAK. Classifications are grouped into bins of 5-minute intervals. The x-axis represents the time before the scheduled event time, while the y-axis displays the fraction of cases with a correct classification [Lem+14].

the lower accuracy of the STOCHASTIC method. The accuracy drops significantly in the 110 minutes to 70 minutes region mentioned earlier.

We observe a trade-off between accuracy and the number of detected cases of type BREAK. High accuracy is crucial since one has to avoid rerouting passengers without any need. On the other hand, we can only utilize the potential of early rerouting, if cases of type BREAK are detected. While STANDARD is the most conservative method with an excellent classification rate, it identifies the overall smallest number of cases of type BREAK. A more detailed analysis of the false-positive cases revealed that the rate of false positives is significantly higher after 22:00h. An explanation for this is that rules for maintaining a connection late in the evening are more generous than that during peak hours.

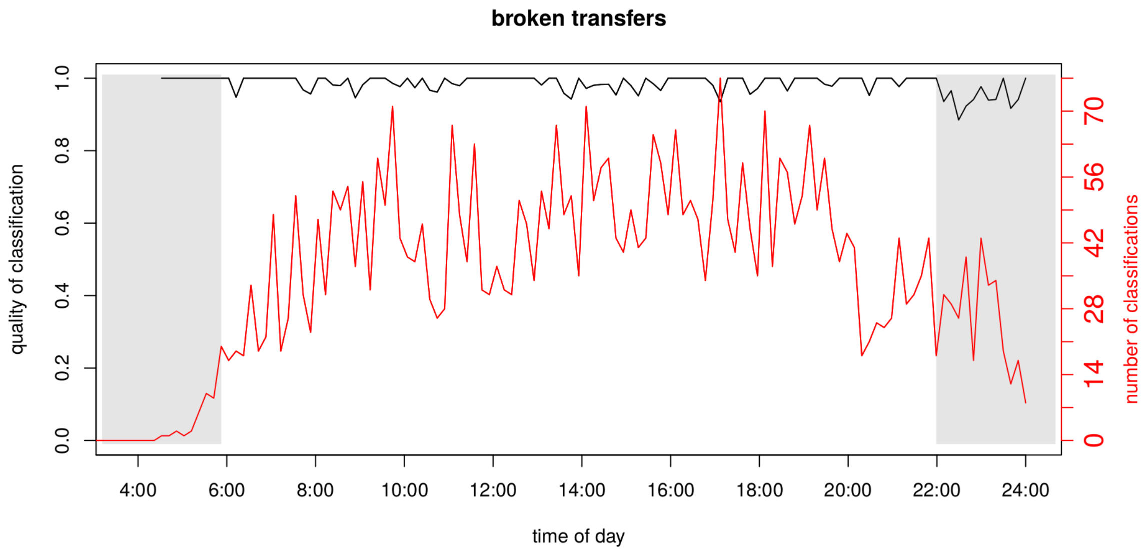


Fig. 5.5: Accuracy of the STANDARD classification of events of type BREAK over time (black). The number of of classifications is shown with the red curve [Lem+14].

Because of its high accuracy, while still maintaining a competitive number of classified BREAK-cases, the STANDARD scheme was selected as the scheme for further experiments and is used as default classification-scheme in the PANDA framework. Having a reliable BREAK classification enables us to conduct experiments rerouting passengers early. The next section will present the result of our experiments concerning the potential benefits of early rerouting.

5.5 Experimental Evaluation of Potential Benefits of Early Rerouting

Now, we answer the question if there are benefits for early rerouting. To observe several aspects of the problem, we conducted three different sets of experiments in [Lem+14].

5.5.1 Experiment A: Synthetic Disruption of Transfers

In the first experiment, we select the 1000 transfers with the most passengers of a given day. For every interchange, we perform an independent simulation. In this simulation, a delay is introduced at the point in time of the first departure event of a train. The added delay is chosen to be large enough for each selected transfer to be classified as BREAK.

There are several possibilities of when to reroute passengers traveling towards a transfer classified as BREAK. The passenger could be notified immediately after information about the BREAK classification is known. Another strategy is waiting until the transfer is x minutes away and then rerouting the passengers. The experiment was conducted several times using the following strategies: decide immediately(∞), 180, 120, 60, or 30 minutes before the planned departure of the broken transfer. This experiment was conducted on several different target days, causing more than 20'000 passengers to be rerouted on these days on average. Figure 5.6 shows the arrival delay of these strategies.

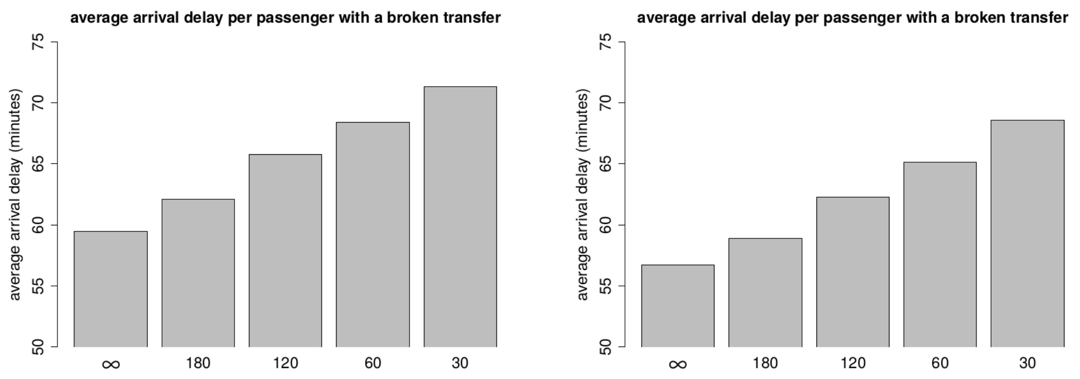


Fig. 5.6: Test day evaluation: April 16 (left) and September 12 (right), 2013. Average arrival delay at the destination if a planned transfer breaks. Rerouting is applied either immediately (denoted by ∞) or (at most) 180, 120, 60, or 30 minutes before the planned departure of the broken transfer [Lem+14].

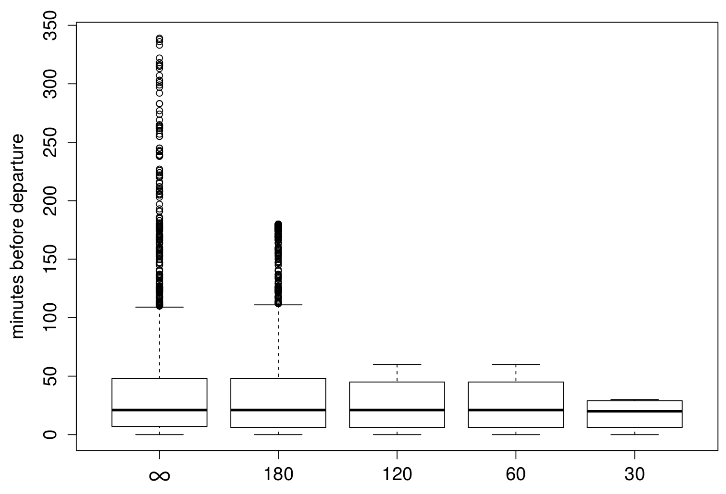


Fig. 5.7: Rerouting is applied either immediately (denoted by ∞) or (at most) 180, 120, 60, or 30 minutes before the planned departure of the broken transfer. For each strategy, the box-plots show the distributions of the moment in time before the planned departure where rerouting is applied for real delay data on September 13, 2013 [Lem+14].

The lower average of 59.5 minutes produced by immediate strategy outperformed the 30 minutes strategy by 11 minutes. We observed a similar difference in average delay on every tested day. The intermediate strategies 60, 120 and 180 show an improvement in average delay.

Due to the position of the major transfers concerning the first departure of the train, not every passenger can be rerouted more than 180 minutes in advance. If the transfer is happening after the first arrival of the feeder train, there is no possibility of early rerouting. This fact led us to look at the difference in time between rerouting and transfer. This difference is captured in Figure 5.7.

The number of passengers benefiting from the immediate and 180 minutes strategy is relatively small. We observed that 75% of all reroutings take place at most 50 minutes before the departure of the connecting train. Therefore for most passengers, all strategies do mainly the same. This fact explains why the 8 minutes of average benefit is relatively small. This, however, also means that the passengers being rerouted more than 120 minutes in advance had a significant impact on the average arrival delay. They still produce an 8 minutes difference for the average passengers.

5.5.2 Experiment B: A Study on Real Disruptions

Experiment A gave a detailed analysis of an upper bound of the potential benefit eliminating interference with other delays. It is, however, necessary to test the benefits of early rerouting in a more realistic experiment. We created experiment B for this purpose. This experiment consists of two simulations of one day of real traffic with the same passenger data used in the first experiment. The only difference in the two runs is the application of early rerouting rules. First, we conducted the simulation with the late 15 minutes in advance routing strategy. In the second run of the simulation, we used the immediate rerouting strategy. Comparing the difference

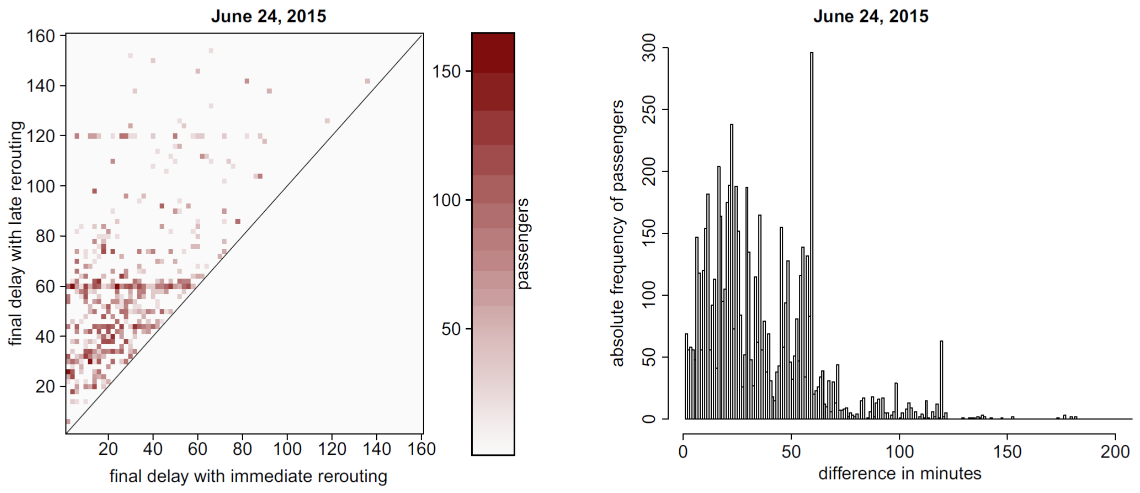


Fig. 5.8: Results of Experiment A: the heatmap $(left)$ shows the final delay at the destination of all passengers for the two strategies: immediate rerouting $(x - axis)$ vs. rerouting only 15 min in advance $(y - axis)$. The histogram $(right)$ shows the difference of the delays, i.e. the arrival time for the 15-min-in-advance strategy minus the arrival time of immediate rerouting final delay [Rüc+15]

in delay of both approaches for every group of passengers revealed new insights. While this difference was always zero or positive by design in experiment A, some groups experience worse arrival times when immediate rerouting is applied. With real-world train delays, one expects this behavior. Any decision about the choice of route can prove to be the wrong one when new delays occur. For a better comparison between delays in the 15 minutes waiting strategy and the immediate strategies, we created heatmaps and histograms for better visualization. Figure 5.8 shows the results of experiment A while Figure 5.9 shows the results of experiment B.

A point above the diagonal means that the final delay of the corresponding passenger is higher with the 15-min-in-advance strategy than with the immediate rerouting strategy. A point below the diagonal means a higher delay in the immediate rerouting strategy. The right part of Figure 5.9 shows the distribution of winners and losers. For most rerouted passengers (namely for 88.4%), the strategy has the same result. Among the remaining passengers, immediate rerouting has an advantage for 896 passengers (68.5%), whereas 411 passengers (31.5%) are better off with the 15-min-in-advance strategy. For those passengers where the choice of the strategy has an impact, the mean difference (that is, the mean reduction of the final delay) is about 14.7 min in favor of immediate rerouting. We conclude that the immediate rerouting strategy is beneficial from the perspective of an average passenger, but can also worsen matters for a non-negligible fraction of passengers.

5.5.3 Experiment C: Real Disruptions with a Refined Strategy

During the evaluation of Experiment B, we noticed cases where the alternative route chosen for rerouting had its differing first event much later than the time of the rerouting. The alternative

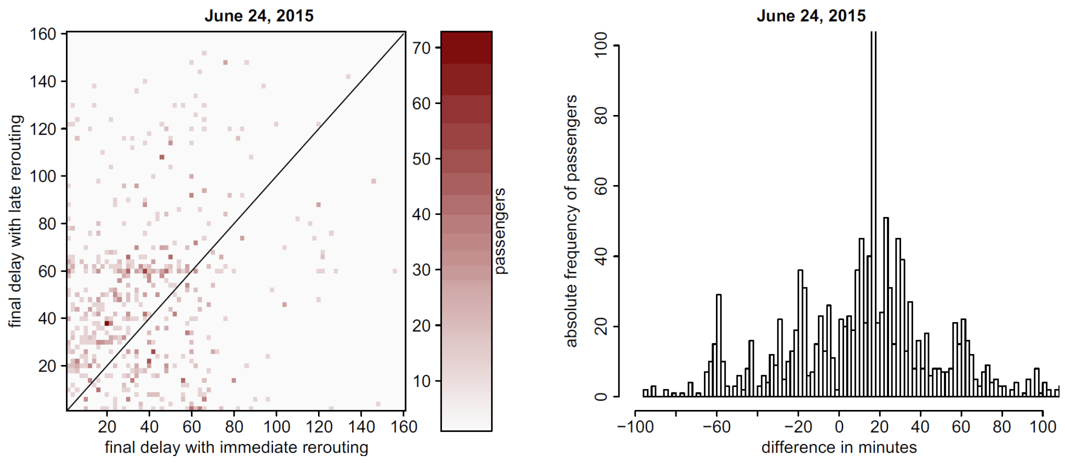


Fig. 5.9: Results of Experiment B: the heatmap ($left$) shows the final delay at the destination of all passengers for the two strategies: immediate rerouting ($x - axis$) vs. rerouting only 15 min in advance ($y - axis$). The histogram (right) shows the difference of the delays, i.e., the arrival time for the 15-min-in-advance strategy minus the arrival time of immediate rerouting. We truncated the cases with no difference in the heatmap and histogram for better visibility [Rüc+15].

route has been better at the time of its creation, but can sometimes be worse at the time the passengers first deviates from his initial route. To change this behavior, we created a hybrid strategy we called *refined early rerouting*.

In contrast to the immediate rerouting strategy, we wait with our recommendation as long as possible. More precisely, we compare the best available alternative with the currently used route and determine the first moment in time where the two routes differ. This analysis gives us the latest moment where a passenger group could switch to that particular alternative. A few minutes before this moment, we reconsider the current delay status. Since the first moment when we classified that the transfer might break, the delay situation may have changed. For example, the connecting train of the originally planned transfer also catches a delay so that our classification of it as being not maintained becomes obsolete. If we still predict the transfer to break, we perform a second search for an alternative route. We only reroute to the previously considered alternative route if the transfer is classified to break and if also the alternative is still feasible. It might be that meanwhile also the alternative route faces some previously unknown delay. By doing so, this refined strategy tends to avoid unnecessary reroutings of passengers.

For Experiment C, we used the same experimental setup as in Experiment B, but this time, we focus on the comparison of the immediate rerouting strategy with the hybrid strategy. Again both strategies have winners and losers. The hybrid rerouting strategy produces more than twice as many winners as losers. The heatmap in Figure 5.10 shows that more than twice as many cases lie above the diagonal than below. A point above the diagonal means that the final delay of the corresponding passenger is higher with the immediate rerouting strategy than with the new rerouting strategy. Likewise, we consider the difference in the arrival times of both strategies (precisely, we take the arrival time of the immediate rerouting strategy minus the

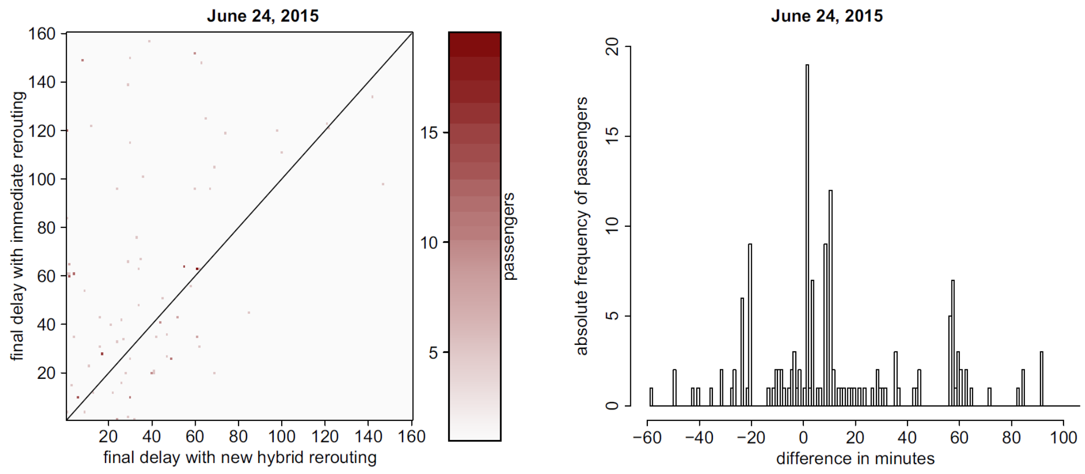


Fig. 5.10: Results of Experiment C: the heatmap ($left$) shows the final delay at the destination of all passengers for the two strategies: hybrid strategy ($x - axis$) vs. immediate rerouting ($y - axis$). The histogram ($right$) shows the difference of the delays, i.e., the arrival time for the immediate rerouting strategy minus the arrival time of the hybrid strategy. We truncated the cases with a zero minutes difference in heatmap and histogram for better visibility.

arrival time of the new hybrid strategy). Hence, if the difference is positive, then this indicates an advantage for the hybrid strategy. The right part of Figure 5.10 shows the distribution of winners and losers. For most rerouted passengers (namely for 98.5%) both strategies are equal because there is no alternative path that is faster.

5.6 Discussion and Summary

This section covered the challenge of timing a decision made by dispatchers. We introduced and tested methods for the classification of transfers. Of the three presented methods, the *classification by lower bound* had the lowest false positive rate while maintaining a similar number of interchanges classified as BREAK. We, therefore, used this method in further experiments. The first set of experiments uses isolated artificial delays to study the potential benefits of early rerouting. We then analyzed benefits by varying the timing of when to communicate the BREAK classification, and passengers are rerouted. Early rerouting improved the average delay of investigated cases by a significant margin. The majority of routings are equal to the ones available in the last 30 minutes. While early rerouting is beneficial, only passengers having no limitations on the validity of their tickets can currently use this method. The experiment with real data also shows that a non-negligible number of passengers may suffer from a higher final delay. As the occurrence of delays and disruptions is unforeseeable, there is no perfect strategy that is optimal for all individual passengers. A notable problem with the second experiment is the fact that not all rerouting occurs immediately after BREAK classification. The communication of the rerouting especially if future transfers are still uncertain is detrimental. This practice causes groups with (later) executable transfers to worsen the total delay.

We suggest using a modified strategy in practice. First, calculate the best alternative route during BREAK classification. Then wait before communicating the alternative until a point in time just before the diverging journey begins. This strategy minimizes the risks of the communication of a more inferior route. Future research could improve real-world performance by reducing the number of cases where we produce unstable alternate routes. One could achieve this by ranking alternative journeys by a measure, including robustness. One key benefit of early rerouting in contrast to our other work with delay management is that it has a positive impact of delay management problem itself. We reduced the number of passengers with critical or broken transfers, which could potentially change dispatching decisions in the future. This concludes our work on early rerouting. The next chapter is investigating the question to which extent PANDA-recommendations are sensitive to inaccuracies in the input data.

Sensitivity Analysis 6

A person ignorant of the possibility of failure can be a halfbrick in the path of the bicycle of history.

— **Terry Pratchett, Equal Rites [Pra87]**

PANDA gives recommendations using assumptions we should examine closely. One assumption is the correct input of passengers traveling on certain routes. In the context of a recommendation in German long-distance trains, this assumption is only applicable to a certain extent. The data for the passenger flow is not exact. We explained this in more detail in Section 2.6.1. Dealing with such inaccuracies is the main focus of this chapter. The core of this work was first published in [LMHR16].

Important parts of the workflow of a dispatcher were already introduced in Section 4.4. At the beginning of this chapter, we describe key performance indicators in greater detail. A combination of these indicators will be used to generate a recommendation for the waiting decisions problem. After analyzing the influence of single groups on this recommendation, a method will be presented, finding the smallest change in passenger data resulting in a different recommendation. The rest of this chapter will evaluate the sensitivity of recommendations made by the PANDA software during the average days of passenger flows.

6.1 Related Work

Most related publications were mentioned already in the last chapter concerning the timing of decisions. [Bie06; BS07] conducted experiments using a passenger flow generated by their own methods. However they reported problems evaluating the realistic benefits of their choices. More accurate passenger data often exists when smart card data is available. Researches using *smart card* data showed that a combination of historical data and short-term predictions for passenger demand could be more than 87% accurate [Che+11]. However, there are multiple causes for misjudging the number of passengers. Fear of strikes, communicated by the press, may lead to a significant drop in passengers. On the other hand, the convenience of a new line can lead to a steady shift in demand. As an example, when the new ICE line between Berlin and Munich became available, the level of demand for the railway connection increased by 23% to 46% [Deu18]. Part of the demand for air travel [tag19] could also shift toward more long-distance train traffic if ambitions to reach climate goals continue. Prediction of future trends in such direction is challenging. In Korea, for example, the demand for a specific flight connection was expected to drop to 14% after the introduction of a high-speed railway alternative. The ticket-sales only dropped to 28% [PH06]. Such examples show that actual demand is hard to predict. Any model making decisions on assumed demands rather than exact data needs to take care not to give dispatcher harmful advice.

6.2 Giving Recommendations

The PANDA software was designed to assist the dispatcher during the decision process giving some indicators about the difference in specific key performance indicators (KPI) such as total delay of a decision. Deutsche Bahn suggested parts of these KPIs:

1. the total delay at destination (or, equivalently the average delay) of all passengers
2. number of passengers with a delay at destination ≤ 5 minutes (regarded as “on-time”)
3. number of passengers with a delay at destination ≥ 6 minutes
4. number of passengers with a delay at destination ≥ 30 minutes
5. number of passengers with a delay at destination ≥ 60 minutes
6. number of passengers with a delay at destination ≥ 120 minutes
7. number of passengers without any acceptable alternative, arriving on the same day or with less than 4h of delay.

There has been some debate about the use of recommendations for one or the other decision. For the prototype and academic purposes, we used the following rule:

Majority Rule A recommendation for a decision is only made if there is a majority of key performance indicators favoring one decision.

An alternative to this rule is a weighted sum of indicators. This can be customized to reflect know preferences and help in atomized evaluations.

A recommendation in both cases should only be made if it withstands small changes in passenger data. If, for example, the algorithm assumes ten passengers are using a particular transfer while in reality, there are only five, this could lead to a different recommendation. While the given example had a major change in passengers concerning this transfer, we expect the data imported into PANDA to be at least accurate to a certain percentage greater than 75%. With a certain degree of guaranteed accuracy raises the question, if we can give a recommendation that is not influenceable by small changes in the passenger flow. Answering this question requires the calculation of the minimum number of passengers that can change a recommendation. We present a technique for this task in the next section.

6.3 ILP for Minimising a Significant Change in Passengers

Before diving deeper into our ILP, we need to introduce a definition of what technically counts as a passenger affected by a decision:

affected passengers A group becomes affected by simulation of two scenarios when A) it has a different arrival time or B) it gets a different route

We present a method with respect to the given flow for changing a recommendation by adding or subtracting the minimum number of passengers. Let us first differentiate between passengers affected by the decision and those passengers that are not affected by the decision. This distinction done by the PANDA software is selecting the relevant groups for the dispatcher. Suppose that we have relevant passenger groups $p_1, p_2, \ldots, p_n$. The set of affected passenger groups is denoted by $\mathcal{P}$ and let the expected number of passengers in group p_i be c_i where $c_i \in \mathbb{N}$.

The delay at the final destination of a group p_i in the waiting and the non-waiting case are denoted by the function $W(p_i)$ for waiting and $N(p_i)$ for the non-waiting case. To model the different KPIs, we create a function for every criterion. For the ℓ-th decision criterion, let $crit^{\ell}(\mathcal{P})$ be the function which maps to $\{-1, 0, 1\}$, where we identify the function value 1 with the decision WAIT, the value -1 with the decision NO-WAIT, and the value 0 for a tie (NEUTRAL). For example, we can define for the first criterion (the total amount of delay at the destination) the function as

$$crit^1(\mathcal{P}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} W(p_i)c_i < \sum_{i=1}^{n} N(p_i)c_i \\ -1 & \text{if } \sum_{i=1}^{n} W(p_i)c_i > \sum_{i=1}^{n} N(p_i)c_i \\ 0 & \text{otherwise .} \end{cases} \tag{6.1}$$

With k different criteria, we decide WAIT if $\sum_{\ell=1}^{k} crit^{\ell}(\mathcal{P}) > 0$, and NO-WAIT if this expression is negative. Otherwise, we obtain a tie. Given these definitions, the sensitivity problem can now be formalized in the following fashion:

Given: A critical or broken transfer for which the decision is either WAIT or NO-Wait, and a set of n affected passenger groups $\mathcal{P}$, and values $c_i, W(p_i), N(p_i)$ for each $p_i \in \mathcal{P}$.

Task: Determine numbers $\tilde{c}_i \geq 0$ such that the total deviation from the given multiplicities of passenger groups $\sum_{i=1}^{n} |\tilde{c}_i - c_i|$ is as small as possible and the overall decision is reversed to the opposite one. Here we only allow changes in size of existing passenger groups.

We investigated several greedy approaches to solve this problem, but none proved to have a guaranty of optimality. Here is one example of a greedy method that always returns a valid solution for changing a decision if one exists. It also can be optimal in some cases. For simplicity, the algorithm is written to change a non-waiting decision to a *waiting decision*. Algorithm 1 does not find an optimal solution in cases where many groups have small changes in their final destination. No other greedy algorithm was found, guaranteeing an optimal solution. The complexity status of this problem, therefore, remains unknown to us.

To solve this problem to optimality, we created an ILP formulation. This ILP formulation was presented in [LMHR16] and its notation is used in this section. Let us first sketch the basic ideas behind this formulation. Let x_i^+ and x_i^- be integral variables, describing the number of passengers which are added to or subtracted from a group $p_i \in \mathcal{P}$, respectively. In this formulation, k is the number of criteria.

```
Algorithm 1: Greedy for changing a recommendation - to waiting
  Data: Groups: $p_i \in \mathcal{P}$ , passengers in group $c_i \in \mathbb{N}$, function of delay data $W(p_i), N(p_i)$
  Result: $\tilde{c}_i \in \tilde{C}$ for which the recommendation is reversed
  $\tilde{C} \longleftarrow C$;
  forall $\tilde{c}_i \in \tilde{C}$ ORDERED BY $W(P_i) - N(P_i)$ do
      while $\tilde{c}_i > 0$ do
          $\tilde{c}_i \longleftarrow \tilde{c}_i - 1$ ;
          if recommendation changed then
              return $\tilde{C}$
  return "Msg: No change by subtraction possible"
```

The total difference in evaluated criteria is denoted as $\Delta := |\sum_{j=1}^{k} crit^j(P)|$. To reverse the overall outcome, the net change of the individual criteria must be at least $\Delta + 1$.

A technical complication for our ILP formulation comes from the fact that each criterion can assume the NEUTRAL state in addition to WAIT and NO-WAIT. Hence, we have to distinguish the cases of changing a criterion from WAIT to NEUTRAL or NO-WAIT, from NO-WAIT to NEUTRAL or WAIT, and from NEUTRAL to WAIT or NO-WAIT. For this purpose, we introduce the following three $\{0, 1\}$-decision variables for each criterion. For criterion, j, variable w_j denotes that the j-th criterion will vote for WAIT in an optimal solution of the sensitivity problem, whereas variable $\overline{w}_j$ means that it is in favor for NO-WAIT. The third possible outcome, a tie (NEUTRAL), is denoted by t_i. Let $\mathcal{K}$ be the set of all criteria. Concerning the situation before solving the ILP, let $\mathcal{W}$ be the subset of criteria in favor of WAIT, $\mathcal{NW}$ the subset of criteria in favor of NO-WAIT, and $\mathcal{T}$ the remaining subset of criteria with a tie. Assuming that the current decision is NO-WAIT, we obtain the ILP minimizing the objective function 6.2 (a similar ILP is derived if the current decision is WAIT).

M is a sufficiently large constant, and the coefficients $a_{ij} \in \mathbb{Z}$ denote the contribution of passenger group $i \in \mathcal{P}$ to criterion $j \in \mathcal{K}$, and the coefficients $b_j \in \mathbb{Z}$ the amount by which the current evaluation of criterion $j \in \mathcal{W} \cup \mathcal{NW}$ has to be changed in order switch this criterion from WAIT or NO-WAIT to NEUTRAL. The objective function expresses that we want to minimize the necessary change in passengers. In an optimal solution, at most one of each pair of variables x_i^+, x_i^- can be strictly positive. Equality (6.3) in combination with the 0-1-variable bounds in (6.13) ensures that exactly one of the three possible states (WAIT, NO-WAIT, NEUTRAL) is chosen for each criterion. In inequality (6.4), the left-hand-side sums up the total change of criteria. The sum must be large enough to change the decision from NO-WAIT to WAIT to fulfill the inequality. Inequalities (6.5)-(6.10) link the change in passenger flow to the different criteria. We use a "big-M" formulation to ensure that we can always fulfill all of these inequalities. The expression $z_j = \sum_{i \in \mathcal{P}} a_{ij}(x_i^+ - x_i^-)$ measures the effect of the passenger flow change on criterion $j \in K$. For $j \in \mathcal{NW}$ (the j-th criterion is currently in favor of NO-WAIT), we can fulfill Inequality (6.5) with $w_j = 1$ (or $t_j = 1$) if z_j is large enough to reverse the criterion to WAIT (or to NEUTRAL, respectively). Otherwise, we can always choose $\overline{w}_j = 1$. Since it

$$\min \sum_{i \in \mathcal{P}} (x_i^+ + x_i^-) \tag{6.2}$$

subject to

$$\begin{aligned}
w_j + \overline{w}_j + t_j &= 1 && \text{for } j \in \mathcal{K} && (6.3)\\
\sum_{j \in \mathcal{NW}} (2w_j + t_j) + \sum_{j \in T} (w_j - \overline{w}_j) + \sum_{j \in \mathcal{W}} (-2\overline{w}_j - t_j) &\geq \Delta + 1 && && (6.4)\\
\sum_{i \in \mathcal{P}} a_{ij}(x_i^+ - x_i^-) - (b_j + 1)w_j - b_j t_j + M\overline{w}_j &\geq 0 && \text{for } j \in \mathcal{NW} && (6.5)\\
\sum_{i \in \mathcal{P}} a_{ij}(x_i^+ - x_i^-) - (b_j + 1)\overline{w}_j - b_j t_j + M w_j &\geq 0 && \text{for } j \in \mathcal{W} && (6.6)\\
\sum_{i \in \mathcal{P}} a_{ij}(x_i^+ - x_i^-) + M(t_j + \overline{w}_j) &\geq 1 && \text{for } j \in \mathcal{T} && (6.7)\\
\sum_{i \in \mathcal{P}} a_{ij}(x_i^+ - x_i^-) - M(t_j + w_j) &\leq -1 && \text{for } j \in \mathcal{T} && (6.8)\\
2 \cdot \sum_{i \in \mathcal{P}} a_{ij}(x_i^+ - x_i^-) - t_j + M\overline{w}_j &\geq 0 && \text{for } j \in \mathcal{T} && (6.9)\\
-2 \cdot \sum_{i \in \mathcal{P}} a_{ij}(x_i^+ - x_i^-) + t_j + M w_j &\geq 0 && \text{for } j \in \mathcal{T} && (6.10)\\
x_i^- &\leq c_i && \text{for } i \in \mathcal{P} && (6.11)\\
x_i^+, x_i^- &\in \mathbb{N}_0 && \text{for } i \in \mathcal{P} && (6.12)\\
w_j, \overline{w}_j, t_j &\in \{0, 1\} && \text{for } j \in \mathcal{K}, && (6.13)
\end{aligned}$$

helps to fulfill Inequality (6.4), we may safely assume that an optimal solution prefers setting $w_j = 1$ over $t_j = 1$ and the latter over $\overline{w}_j = 1$. For $j \in \mathcal{W}$, Inequality (6.6) works analogously. Inequalities (6.7)-(6.10) together model the case that the current state of a criterion is NEUTRAL. Here, a case analysis shows that $z_j > 0$ implies $w_j = 1$, $z_j < 0$ implies $\overline{w}_j = 1$, and $z_j = 0$ implies $t_j = 0$. Inequality (6.11) ensures for each group that we cannot subtract more passengers than we currently have. In the next section, experiments are presented that illustrate how vulnerable recommendations of PANDA can be.

6.4 Experiments

For all experiments, we used the German train schedule of 2015, including all long-distance and regional trains. This schedule includes about 66000 trains and a million events per day. The realistic passenger flow used in this experiment is similar to the ones used in Chapter 5. For our evaluation, we used recorded data for actual delays of eight weekdays in June and October 2015. These actual delays are part of the PANDA data import workflow (see Section 4.4). For every test day, we simulated the same conditions for the recommendation as they were on the day of the field-test. The framework begins at midnight, introducing delays every minute at precisely the time they would be entered into the live system.

For each detected critical or broken transfer, we simulate a PANDA decision 15 minutes before the connecting train is scheduled to depart. These 15 minutes are chosen purposefully and are motivated by decision times of real dispatchers (see Chapter 5). If the evaluation suggests a recommendation, we calculate the minimal number of passengers to change the suggested strategy. Based on the data available in the computation of the recommendation, the ILP is constructed using affected groups. We then solve the corresponding ILPs by using the non-commercial SCIP Optimization Suite[1] in version 3.2.1 with SoPlex 2.2.1 as the ILP-solver [Ach+08]. Overall, 73486 PANDA decisions were calculated and analyzed according to their sensitivity. We present the results in the next section.

6.4.1 Experimental Results

For evaluating the results, another step is necessary. Because the different simulations can have a massive difference in affected passengers, comparing the absolute number of passengers necessary for flipping a decision can be misleading. Therefore, we normalize the necessary total passenger change (i. e., the value of the optimal ILP solution) by the number of affected passengers. This method gives us a kind of reliability measure

$$rel = \frac{\text{total passenger change}}{\text{\#affected passengers}}.$$

The optimal solution will never change the passenger flow by more than removing all existing passengers. Regardless of whether the recommendation favored WAIT or NO-WAIT by removing

[1] http://scip.zib.de

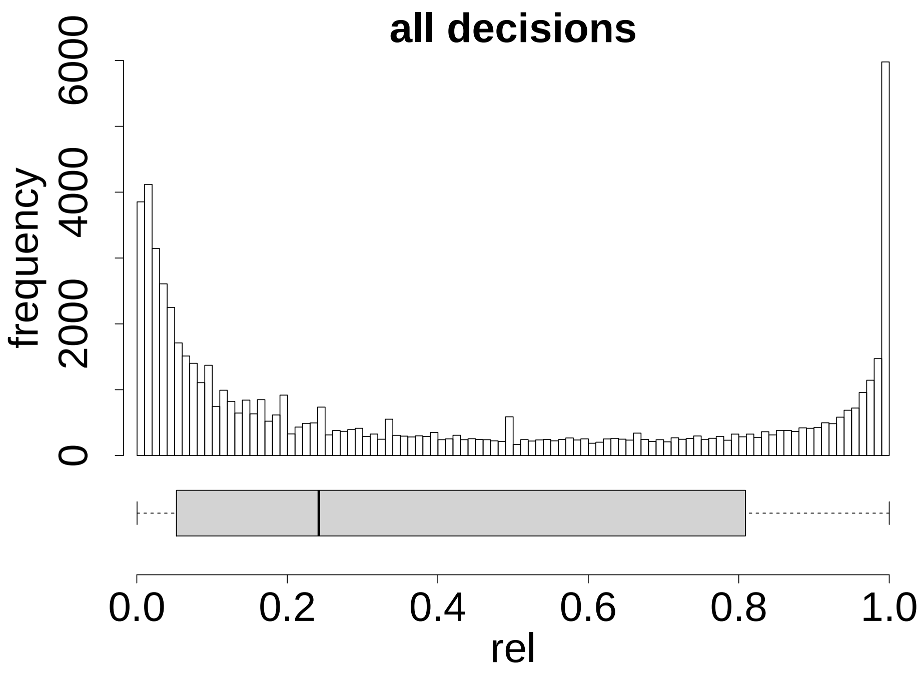


Fig. 6.1: Empirical distribution of rel values of all decisions [LMHR16].

all affected passengers the original problem will become nonexistent. Therefore having a positive contribution to the current decision, we have $0 \le rel \le 1$. The larger the value of rel, the more robust is the corresponding decision. Using this reliability measure looking at the observed distribution of rel in Figure 6.1 gives us important information about the general sensitivity of decisions.

This distribution is bimodal. Therefore, any assertion about an average sensitivity would be misleading. There are two distinct clusters of recommendation. The cluster on the left-hand side of Figure 6.1 is showing a large number of cases where only 1% to 10% need to change to reverse the recommendation. On the right-hand side, there are several recommendations very stable. More than 6000 decisions are trustworthy unless we remove every single passenger involved.

What is the result of those findings in the context of recommendations with a given input? Let us suppose our input is at least 80% accurate. This accuracy implies that we can trust any recommendation having a rel value greater than .2. Because the median of our distribution is at $rel = .24$, more than half of our recommendations would be trustworthy. Having a number that quantifies the accuracy q, a recommendation should always be displayed if $rel > 1 - q$.

If we distinguish between WAIT and NO-WAIT recommendations, we can gather further useful information about the sensitivity. For this reason, they are shown in Figure 6.2 and Figure 6.3 side by side.

It is an intriguing observation that WAIT decisions turn out to be more robust than NO-WAIT choices on average. The median of the sensitivity measure is .29 in case WAIT, in comparison to .19 in case NO-WAIT. It is also worth mentioning that there are many cases of waiting

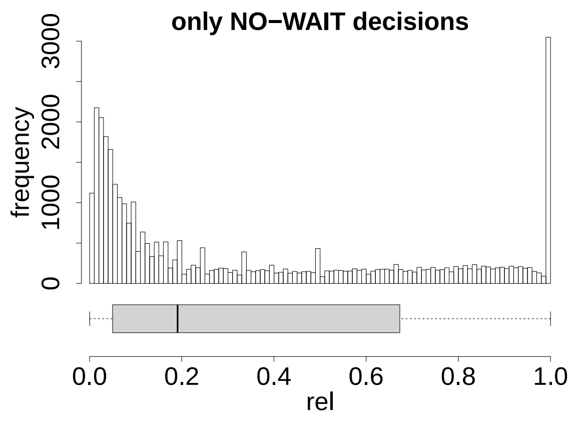


Fig. 6.2: Empirical distribution of *rel* values of all decisions where not waiting would be a recommendation [LMHR16].

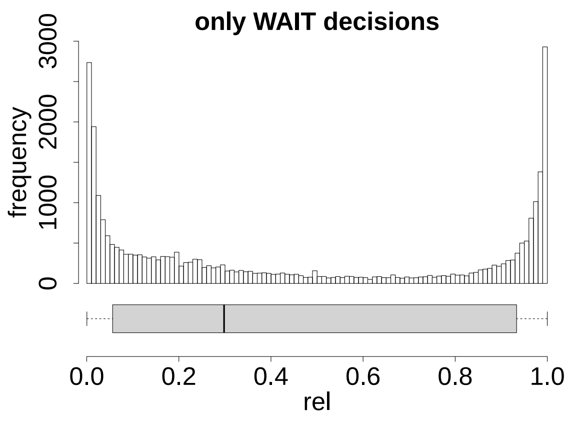


Fig. 6.3: Empirical distribution of *rel* values of all decisions where waiting would be a recommendation [LMHR16].

recommendations that are incredibly fragile, having a *rel* value of $< .025$. These are cases where only one or two passengers want to transfer into a vehicle having many passengers that are robust to an additional small delay. Especially for this case, it would be good practice to verify the existence of these passengers in real-time. Then we can identify such scenarios, and a dispatcher could be informed about that.

6.5 Summary and Criticism

In this chapter, we investigated the sensitivity of recommendations made by PANDA. The recommendations were created based on a majority rule of KPIs used by dispatchers. The stability of those recommendations was tested, finding the minimal number of passengers needed to change a recommendation using an ILP. Experiments using an 8-day data set revealed that there are two distinct clusters of decisions. One set of choices was very robust to input inaccuracies. Another part of decisions can be altered by little changes in the input. The computation of the sensitivity of a recommendation using the SCIP-software can be performed in less than one second making it usable for live calculation in PANDA.

One major problem is the big set of cases with high sensitivity. Small changes in passengers often can change the recommendation. Because of this fact dispatchers requested an input field for the simulation where they could change the number of passengers in certain groups matching their information obtained by other staff members.

Another problem is the sensitivity of the results to additional delays or wrong interchange times. If the execution of a waiting decision in the real world takes more than two minutes longer than in the simulation, the results can also differ. This discrepancy in time makes waiting recommendations even more vulnerable to criticism by the dispatchers. During the field test, dispatchers sometimes notice that the assumed interchange times did not always match their experience.

7 Coupled Decisions

Shallow men believe in luck or in circumstance.
Strong men believe in cause and effect.

— **Ralph Waldo Emerson [Eme60]**

Continuing with the line of work improving PANDA, we discovered two weaknesses. The nature of the simulation process creates the first one; the second one is of a practical kind.

1. Passengers with two interchanges and a delay that is greater than each buffer on their interchanges can not get their route fixed by a waiting decision. Even if the first departing train waits the passenger is rerouted to a route that does not contain the planned second transfer.

2. To handle all waiting decisions, the overall task is distributed to regional disposition centers. Moreover, several dispatchers work in the same center, each being responsible for a certain subset of trains. When decisions affect trains and passengers outside this area we assume that there is clear communication of measures and intent where there maybe not.

Decisions may create new conflicts where acting is also necessary. Delaying a train can cause not only knock-on effects on delays but also create new critical transfers. One fact that makes delaying a train hard is that passengers often plan their interchanges with little or no slack. Even though there were many problems with the punctuality of the German railway, this had little influence on passenger behavior. 35% of passengers having interchanges from one long-distance train to another planned their trips with less than four minutes slack, as seen in Figure 7.1. Due to this lack of slack, a train containing many passengers cause the simulation to report many new broken connections. If we simulate a decision, every transfer that is currently critical will be interpreted as broken, rerouting all affected passengers. Essentially two distinct simulations are made. One where we break every critical transfer, and one were the current ones are kept, but there may be some additional broken transfers. For some of those additionally interrupted transfers, a PANDA decision could produce a waiting recommendation with low sensitivity. If the combined scenario with every affected passenger group is analyzed, a recommendation could be "keep both interchanges". Such a scenario will be called *coupled decision*. Figure 7.2 illustrates this by an example involving three trains. This chapter is about the integration of *coupled decisions* in the PANDA-framework and experiments analyzing the usefulness of this technique. In the next section, related work concerning the calculation of more than one decision at a time is discussed.

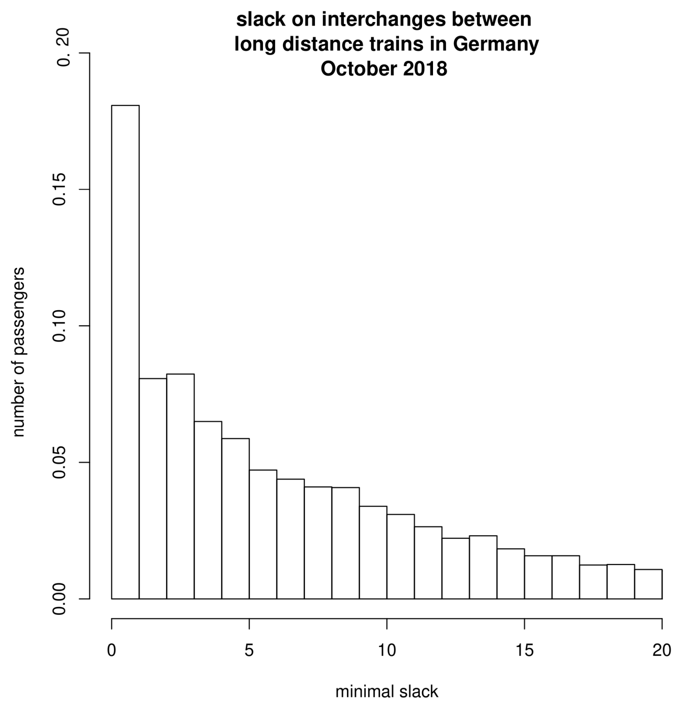


Fig. 7.1: Distibution of the slack of interchanges used by passengers in October 2018

7.1 Related Work

Most of the related work about this topic was already presented in Section 4.2. Sophisticated global waiting strategies can be beneficial to customers like the ones described in [Bie06; BS07]. In addition to those strategies, optimization of waiting time rules can be a useful tool for improving the robustness of connections. Most of the research already mentioned using ILP formulations of the delay management problem like [Sch01] are optimizing all decisions in one step. This method, however, makes it impossible to evaluate the benefit between single isolated decisions as is done later in Section 7.4.2.

7.2 Coupled Decisions and Waiting Time Rules

Before going more into detail about *coupled decisions*, a similar approach has to be covered. The concept of waiting time rules was presented multiple times in this dissertation. When working with dispatchers of the Deutsche Bahn there is a distinction between *standard WTR* and *official WTR*. The *standard WTR* can be stated as a small table having an entry for every combination of train-type. The entry states how long each type of train has to wait for the other type if there are changing passengers. The *official WTR* is a complex set of rules applicable to special connections. In the context of this dissertation, we only talk about *standard WTR*. A controversial topic is whether to use waiting time rules automatically during delay propagation. Generally, the automatic application of waiting time rules has a positive effect on total passenger delay. The

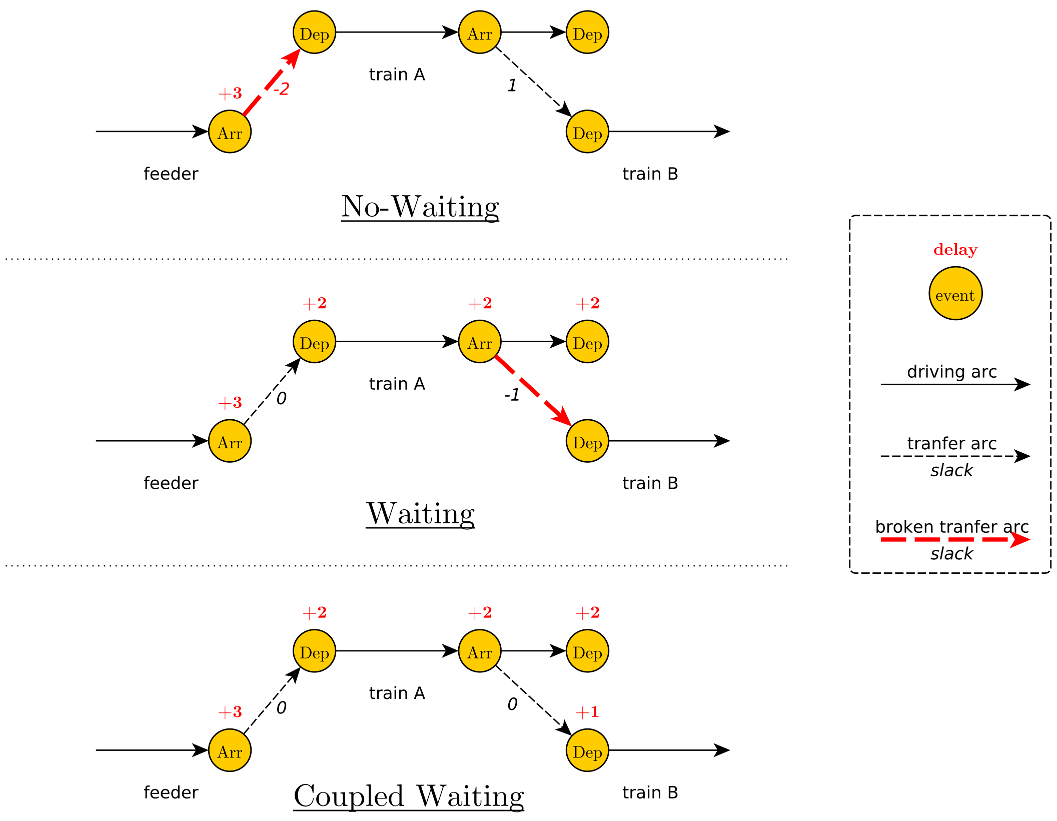


Fig. 7.2: Illustration of a *coupled decision* involving three trains. (Top) No-Waiting case: The first train is three minutes late, resulting in a transfer lacking two minutes in slack. (Middle) Waiting case: Delaying the departure of $trainA$ causes an additional broken transfer between $trainA$ and $trainB$. (Bottom) Coupled Waiting case: Both the initial and resulting transfers are kept.

schedules of the long-distance trains of the Deutsche Bahn have several time supplements applied to dwell times. We investigate the benefits of a combination of time supplement and waiting time rules in Section 8.4.8.

On the other hand, there are two main problems with the application of waiting time rules during propagation. The first problem is the correct usage of special cases. Waiting time rules can either be defined for general train classes or special train specific connections. Special train specific waiting time rules can cause delays of up to 15 minutes during daytime. Special trains that run during the nighttime can have even longer waiting time rules. The application of a general waiting time rule to a pair of trains having no planned interchanges can also be harmful. The second danger in the use of waiting time rules is the correct communication of their implementation to a dispatcher as a user of the PANDA GUI. These dangers can lead to the following problems:

- the dispatcher does not recognize his demand for action anymore

- the dispatcher is confused by a big visual difference between the situation displayed in PANDA and other used software
- the dispatcher acts in a way that decisions lead to loss of connections supposed to be kept by the waiting time rules

The most significant danger of waiting time rules, in general, is that their application can lead to worse performance in passenger delay. Because of those drawbacks of the waiting time rules, they are disabled during the experiments conducted in this chapter. We discuss the effects of waiting times rules in combination with *coupled decisions* in Section 7.5 in detail. In the next section, we present the concept of how coupled waiting can create a whole substructure of problems.

7.3 The Conflict Tree

In the previous example of Figure 7.2, there was only one additionally broken transfer. What if there is more than one additionally broken transfer and if these trains will cause further broken transfer when delayed? To simplify *coupled decision* involving more than two trains, we introduce the concept of a conflict tree in the context of train dispatching [LMHR16]. Let us suppose that an additional conflict results during the first waiting scenario. For this transfer tr and its delay propagation, we decide *waiting* recursively for every decision. Now we display every waiting train as a node of a new graph. We create a node and edges in this graph whenever a train is causing the delay of the target train. Due to its construction, this graph has a tree structure and will, therefore, be called the conflict tree having the first delayed train as root. Note that the node for delaying a train x by i minutes can appear multiple times in this structure. This is because the corresponding scenarios to reach this node are different. This *conflict tree* is denoted as $T_{tr} = (V(T_{tr}), E(T_{tr}))$.

Always waiting, however, will not be the best strategy in most cases. The optimal solution will consist of a wait decision for a subset of trains of the conflict tree. The resulting subgraph of the decision tree has to be either empty or still retain a tree-structure with the initially delayed train as root. Any other subset of delayed trains would imply that a train was delayed without an introduced conflict. The number of simulations necessary to cover every possible scenario is the number of valid subsets of trains.

Conflict trees may have a self-similarity or fractal property, as shown in Figure 7.3. Self-similarity/fractal means that subtrees and their associated transfers are similar to other subtrees. In the example presented in Figure 7.3 the subtree of vertex $v2$ is similar to the subtree of vertex $v1$ (without $v2$). However, the necessary delays in maintaining individual transfers can be different because of the different delay situation in both subtrees. For instance, $v3$ and $v4$ are *critical/broken* transfers by spreading the delay $d(v1)$ into the network. Nevertheless, $v3'$ and $v4'$ are the same *critical/broken* transfers, but by spreading the delays $d(v1)$ and then $d(v2)$ into the network. For a large-scale network like that of Germany, this property could lead to large conflict trees with over several thousand vertices.

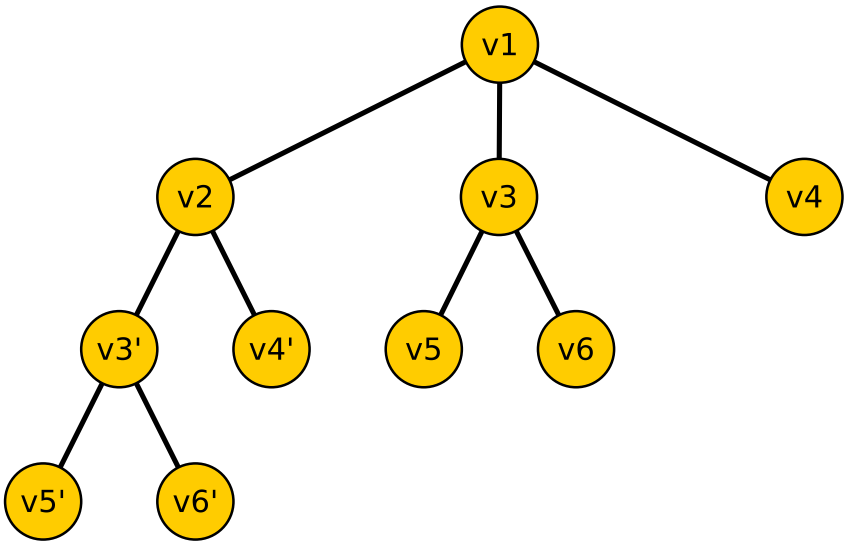


Fig. 7.3: An example of a conflict tree with a self-similarity property. The subtree of vertex $v2$ is similar to the subtree of vertex $v1$ (without $v2$). The vertices $v3$ to $v6$ and $v3'$ to $v6'$ correspond to the same transfers, but the necessary delays to maintain the transfers might be different [LMHR16].

Construction of a Conflict Tree

The algorithm to create a conflict tree is similar to a breadth-first search. We start with an empty queue $\mathcal{Q}$ and push the root vertex with the initial *broken* transfer into it. As long as there are vertices in the queue, we explore its first element (and then perform a dequeue operation). The current vertex v has to be maintained by propagating an artificial delay $d(v)$ in the underlying network $\mathcal{N}$. All thereby induced *critical/broken* transfers are collected and then inserted into the tree and queue. Let us consider the example given in Figure 7.3. We start with vertex $v1$ and spread the delay $d(v1)$ into the network $\mathcal{N}$. Then, we collect the induced conflicts $v2, v3$, and $v4$. Next, we continue with vertex $v2$ and spread the delay $d(v2)$ into the network. We also collect the subsequent conflicts $v3'$ and $v4'$. Now we would like to process $v3$, but the two delays $d(v1)$ and $d(v2)$ modify the current state of the network. The delay $d(v2)$ is independent from the measuring of the impact of delay $d(v3)$. Therefore, we have to re-establish a valid state of the network before spreading the delay of the current vertex. For simplicity, we remove *all* artificial delays directly after we inserted the induced subsequent *critical/broken* transfers into the queue. Then we can re-propagate all artificial delays from the root vertex to the predecessor of the current vertex v directly before spreading the delay. Thereby, we always ensure that the current state of the underlying network is valid. We implemented Algorithm 2 and added it to the PANDA framework.

The next section focuses on properties and the evaluation of the conflict tree.

Algorithm 2: Creating a conflict tree

```
Data: Queue of delays: $\mathcal{Q} = \{d_1\}$
Result: Complete tree of delays $T$
$T = \{d_1\}$;
while $\mathcal{Q}! = \emptyset$ do
    enable_undo_handler();
    $d_i = \mathcal{Q}$.pop();
    forall nodes $d_k$ in path $d_1 ... parent(d_i)$ in $T$ do
        apply_delay($d_k$) ;
    $C_{base} \leftarrow$ { all delays to keep critical transfers of affected groups} ;
    apply_delay($d_i$) ;
    $C_{final} \leftarrow$ { all delays to keep critical transfers of affected groups} ;
    forall $d_k$ in $C_{final} \setminus C_{base}$ do
        $\mathcal{Q}$.push($d_k$);
        $d_i$.add_child($d_k$);
    undo_all_delays();
return $T$
```

7.4 Evaluation of a Conflict Tree

For each vertex in the tree, we introduce a binary decision variable that this transfer will be maintained (one) or not (zero). We can interpret these $|V = V(T_{tr})|$ many binary decision variables as a $|V|$ bit long variable x. This variable x can theoretically attain $2^{|V|}$ different values, but not all of these values are feasible, as was explained in Section 7.3. Once we created a complete conflict, we calculate the real number of possible scenarios with the following formula applied the to root vertex:

$$f(v_{node}) = \begin{cases} 2 & \text{if } v_{node} \text{ is a leaf} \\ 1 + \prod\limits_{v_{child}} f(v_{child}) & \text{if } v_{node} \text{ is a inner node} \end{cases}$$

This recursive formula gives the number of conflicts resulting in a delay of the current node. The first case is that the delay will result in no further conflicts. Therefore there are two distinct possibilities in not dispatching and dispatching. In the second case where the decision in the current node has further conflicts, we count one for the possibility of not dispatching with no conflicts plus the product of all (independent) combinations in dispatching child nodes of the resulting conflicts.

The evaluation algorithm has two phases. In the first phase, the framework determines the set of all affected passenger groups. All feasible *coupled decisions* are simulated successively to achieve this. Working with a collection of all affected passengers is necessary to have an unbiased comparison between the impact of all simulated feasible decisions on the passenger flow.

In the second phase, the framework iterates over all *coupled decisions* successively. In each step, the impact of the injected delays on the previously collected passenger groups is measured and stored. Passenger groups may have to be rerouted at this point if their route is not feasible anymore. After these two steps, we compare all feasible *coupled decisions* in an unbiased way. Finally, we can evaluate the impact on the passenger flow of all *coupled decisions*. For each scenario, we compute objective values as described in Section 6.2. We can compare two solution vectors by counting the number of criteria where one solution is strictly better than the other. Hence, scenario A is considered as better as scenario B if the majority of criteria is in favor of scenario A.

Now we introduce experiments to verify if *coupled decisions* have an impact on realistic situations.

7.4.1 Experimental Setup

For this experiment, the simulation is also made 15 minutes before the departure of the first connecting train of the analyzed critical transfer. Determining the optimal solution for conflict trees with a large number of vertices/conflicts requires a high computational effort. If a conflict tree is very large, executing the delay of the source train is typically the inferior decision. A later result (see Figure 7.5) shows that the average probability for benefits of a decision decrease as the size of the tree grows. After a detailed investigation of several large trees, we decided to limit our simulation to trees with less than ten vertices/conflicts. Finally, we collect all evaluations and compare them with pure NO-WAIT and WAIT decisions. Overall as many as 20'920 different conflict trees were analyzed during our experiments.

7.4.2 Experimental Results

Among these 20920 conflict trees, the result showed 4941 cases (about 23.61%), where coupled waiting decisions are better than single WAIT-decisions. Furthermore, there are 2982 cases (about 14.25%) in which the combined waiting choices are better than NO-WAIT-decisions. However, there are 1319 cases (approximately 6.3%) where coupled waiting decisions are better than both WAIT- and NO-WAIT-decisions. Finding evidence for the hypothesis that *coupled decisions* could lead to an improvement in the quality if dispatching assistance software can, therefore, be seen as a success.

Those discovered cases can be used to understand under which circumstances *coupled decisions* are preferable. For this purpose, we looked at cases where coupled waiting decisions are at least better than WAIT- or NO-WAIT-decisions. Of those cases, we collected all maintained non-root vertices. Similarly, we also accumulate all not maintained non-root vertices of all remaining scenarios in a second set. For both sets of vertices, we consider several properties of its members. These properties are, for instance: the number of minutes required to maintain the corresponding transfer and the number of its children in the conflict tree.

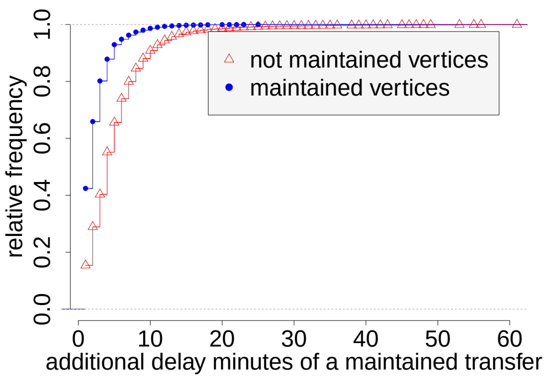


Fig. 7.4: Cumulative distribution functions of the additional delay minutes for both maintained and not maintained vertices of all conflict trees [LMHR16].

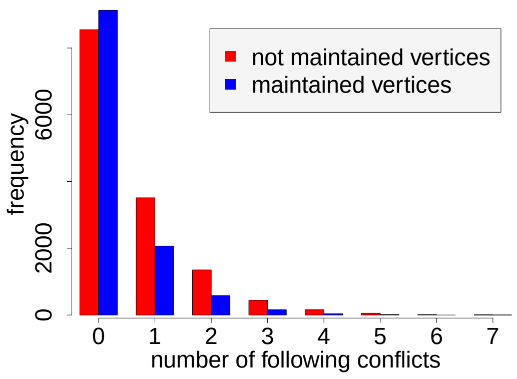


Fig. 7.5: Distribution function of the number of follow-up conflicts for both maintained and not maintained vertices of all conflict trees. Vertices with zero conflicts are leave nodes in the decision tree [LMHR16].

Tab. 7.1: The average change for three criteria by applying coupled waiting decisions in comparison with standard single WAIT/NO-WAIT decisions.

criteria	benefit of coupled waiting decisions
total arrival delay	-3.02%
# passengers with $\geq$ 60 min. delay	2.34%
# passengers with $\geq$ 120 min. delay	.58%

Figure 7.4 shows that it is more likely to have a kept transfer if the additional delay minutes are quite small. Also, Figure 7.5 reveals that it is more probable for a vertex to become a maintained transfer if it is a leaf in the decision tree having zero conflicts. If a vertex has at least one child, it is about 15% more likely to be a non-maintained transfer. We could create a heuristic pruning scheme making use of these likelihoods. By neglecting decisions where follow-up conflicts introduce larger extra delays, computational time would be reduced drastically. This strategy also has a low probability of losing good solutions. Finally, we want to quantify the benefit of *coupled decisions*. To measure this benefit, we compare the standard single WAIT/NO-WAIT decisions with the best solution (for either WAIT, NO-WAIT, or a *coupled decision* according to three different criteria). Now we can measure the gap between the traditional and improved strategy. Improvement found by using the new approach often contains less substantial arrival delays at the costs of creating many smaller delays. These cases are the reason why in Table 7.1, the coupled waiting decisions have slightly worsened the total arrival delay by about 3%. Nevertheless, the number of passengers with an arrival delay of at least 60 or at least 120 minutes could be reduced by approximately 2%, respectively, by .58%. Thus, the overall benefit of *coupled decisions* is mixed, but the improvements for passengers with massive delays should outweigh their slightly more significant average delay.

7.5 Summary and Future Work

In this chapter, we investigated the possibilities of a *coupled decision*. In contrast to the classical paradigm of deciding for one departure delay of the connecting train, this method handles resulting conflicts. A decision tree represented the chains of decisions is created. A case study using more than 20k decision trees revealed a significant fraction of choices where passengers benefit from *coupled decisions*. However, the computational effort for exploring and simulating *coupled decision* is relatively high rendering the method (at least currently) unusable in live dispatching assistance using PANDA.

We neglected *waiting time rules* in those experiments. The use of *waiting time rules* can result in resolving several conflicts heuristically. The application of those rules will contract every leaf of the conflict tree created by a smaller delay. Generally, the use of short waiting times rules leads to the following changes in passenger delay. Minor delays are increased significantly because of a significant increase in knock-on delays. Passengers with small slacks planned on their connections (Figure 7.1) benefit from more robust interchanges. Waiting time rules can also be combined with *coupled decisions*. An exploration of this topic could lead to hybrid strategies using a set of waiting times rules. Such rules (smaller than the original ones) can leave room for optimization while improving computational effort to a point where a live calculation is possible.

Other improvements can also be made by exploiting the fact that large conflict trees often result in worse passenger delays. A deeper understanding of unfavorable attributes for waiting decision can lead to a set of useful rules for a pruning heuristic of conflict trees.

In part II, we could contribute to the practical application in delay management. We investigated several noteworthy aspects of the affects disruption management and their communication to the passenger have.

Part III

Robustness in Public Transport

8 Robustness Tests for Public Transport Planning

We learn from failure, not from success!

— **Bram Stoker, Dracula** [Sto97]

8.1 Introduction

One major challenge in public transport planning is establishing reliable services. Services can be fast and cheap, but reliability is also essential. When a passenger has to arrive before a deadline, he will choose another means of transport if the provided service is not reliable enough. If other means of transport are not an option earlier services than theoretically necessary have to be chosen, thus spending more time to commute. In a survey from 2017, 41% of Germans, commuting by car stated that they would consider using public transport if only reliability were not so bad. In this survey, reliability ranked second after ticket-price outranking speed and a smaller number of interchanges [Sta17].

Public transport planners have a hard time improving the reliability of their services. In cities of the size of Karlsruhe (300k inhabitants) planners manually add buffer times to driving edges, that often encounter delays. This practice, however, is not compatible with high levels of optimization as done in the field of integrated line planning and timetabling (see Section 2.2). Integrated line-planing and timetabling can improve the average traveling times of passengers and cost for the service provider significantly. The problem with these methods, however, is that they do not optimize their solution for robustness. Using the term robustness in the context of optimization, yet, can easily be confused with *robust timetable optimization.* The next section will focus on a clear differentiation between both terms.

The Term Robustness in *Robust Timetable Optimization*

There are several types of robustness to express defined aspects of a solution. Strict robustness, for example, is one of those robustness types introduced by Soyster in [LS73]. Under his definition, a timetable is strictly robust if the execution is feasible under all possible scenarios. Timetables designed with this type of robustness typically have very long driving times due to extreme buffers. Liebchen et al. introduced a more practically oriented robustness type; the *r*ecoverable robustness [Lie+09]. Here changes in timetable (by delays or otherwise) are allowed as long as feasibility can be restored afterwards. Goerigk and Schöbel conclude in [GS10], that it is crucial for the application of these robustness concepts that the uncertainty set defining possible delays has to be known. As a consequence of the given definitions, the robustness concept of a timetable can only be deduced with the uncertainty set used to construct the solution.

robust timetable optimization The creation of a timetable, that is able to recover the delays of vehicles according to defined requirements.

robustness The quality of a timetable, that delays introduced to vehicles have a low impact on the arrival delay of passengers measured in arrival delay or lost utility for all passengers.

The notion of robustness used in this research is an empiric measurement to compare different timetables rather than formal classification. Real-world scenarios do not follow all feasibility constraints many models enforce during computation. The breakdown of a vehicle alone at any point in time challenges those models. Then the execution of a schedule often is no longer possible.

We consider a timetable as robust if it can handle disturbances with little loss in quality of service (compared to alternatives). The field of improving the robustness of public transport has many aspects. In the next section, we set goals for the research done concerning robustness.

Goals of this research

In the context of FOR 2083, there has been a challenge to understand, create, and evaluate line plans and timetables for a given network. The most frequently studied of those networks was the grid-network shown in Figure 3.5. Several approaches were made manually creating and generating line plans and schedules. Computer scientists took a different approach. First they created solutions that were the fastest in terms of perceived journey time. Later they were also able to generate the solution with minimal cost, satisfying all constraints made to a solution for a schedule [PSS18]. They also created many sets of optimized solutions based on manually created line-plans as well as heuristically generated ones [Pät+17]. Several solutions had a good compromise between cost and perceived journey time outperforming manual solutions. There still were remaining questions. How reliable were these plans in terms of robustness? If reliability were measurable, can we create more robust schedules? Creating a schedule that is also robust, presents an entirely different challenge. This is because there is no formal definition of the term robustness, which is suitable for practical application. We suggest using a set of generic simulations containing common scenarios found in most public transport systems. After these simulations, the performance of each instance should be evaluated using passenger delays at their final destination, and several other indicators as robustness measurement. More specifically, these measurements should serve multiple purposes in the context of comparing different plans of the same infrastructure network.

- being suitable to replicate the expected shortcomings of a plan,
- highlight differences in the choice of the line network,
- observing trade-offs between robustness and travel times based on timetables which **LinTim** optimized for the respective line plan,

- identifying the best methods for robust timetable creation.

In addition to these purposes, there are several other properties robustness test should have.

- The robustness tests shall apply to all possible line networks and corresponding timetables in public transport and for general demand patterns varying within a day.
- There should be multiple scenarios/test each potentially capturing different strengths and weaknesses of a given schedule.
- The tests support parametrization for changing delays.
- The tests should work under different assumptions for passenger behaviour.

There have been several works of research influencing this investigation. The next section will highlight the ones of most significant importance.

Related Work

The study of punctuality and its effect on passengers has always been of high importance. Reports often provide the percentage of services that arrive on time [Ver22]. *Being on time*, however, is defined as to arrive not later than within a given margin (e.g., 5 or 15 minutes for long-distance trains) of the planned arrival time. More recent publications often use metrics measuring the dissatisfaction of passengers. In a Dagstuhl seminar in 2016 [Dag16], Dennis Huisman coined the phrase "*passenger punctuality 2.0*" for measuring the (weighted) total passenger delay at the destination for all passengers. [Sch01; Sch06] uses the latter as well as [HGL08] and others. Less sophisticated indicators include the mere number of delayed departure and arrival events [Cic+09]. Alternatively, Acuna-Agost et al. [AA+11] propose to count every time unit of delay at every planned stop and the last stop. In the PANDA software framework used in Part II, dispatchers relied on viewing several KPIs measuring different aspects of passenger discomfort. Complex cost functions resulting in one numeric value for satisfaction/dissatisfaction have several advantages when comparing situations. The drawback is that they are hard to understand/interpret for operators and even for planners. Now we look at robust timetabling from an operator's perspective.

In this setting, it is desirable that a timetable can absorb delays and recover quickly (thus avoiding penalties for the operator). To this end, inserting buffer times in the schedule may help to reduce the effect of disturbances, but may harm the realized travel times. Not only the total amount of buffer times but also their distribution along the lines is essential. These aspects have intensively been studied in operations research. For example, Kroon et al. [Kro+08] use stochastic optimization to allocate time supplements to make the timetable maximally robust against stochastic disturbances. They use the expected weighted delay of the trains as an indicator. Using mixed-integer linear programming, Sels et al. [Sel+16] improve punctuality for passenger trains in Belgium by minimizing the total passenger travel time as expected in practice. Bešinović et al. [Bes+16] optimized the trade-off between minimal travel times and maximal robustness using integer linear programming. Their formulation includes a measure for delay recovery computed by an integrated delay propagation model in a Monte Carlo setting. In these works, the line network is usually already fixed. Robustness of timetables was empirically investigated (concerning different robustness concepts) in [GS10]. [GSS13] studied the robustness of lines. [Sch12] provides a general survey of line planning in public transport. A recent integrated approach combines line planning and timetabling, but without considering robustness [Sch17].

Our work proposes to create a way of analyzing more than one aspect for a more robust public transport system. At least the issue of passenger satisfaction and features relevant for operation shall be contained and analyzed. To achieve our goal, we suggest the combination of integrated line-planning and timetabling, creating multiple instances. Subsequently, we run robustness tests evaluating every single instance. These robustness tests have been introduced by us during the work on [Fri+17b]. This chapter contains our methods and findings.

8.2 Robustness Tests

A test is defined as a challenge or trial subjects have to undergo to prove that they fulfill their purpose up to a defined extent. In software engineering, for example, methods and programs are often tested to an extent where every input has to result in a correct output or error message. Material products, on the other hand, are tested to ensure the safety of the product as well as establishing that they provide the proclaimed purpose. Testing the output of an optimization problem concerning robustness is somewhat unusual. We see robustness as a problem that optimization should solve. There are however two significant reasons for doing otherwise.

The first reason is that the sheer complexity of this task concerning computational time. The task of integrated line planning and timetabling is already complex and reaches its limits when computing medium-sized networks. While larger-scale networks are still solvable, methods rely on heuristics in the process. The second and more important reason is the fact that these robustness tests can be of immense value even if optimization, including robustness, would be possible. In this hypothetical case, planners would need to specify a specific set of parameters for the optimization. A produced solution can make the binary claim to fulfill all of these parameters. The planner will face many problems with this approach. Specifying the parameters can be extremely hard, especially when creating a network from scratch. The binary answer of optimality does not provide any insights about the performance under certain conditions or certain types of disruptions. While a robustness test itself will not influence a given solution, it will be able to produce advanced metrics describing and evaluating solutions in great detail.

Now that we established the inherent advantages of this method, single tests need to be specified that can indeed produce different metrics for defined types of disruptions. In [Fri+17b; Fri+18], we created four types of robustness tests for a later evaluation of instances. We now present these tests with their description, motivation, and possible identifications of attributes.

Robustness Test 1 (RT-1): Delay of Single Service Runs

Real world motivation: Initial delays of vehicles are very common. Long-distance trains in Germany, for example, depart with 58% chance of at least one minute delay (see Section 8.4.7). Initial delays can be caused by late drivers or technical problems with vehicles or railway infrastructure (see Figure 8.1).

Description: This robustness test is delaying a single service run at its beginning. For every trip contained in the schedule, the first departure event is postponed by x_1 minutes. We simulate the whole day with only this delay and its effects on passengers. If the schedule contains n trips, we end up with n sets of indicators for this robustness test.

Possible identification of:

- trips containing insufficient buffer times
- whole day delays in case missing buffer times paired with short/no turnaround times
- significant changes in arrival delay due to over-optimization of transfers

Fig. 8.1: Employees of Deutsche Bahn fixing a railway switch near the depot. This operation will most likely result in a starting delay of trains. Image [Cc0b]

Robustness Test 2 (RT-2): Slow-Down of Single Network Sections

Real world motivation: Construction work on the infrastructure that might cause delays are common (see Figure 8.2). Especially when construction work is only during a short period of days, planners often not create special schedules. There can be multiple other causes for a driving-speed limitation due to road/track conditions during one or more days.

Description: This robustness test is selecting one directed edge of the underlying infrastructure network. If, during the simulation of one day, any service is using this edge of the system, the next arrival is delayed by x_2 minutes. If the schedule contains n directed edges of the infrastructure network, we end up with n sets of indicators for this robustness test.

Fig. 8.2: Construction work slowing regular speed on a track. Image [Cc0a]

Possible identification of:

- build-up of large delay of a vehicle operating multiple times on the same infrastructure
- bottleneck of capacities when multiple vehicles have delays in the same area of the network
- significant changes in arrival delay due to over-optimization of transfers

Robustness Test 3 (RT-3): Temporary Blocking of Single Stop

Real world motivation: Sometimes there is significant disruption at a station because of the breakdown of specific infrastructure. A power outage may cause this. Other events can have a similar effect. The police might have to investigate unattended luggage or block the only road exiting the stop. Such measures can lead to vehicles queuing for entry and also for exiting the stop (see Figure 8.3).

Description: The temporary blocking of a complete stop. The defining input parameter for this experiment is the duration x_3 of the blocking in minutes. This robustness test is selecting one of n stops included in the schedule. At 05:00, this station will be blocked for at least x_3 minutes. In this frame of time, arriving vehicles will be delayed until they are allowed to depart from this stop. The first vehicle to arrive during this blocking will be the first vehicle allowed to depart. Vehicles will then be allowed to depart using a headway of $t^{headway}$ minutes. As long as there are vehicles marked as blocked and not having been departed yet, the blocking continues. To have an interval of duration x_3 to cover every time of one day this experiment is conducted once for every period from 05:00 to 22:00. The intervals tested this way are $[05:00 + \delta \cdot x_3; 05:00 + (\delta + 1) \cdot x_3]$ $\forall \delta \in [0; \frac{22:00-05:00}{x_3}]$.

Fig. 8.3: Problem inside the station leading to lines of waiting trains. Image [Cc0c]

Possible identification of:

- a station of crucial importance at all times
- bottleneck of capacities when blocking a station for a certain period
- too closely planned departures in a short period leading to a prolonged disruption

Robustness Test 4 (RT-4): Historical Delay Distribution

Real world motivation: Random delays based on empirically observed delay distributions are best suited to replicate the unforeseen delays in a real-world setting (see Figure 8.4).

Description: We simulate a day of travel, introducing random delays based on real live observations. Because one single simulation may not be representative for an average day, we repeat this process multiple times. Every day will produce different sets of indicators. We store the mean value for each indicator for this robustness test.

Possible identification of :

- expected average delay on a real-world day
- variance in delay indicators of different days

In contrast to other synthetical tests, the results in this robustness test can be used to make significant predictions for the quality of service in a given network. There are multiple ways of modeling random delays. [Ber+11b] uses discrete delay distributions and focus on the aspect of the propagation of the delay rather than the occurrence. Other researchers use neural networks for future delay prediction [YKS13]. There is a good reason for using discrete distributions. The most important reason is that a delay can have several different reasons, all of which may have different distributions. We later (in Section 8.3.6) prove the influence of crowding as one reason.

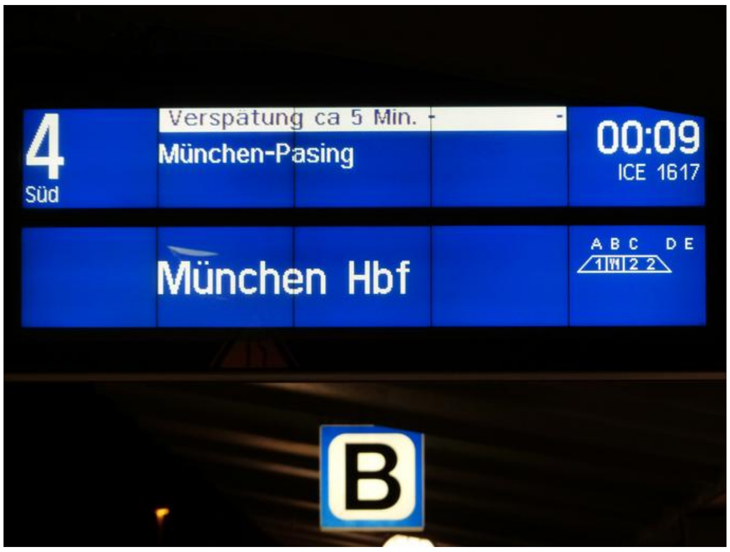


Fig. 8.4: regular delay information in Germany. Image [Cc0c]

The model for generating delays can be more detailed if the network and schedule for simulation resemble the network and schedules with delays observed in the real-wold. If the same schedule has real-world delay data, every event could have its delay distribution. In robustness tests where the timetable is new, we have to use a more general model for generating delays. If infrastructural data is available, every network edge should have different delay distributions, possibly based on the time of day it is accessed. Because of the lack of infrastructural data of the Deutsche Bahn network our model does not capture all aspects influencing delay distributions. When learning delay distributions from real-world delays, it is also essential that a suspension that is caused by dispatchers is not counted/determined as a random delay. The latter would overestimate random delays significantly. The model used for the experiments conducted in this research is explained in detail in Section 8.4.7. Improving the model of how passengers behave is as important as modeling realistic disruption themselves. The next section will focus on this topic to improve previous work inside the PANDA framework.

8.3 Improvements in the Passenger Behavior Model

Passenger behavior is taking a significant part in influencing the results of a simulation. An unrealistic passenger behavior model can compromise the decisions made by the train dispatchers. If, for example, passengers supposed to be on an interchange unanimously decided to board another train instead of the feeder, recommendations will be different. This section will go into detail about what the passenger knows about the system and what his or her options and preferences are.

8.3.1 Degree of Information for the Passenger

Existing models of passengers focus on differing degrees of knowledge about their journey. One extreme would be that the passenger knows nothing about any vehicles and even the schedule, and a central system tells him every action. Another extreme is that the passenger knows everything about the current system, historical data, and prediction probabilities about the future. Both extremes are, however, unrealistic for most settings.

In a realistic environment, a passenger is not very well informed. Let us consider the knowledge of long-distance travelers in Germany. Most passengers only receive information when they actively request a status update via the DB-Navigator App. Another possibility is that there is a service announcement informing the passenger. In both cases, the passengers only receive a curated subset of the information provided by the Deutsche Bahn.

It is essential to define the degree of information known to passengers in the simulations because it has a significant impact on their behavior.

Information possibly available to the passenger:

- current delay of arriving/departing vehicles at the current stop
- current delay of all vehicles
- estimation of future delays (from experience)
- delay predictions of vehicles (from our system)
- decisions if their future transfer will be kept
- current and past rates of vehicle occupation
- expected future rates of vehicle occupation

Information available to passengers in our model:

- current delay of arriving/departing vehicles at the current stop
- current delay of all vehicles
- delay predictions of all vehicles (from our system)

Availability of this information has a crucial impact on the choice of route of a passenger. Leng and Corman showed with a detailed agent-based model simulating delays near Zürich that gives agents different degrees of knowledge about delays [LC20]. The experiments showed the severity of having no information in comparison to *timely* and *advance* information. They show a significantly higher amount of delays when passengers are uninformed. The degree of information used by our passengers is similar to their *timely informed* model. This implies that the majority of arrival-delays produced during rerouting decisions are not caused by poorly informed agents, but the result of the delay robustness of the network and available alternatives. The next sections will expand on the route choice and how the model can influence it.

8.3.2 Available Rerouting Possibilities for the Passenger

Whenever delays occur, a passenger may change their route. This operation, however, can have several restrictions. One essential limitation is that the new route has to be compatible with a ticket bought by the passenger. In German railways, this means that certain trains of a higher category are forbidden. Some tickets even restrict the passenger to the use of one designated connection. Only if this route is no longer possible, passengers can use other connections without extra cost.

The model defines the time when a passenger is allowed to change his/her route inside the simulation. Most models let the passenger either change his/her path at every boarding event or only in case of a lost connection. For an optimal solution during disruptions, it could be beneficial for every passenger to change his/her journey at every update. This practice would result in a massive amount of routing requests. The resulting routes may not always be of use to the passenger. Some routes may be no longer optimal by the time the passenger reaches the next station where an interchange is taking place. Therefore, the number of possible requests for an alternative route needs to be limited to a realistic amount. In a previous version of the MOTIS/PANDA software, the request could only be made after a broken interchange or during a simulation. During the work with robustness analysis, we improved the results of a routing request to include more realistic scenarios. The actual amount of possible re-routings, however, also depends on the degree of information available to the passenger. For example, a passenger will not reroute to a connection where he will not be able to board a train because the latter has no remaining capacity.

Passengers are now able to change their route whenever:

- they have not yet started their journey, but observe that the intended first vehicle is delayed
- they are interchanging between trains and the connecting train is not on time
- they are currently arriving at some stop while knowing that they have a broken transfer in the future
- they are attempting to enter a vehicle that is currently full (after Section 8.4.2)

We have now covered the influence of the degree of information and modeling of the system. In the next section we will expand on subjective preferences of the passengers and its impact on route choice.

8.3.3 Preferences of the Passenger

We already covered the question when a passenger can switch to what kind of other routes. The final aspect of the rerouting process is the route choice. In large networks, there are often many routes available that will get passengers from origin to their destination. In previous works, the routing process consisted of listing all Pareto-optimal routes (see Section 2.3) and then selecting from all connection with minimal travel time one with the smallest number of interchanges. For passenger information, the listing of different routes is essential. In a simulation of passengers with defined preferences, however, there is no need for the algorithm to produce more than one path if this route maximizes the passenger's satisfaction. In the new framework, we switched back to a one-criteria Dijkstra algorithm, but with an advanced passenger satisfaction function. Now the objective can be specified to put weights on travel time, interchanges and occupation penalties. [MS98] uses such an approach first in the context of public transport. We chose this method for its inherent simplicity in implementation as well as extensibility. One particular feature of our model is the existence of capacities. While capacities are crucial in every step of the planning process, delay management often neglects them in online systems. With a new rise in demand for public transport, more and more systems are using capacities. The next section will introduce the different terms and concepts needed for simulations, which include capacity limitations.

8.3.4 Introduction of (Hard) Capacities

Even though this dissertation focused on the simulation of passengers during their journey, it is crucial to investigate the concept of capacities from different aspects of research in public transportation. Traditional research in computer science often focuses on the task of routing all passengers using valid capacities. A practitioner in planning is concerned with quality control and will focus on the task of balancing demand and capacities. No vehicle in the public transport plan should exceed their seating capacity frequently. Both parties may use different terms when talking about capacities:

seat capacitiy The number of seats available inside a vehicle.

nominal capacity The number of passengers expected before crowding is experienced (in long-distance public transport equal to seat capacities).

maximum/hard capacity This number of passengers cannot be exceeded in practice for safety reasons.

Using capacities introduces two significant challenges during the simulation process. The first one (and more challenging one for a computer scientist) is that any flow of passengers simulated has to satisfy the hard capacity limitations. The second challenge is to include the rate of occupation of vehicles in passenger satisfaction. Especially on long journeys, the experienced quality of service can differ strongly. A journey is different in a train where most seats are free in contrast to a trip where there are no unoccupied seats available. The model here has to be appropriate for the specific public transport system. While in an urban underground network standing for a small number of stops is a regular occurrence, this is not acceptable in the long-distance railway or busses. The first step for the simulation of passengers traveling in a network, including capacities, is defining a set of rules/constraints. Here is the set of rules used in the following implementations:

1. a passenger having boarded a vehicle will remain seated if he/she plans to continue his journey in this vehicle
2. before any boarding takes place all passengers leaving the vehicle are removed
3. the remaining capacity of this train is filled with passengers randomly selected from those who plan boarding the vehicle
4. once the capacity limit is reached, all remaining passengers are rerouted.

8.3.5 Crowding as a Measurable Source of Passenger Discomfort

Several approaches include the occupation of an edge or trip into a utility/satisfaction function for the passenger. Other forms of public transport can neglect such endeavors. Flights, long-distance buses, or some long-distance railway lines like the Japanese Shinkansen do not need any such function because standing is not permitted. A vehicle where all passengers are standing, on the other hand, is often modeled using a nonlinear function connected to passenger density measured in passengers per m^2 [HK15]. Distinct models can be more specialized. In the German long-distance railway, passengers were used to search for free seats quite frequently. Trains had an average occupation of 42.1% in 2007. Since then, the average occupation steadily increased to 56.1% in 2018 [sta17]. During the early phase of the COVID-19 pandemic the average occupation dropped massively because the demand sank by 60% [Deu21]. However the trend for a greener mobilty using trains is continuing and the Deutsche Bahn is expecting higher demands and making profits again in 2022 [Foc21]. Because of these trends and the availability of tickets without a seat reservation, the problem of standing passengers becomes more frequent. When a long-distance train in Germany reaches a certain threshold above their nominal capacity, the train is not allowed to depart. Because of this situation, we decided to use a particular function to calculate discomfort in [MHRS19]. For any edge, a passenger either finds a double seat, single-seat, or is standing inside the train up to an occupational rate of below 120% of nominal capacity.

The crowding discomfort function currently available in our framework is seen in Figures 8.5. Until .65 relative occupation, there is no crowding penalty. Until every seat is taken there is only a small penalty of .2 per minute. For every passenger standing inside a vehicle, there is a cost of 1 additional unit per minute.

This concludes essential changes in the passenger-model and implementation of hard capacity constraints. In the next section we present an observation resulting in the use of capacities in our model.

8.3.6 Crowding as a Measurable Source of Primary Delay

Many situations in public transport can result in delays. When looking at delays as the manifestation of some problem in execution, the reason for a particular delay is often not distinguishable in the data. The distribution of primary delays (see Figure 8.39) is composed of delays created by a mixture of reasons. While a random delay is not predictable, risk factors

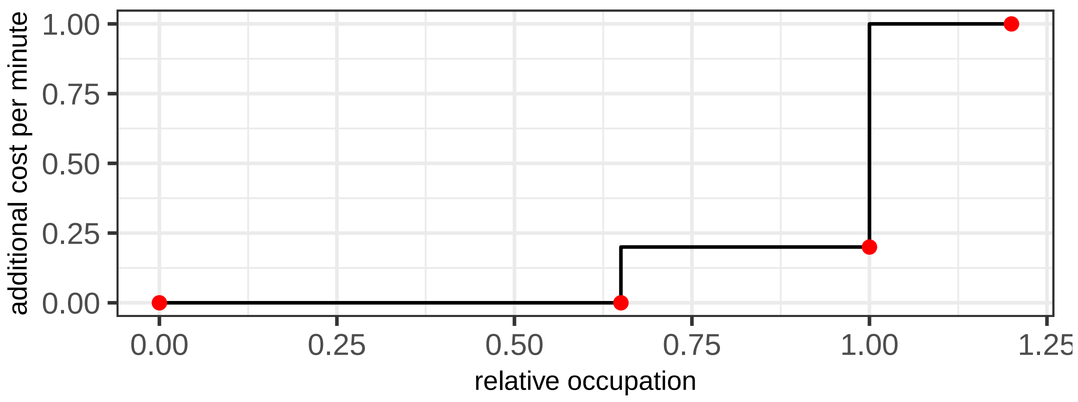


Fig. 8.5: Crowding cost function of our framework for passengers on long distance trains.

can be analyzed. Looking at data that correlates with delay and splitting the data into subsets can result in stochastically different delay distributions.

In [MHRS19], we define three classes of occupation of a train:

free the rate of occupation allows for any boarding passenger to sit in a free double seat.

crowded boarding passengers will have to take a seat next to a stranger.

standing boarding passengers have to stand.

The first question to answer is if there is any stochastical evidence that the primary delay distribution in category *free* is different from category *standing*. To test this, we assume that both distributions are equal and test the hypothesis using the Kolmogorov–Smirnov test. The data for this test was gathered using every departure event for German long-distance trains from May 2019, whenever capacity data was available to us. Using this data, we could measure 99'000 departure events. The Kolmogorov–Smirnov test performed using the R Software [Rc19] stated that the probability that both distributions are equal is $2.2 \cdot 10^{-16}$. This value is significant evidence that the rate of occupation has a measurable influence on the delay distribution.

We now compared some key features of both distributions. First, we looked at the mean values (see Figure 8.6). For a regular departure event, we can expect a primary delay of 25 seconds, when there is enough free capacity. The *crowded* and *standing* classes have a slightly larger expected primary delay. However small the difference is, this can accumulate over many stops, resulting in substantial delays. The second important metric for a delay distribution, is the rate of non-zero values. In other words, the probability of gaining any new delay. Because the delay distributions for initial departure and any departure of a trip are also different, we compared both to their counterparts when the rate of occupation reached *standing*. Figure 8.7 shows our results. There is a significant increase in the probability(+60%) of gaining a new delay when people are standing inside the long-distance trains. These two observations emphasize that we should avoid standing passengers inside trains. Decreasing the risk for delays is one of the top

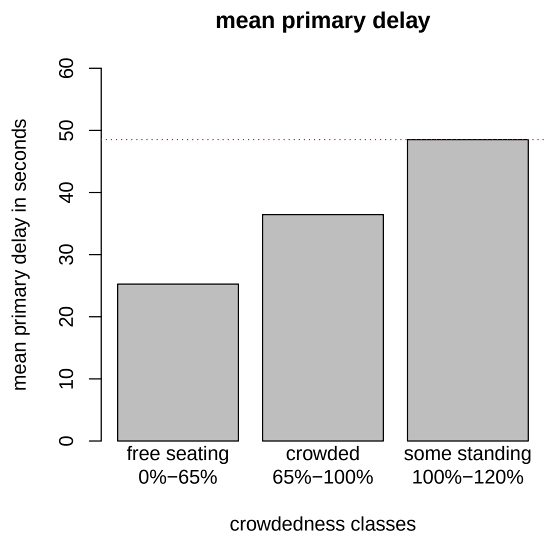


Fig. 8.6: Mean value of primary delay distribution for departure events. Split into capacities categories presented in [MHRS19]

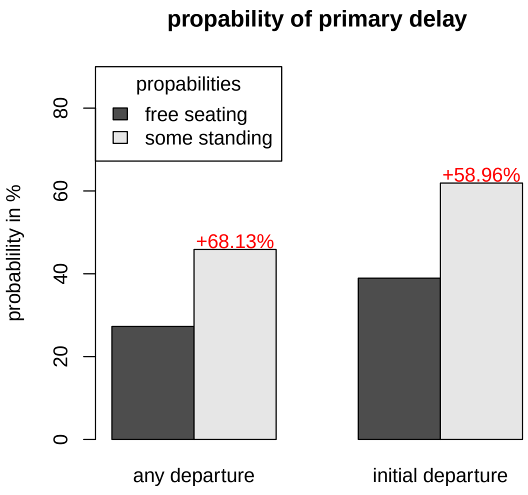


Fig. 8.7: Chance of acquiring any primary delay on departure.

priorities of public transport companies. Furthermore, when routing passengers in a simulation model, this should be heavily penalized. In [MHRS19] standing has double the cost than being seated within a vehicle with less than 65% of total capacity.

8.4 Experiments and Results

To test if the presented robustness tests can be used to gain the expected indicators for weaknesses of networks, we conducted several sets of experiments. In the first set of experiments, we used various line networks. These distinct line networks are likely to produce a different order of robustness for tests 1-3. In a later test, we used the same line network but different strategies for the application of time supplements. Those tests are more likely to produce results that maintain the order of measured robustness in most tests. The last set of experiments was conducted on a large long-distance network to analyze the difference in robustness when adding time-supplements to driving or dwelling times. The next section presents all the datasets in detail.

8.4.1 Experiment Set 1 - Different Line Plans on a Grid Network

As a well-studied model network, LinTim and planners created line plans and schedules for the grid-network (see 3.2.3) introduced in [Fri+17a]. These have been evaluated according to costs and travel time. An open question, however, was how those different line networks perform during disruptions. Ten different combinations of line plans and timetables were selected to perform robustness tests, answering this question. A planner created some of those line plans

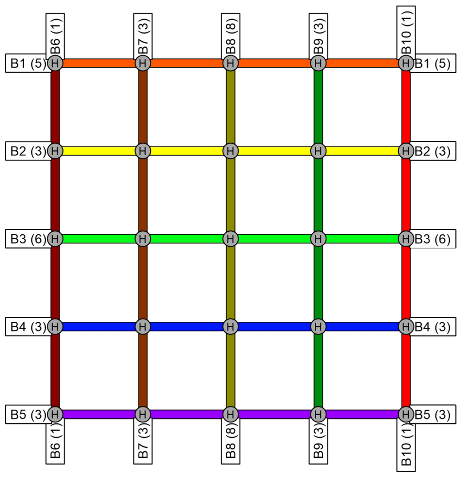


Fig. 8.8: Example 2 of lines in grid network (instance A_2_3)

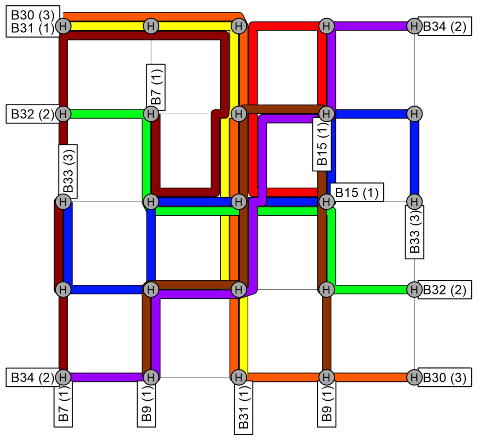


Fig. 8.9: Example 3 of lines in grid network(instance A_2_5)

like the one in Figure 3.5. Some line plans were chosen to represent simple concepts like the one shown in Figure 8.8 and others were generated using algorithms (see Figure 8.9) [Fri+17b]. As can be seen in Figure 8.8, the grid network has 25 stops connected with an infrastructure consisting of 40 edges. There is no fixed set of terminals for vehicles to return to after a day of traffic. The cost function only considers the used number of vehicles, driven kilometers, and empty kilometers. We neglect costs and schedules for the crew in this process. The twelve instances created will now be explained in detail.

The first instances called P_1 and P_2, depicted in Figure 3.5, are created by hand, using the experience of public transport planners. They are constructed using a system-wide frequency of 20 minutes. The other instances contain components which are automatically generated using routines of LinTim [Lin; GSS13]. LinTim optimized the lines and their frequencies concerning their costs. Timetables were optimized heuristically to minimize the traveling time of the passengers. The modulo simplex algorithm described in [GS13] was used while vehicle schedules are again optimized for their costs. For a clear notation, A_x_y denotes algorithmic solution y, based on the manual solution P_x.

A_x_1**:** The lines and frequencies are fixed as in the manual plan, the timetable and vehicle schedule are automatically optimized.

A_x_2**:** Only the lines are fixed as in the manual plan while their frequencies, the timetable, and the vehicle schedule are automatically optimized.

A_x_3**:** Here only vertical and horizontal lines were allowed. Based on these lines, the line plan, the frequencies, the timetable, and the vehicle schedules are automatically optimized.

A_x_4**:** Here, everything is calculated automatically, including the line pool which is generated by the algorithm presented in [GHS17].

A_x_5**:** Everything is calculated automatically but using a combination of the manual line pool and the line pool of A_x_4.

For each of those instances, we first routed passengers. Based on the demand model, every passenger has a station of origin, a target station, and an initial desired time of departure. The demand for the grid-instances was created by public transport planners for [Fri+17a] and a visualization can be seen in Figure 3.6. There are a set of OD matrices, that specifies different demands for each hour of the day. The relative utilization of vehicles can be seen in Figure 2.7. Routes are chosen using preferences of passengers described in Section 8.3.3. We selected a simple set of parameters for the passenger satisfaction function. The cost of driving and waiting are equal, and the costs for a transfer is worth five minutes of additional travel. We neglect other penalties. Our passenger satisfaction is normally measured using a cost-model where delay minutes is only one KPI. In the following evaluations, however, we only measure the total delay minutes. By doing so, we avoid complicated explanations in the later analysis. For example, a higher occupation in a scenario could be evaluated worse than larger delays with lower occupation.

Each simulation of the different robustness tests will start using these initial routes. These initial routes can be analyzed, revealing a fundamental difference in essential attributes. These features include the average undisturbed travel time, the average number of interchanges per journey, and the average buffer times of those interchanges.

Figure 8.10 shows the initial travel times. The best instance A_1_5 and the worst instance A_2_4 with an initial travel time of 19.57 minutes are 2.7 minutes apart. Here is a major gap of more than 10% in average travel time. Another noteworthy fact is that A_1_1, an algorithmically optimized version of P_1, shows a .33 minutes faster average travel time than the solution manually generated by planners. Comparing the average number of transfers (see Figure 8.11), the difference becomes more apparent. In instance A_1_3 (Figure 8.13) passengers have an average of .8 transfers per route while in instance A_1_5 only .22 transfers are necessary. Completely different line pools are the reason for this discrepancy. Instance A_1_3 (see Figure 8.13) connects stops in the same row with one line each. One additional vertical line in the central column then connects the five resulting lines. Because of this layout, many passengers have one or two interchanges. In most other instances, there exists a path for any two pairs of stations using only one interchange. Instance A_1_5 (Figure 8.14) has this property using only two additional lines while minimizing the number of interchanges. The layout, however, is asymmetric and would be hard to remember for passengers. While in instance A_1_3 the cental stop is only served by two lines, in instance A_1_5 all lines pass through this station.

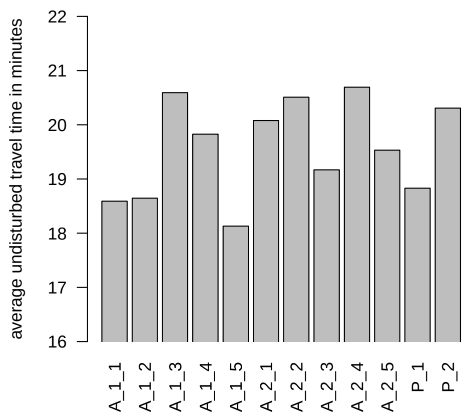


Fig. 8.10: Average undisturbed travel time [Fri+17b].

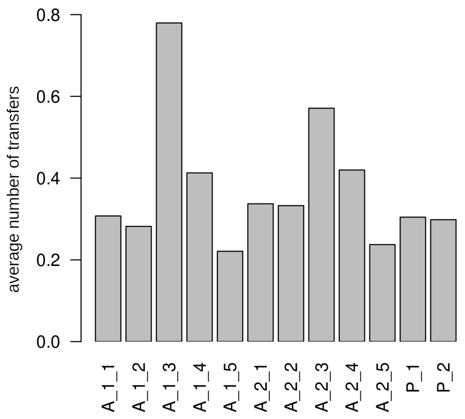


Fig. 8.11: Average number of transfers per route [Fri+17b].

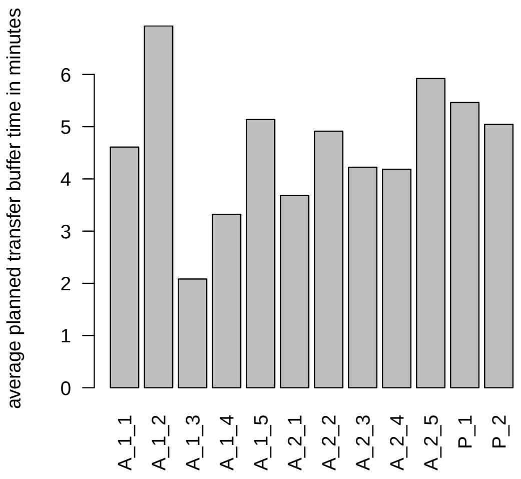


Fig. 8.12: Initial buffer times of routes used by passengers in different instances.

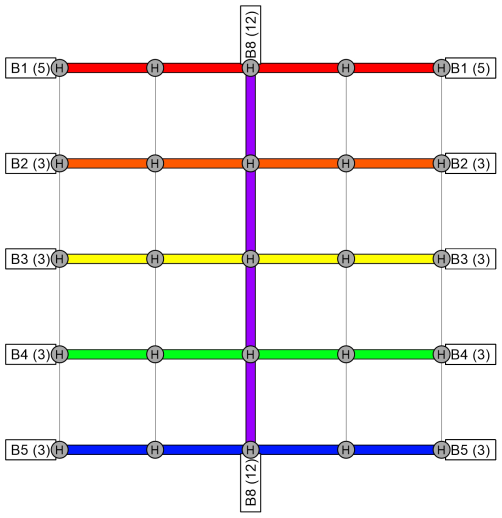


Fig. 8.13: Instance A_1_3 [Fri+17b].

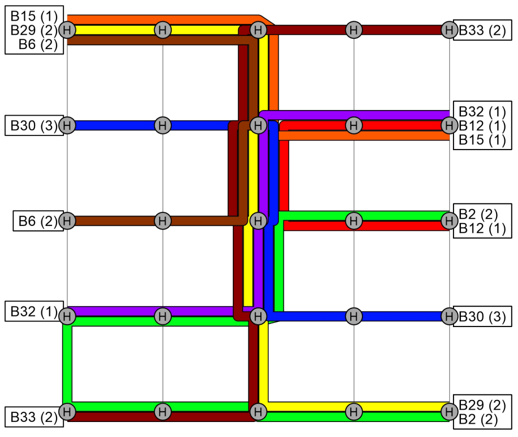


Fig. 8.14: Instance A_1_5 [Fri+17b].

Another feature of importance is the average slack on transfer activities of passengers. As mentioned previously, this is excess time available to perform an interchange. In networks with very high frequencies, this feature is of little importance because a missed connection has little influence on the total travel time. In a system with low frequencies and frequent delays, it is an indicator of potentially substantial delays for passengers using interchanges.

Figure 8.12 shows the initial buffer times of the 12 instances. Instance A_1_3 shown in Figure 8.13 has the lowest average buffer because of the high frequency. The central vertical line will depart every 5 minutes. Instance A_1_3 contains rare but expensive interchanges. The average waiting times are exceeding 6 minutes. There are also other features of the instance explaining later differences in performance during the robustness test. These features include the average length of lines and the number of lines serving one used edge of the infrastructure network. Using these 12 instances, robustness tests RT 1-3 were conducted using only one set of fixed parameters for each robustness test. In contrast to the other set of experiments, there are no circulation edges. One trip of a vehicle is not dependent on the previous delay of other trips. Neglecting circulation edges, however, may result in an underestimation of real delays. In Section 8.4.2, we investigate the importance of those circulation edges in detail.

Robustness test RT-1: simulation of initial vehicle delays

The first experimental results of the robustness test using this framework were conducted using the robustness test RT-1 with the fixed-parameter 10 minutes as an initial delay for each separate trip. The results are presented in Figures 8.15-8.18. On the x-axis, the instances are ordered by planned travel time, while there will be one performance indicator of the y-axis.

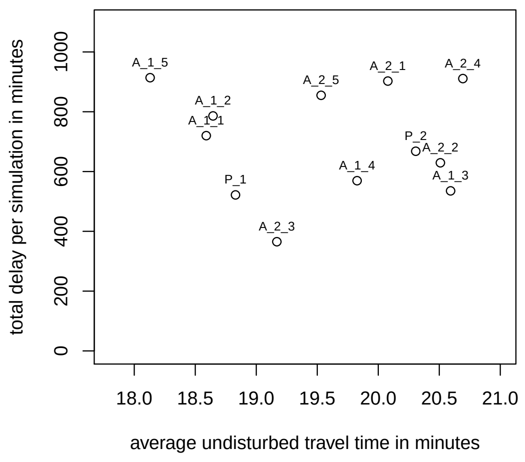


Fig. 8.15: RT-1: total delay per simulation in minutes [Fri+17b].

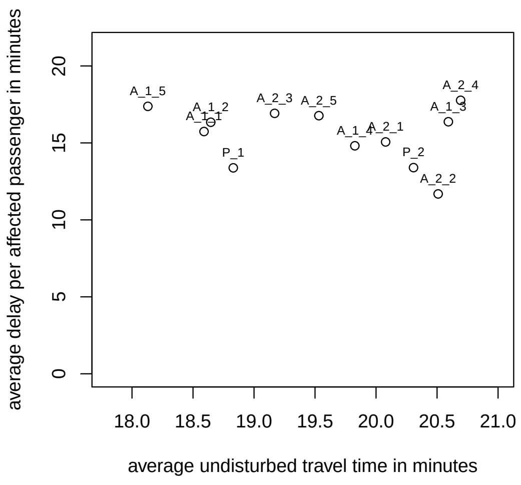


Fig. 8.16: RT-1: average delay of affected passengers in minutes [Fri+17b].

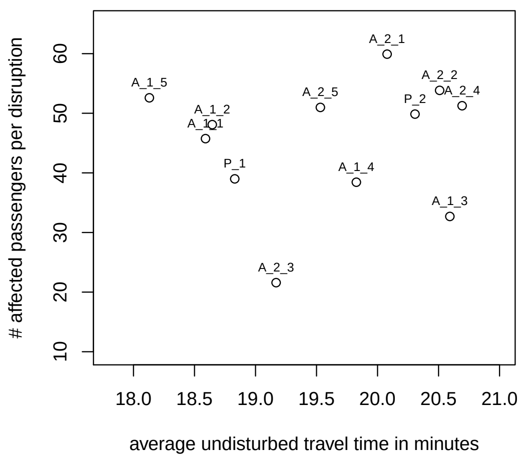


Fig. 8.17: RT-1: number of affected passengers per disruption [Fri+17b].

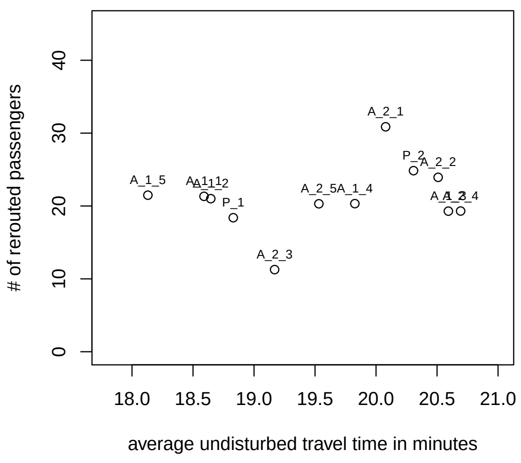


Fig. 8.18: RT-1: number of rerouted passengers [Fri+17b].

The total delay per simulation is shown in Figure 8.15, the top 5 instances concerning undisturbed travel time show a definite trade-off. The case with the best-undisturbed travel time A_1_5 is generating the most significant passenger delay. The instance using a grid of lines A_2_3 (shown in Figure 8.8) has the best robustness concerning 3 of 4 indicators. One explanation for this result is that two equally good routes are often available. Especially the number of affected passengers shown in Figure 8.17 is low. The schedules manually created by public transport planners generally show good robustness. Especially the amount of average delay per affected passenger is small.

Robustness test RT-2: simulation of a slow down of a network edge

The second experiment was conducted, introducing a 4-minute delay to every vehicle utilizing the selected network edge. During one simulation, we chose only one network edge at a time as being affected. The results of this robustness test are shown in Figures 8.19 - 8.22.
The first difference in total passenger delay per passenger is a correlation between average undisturbed travel time and the robustness measured in passenger delay. Instance A_1_5 now has one of the five best plans, while instance A_1_3 ranks among the five worst solutions. Especially Figure 8.20 suggest the longer a passenger initially spends on a journey, the higher the likelihood of an additionally longer delay. The number of rerouted passengers shown in Figure 8.22 is also positively correlated with average undisturbed travel time.

Robustness test RT-3: simulation of disruption at a stop

The third experiment was conducted using the parameter 15 minutes length of the disruption and two minutes headway. During one simulation, only one stop at a time is selected being affected. The results of this robustness test are presented analogously to the last robustness tests in Figures 8.23 - 8.26.

Evaluating the total delays per simulation RT-3 indicates that there are only two sets of instances sharing similar robustness. Instance A_1_2 has a small advantage in this experiment. On the other end of the spectrum, it is of interest to analyze why the instance A_1_3, A_1_4 and A_2_4 have significantly worse performance than all other instances. The key feature of all three instances is a very high frequency of vehicles traveling trough central stops of the network. A_1_3, for example, has a departure event every 3.5 minutes on average. Because of the headway-rule, a disruption in one of those stations will last much longer than disruptions at stations with lower frequencies.

Summary of the results

The three robustness tests conducted highlighted different strengths and weaknesses of the 12 instances given to us by public transport planners. Table 8.1 provides a comprehensive overview of the 12 plans, their features, and performance in the robustness tests measured in the combined rank of all three experiments. The example A_2_3 using a grid of lines has the best robustness. The surprising part is that the second and third most robust solutions all are using the line pool suggested by public transport planners. The line-plan A_2_4 created using sophisticated algorithms described in [GHS17] had the worst performance in terms of robustness. In contrast, the combination of their methods with a manual line pool resulting in A_2_4 came in as the 4th best solution.

8.4.2 Experiment Set 2 - Same Line Plan, Different Time Supplements

After studying different line networks, we ended up with a preferred line network. We use the line network used in P_1 and A_1_1 as a base for further research into robust timetables. The

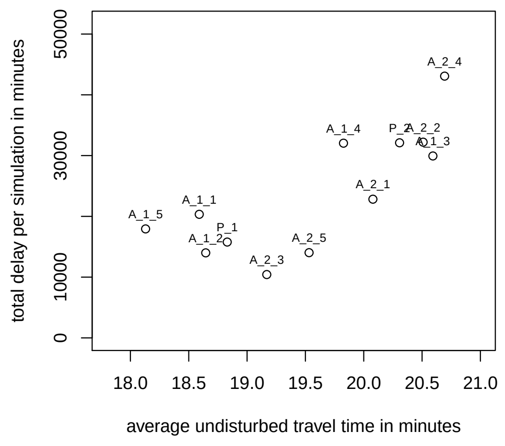


Fig. 8.19: RT-2: total delay per simulation in minutes [Fri+17b].

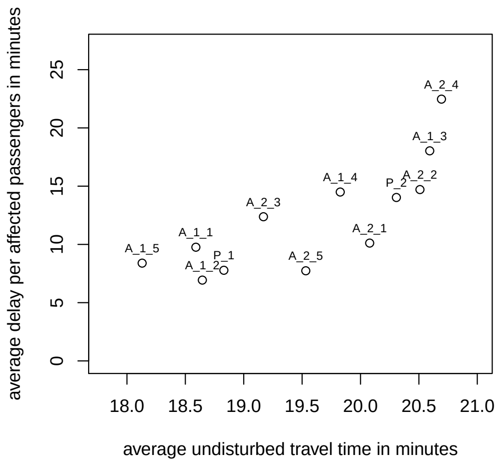


Fig. 8.20: RT-2: average delay of affected passengers in minutes [Fri+17b].

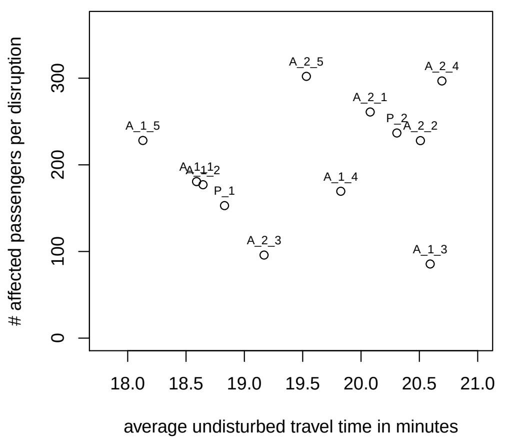


Fig. 8.21: RT-2: number of affected passengers per disruption [Fri+17b].

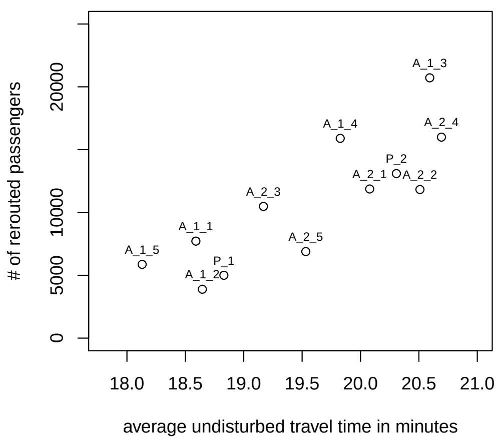


Fig. 8.22: RT-2: number of rerouted passengers [Fri+17b].

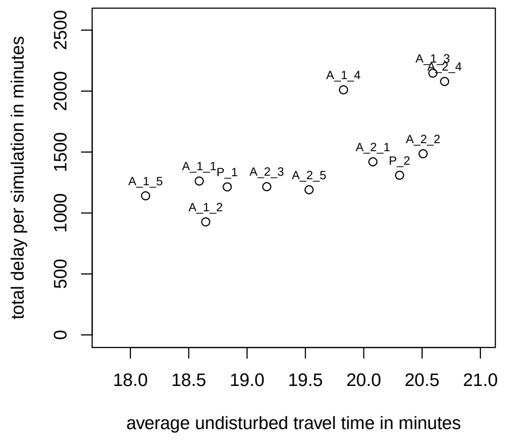


Fig. 8.23: RT-3: total delay per simulation in minutes [Fri+17b].

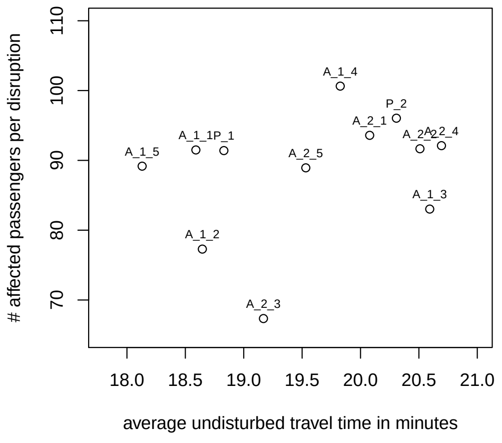


Fig. 8.24: RT-3: average delay of affected passengers in minutes [Fri+17b].

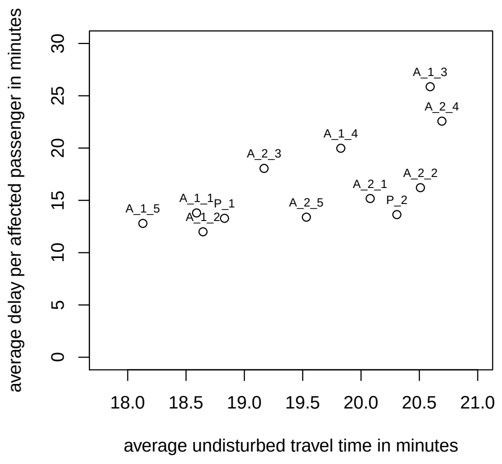


Fig. 8.25: RT-3: number of affected passengers per disruption [Fri+17b].

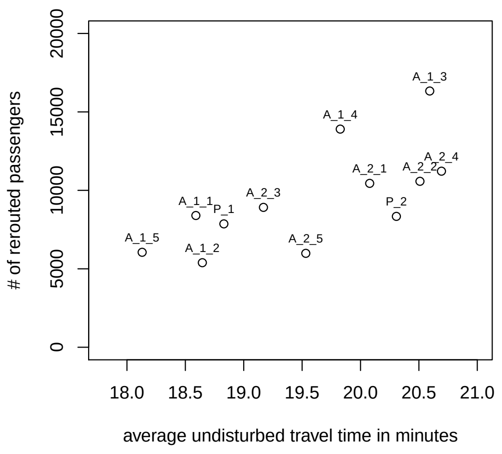


Fig. 8.26: RT-3: number of rerouted passengers [Fri+17b].

Tab. 8.1: Main characteristics of examined line networks influencing the robustness.

line network	lines per stop	lines per edge	line length	characteristics increasing vulnerability	characteristics increasing redundancy	rank test 1-3
P1, A_1_1	1.7	1.5	12.0	one network section serving as backbone, low edge coverage	high frequency lines and overlapping lines compensate delays of other lines	2 5
A_1_2	1.7	1.5	12.0		low service frequency reduces probability of being affected by short disturbances , high transfer buffers	3
A_1_3	1.2	1.0	8.0	one backbone line, only one transfer point per feeder line	short line length reduces impacts of service run delays	8
A_1_4	2.2	1.8	9.8	one network section serving as backbone, low edge coverage, short transfer buffer time		9
A_1_5	2.5	2.2	13.9	one network section serving as backbone		6
P2	1,6	1.4	14.4	few OD-pairs with alternative routes, low edge coverage, long line length	high buffer time for transfers	11
A_2_1	1.6	1.4	14.4	few OD-pairs with alternative routes, low edge coverage, long line length		7
A_2_2	1.6	1.4	14.4	few OD-pairs with alternative routes, low edge coverage, long line length	low service frequency reduces probability of being affected by short disturbances	10
A_2_3	2.0	1.0	8.0		no critical network elements, all edges are served by a line, short line length	1
A_2_4	3.4	2.4	11.1	multiple central stations		12
A_2_5	2.7	1.9	15.0	long line length increases impacts of service run delays	high frequency lines and overlapping lines compensate delays , high transfer buffer times	4

focus of continued research on this topic will be on the introduction of time supplements. In addition to this change in focus we are now able to cover all realistic values for the parameters of the experiments.

For the following robustness test, a new set of instances was created using the LinTim Software [Lin]. These instances now feature circulation edges for the realistic turn around times during delay scenarios. Ten classes of instances were created using different strategies for the application of time supplements (see Table 8.2). Each class of instances has six members with

Tab. 8.2: Classes of instances (timetables) with different time supplements

A	supplements at 5% of all stops
B	no supplements
C	1 minute supplement for few driving sections
D	2 minutes supplement for few driving sections
E	3 minutes supplement for few driving sections
F	supplements at 10% most frequently used stops
G	supplements at 15% most frequently used stops
H	supplements at the 5% most frequently used driving sections
I	supplements at the 10% most frequently used driving sections
J	supplements at the 15% most frequently used driving sections

using at least a defined number of minutes as time supplement on the circulation edge. The defined number of minutes for these instances is $0, 1, 3, 5, 7, 9$ minutes [Fri+18]. All instances have lines operating 2 or 3 times every hour.

In the last section, there were significant differences between all instances. Because of the fixed line-plan, the new instances are very similar in contrast. This high similarity should result in very similar outcomes of the results. Before going into details about the experiments conducted Table 8.3 contains a summary of the differences between both sets of experiments.

We use the same methods as in our fist set of experiments but a large number of parameters for a more specific comparison between classes and instances. The circulation edges and the capacity limitation add detail to our model while the set of parameters adds another dimension for the evaluation. We are now able to investigate new questions using these new instances:

- What are the best strategies for time supplement insertions?
- Is the performance of supplement strategies sensitive to the different robustness tests?
- What is the impact of circulation edges on the results?
- Does the application of vehicle capacities influences the results?

In the first part, we want to focus on the robustness test RT-1,2,3. In Section 8.4.1 we had to use fixed parameters for each robustness test. In this section, we can test all realistic values for each parameter. Our software can perform each simulation in less than 30 minutes per instance

Attribute	**Experiment Set 1**	**Experiment Set 2**
number of instances	10	60
instances differ in	line pool, scheduling method	used time supplement
number of tested parameters (RT-1,RT-2,RT-3)	1,1,1	18,10,10
circulation edges	no	yes
limited capacities	no	yes

Tab. 8.3: Summary of differences between experiments conducted in [Fri+17b] and [Fri+18]

and parameter. First, we will give the set of chosen parameters and explain the motivation behind the limits; then, the experimental results are presented.

Setup of the robustness tests

- RT-1 $x = 1..18$ min initial vehicle delay
- RT-2 $x = 1..10$ min delay for crossing disturbed edge
- RT-3 $x_1 = 10..20$ min blocking time for disturbed station. $x_2 = 1$ minute headway

For RT-1 the maximum number of minutes is set to 18. The reasoning behind this choice is the typical minimum frequency of 20 minutes the vehicle of the next trip is departing from most stops. Choosing a more substantial starting delay than 18 minutes would introduce methodical side-effects. The selected vehicle would now depart just one minute before or even after the next trip. This causes massive problems in theory as well as in practice. In practice, passengers may enter the wrong vehicle according to their original plan generating new problems with a higher rate of occupation for this vehicle. The resulting situation is very different from the intention of RT-1 and its evaluation. Therefore, if such a large delay should be introduced it deserves its own robustness test or at least a separate evaluation. For RT-2, the maximum amount of additional delay when using the disturbed infrastructure edge is 10 minutes. That is quite a significant delay considering the minimal travel times of 6 minutes on any edge. In practice dispatchers would immediately choose another route for vehicles scheduled on a journey with such substantial delays. If the parameter would be 12 minutes of additional delay, the bus could take a detour in the same amount of time, covering two other stops.

For RT-3, the minimal duration of disruption is 10 minutes. Smaller disturbances may not include any vehicles. A disruption more massive than 20 minutes may lead to a situation where the effects will continue until the end of the day. Here the first vehicles leaving the disrupted station could re-enter the disruption on their next trip.

Evaluating the second set of robustness tests is more difficult due to the added number of parameters and instances. In Figure 8.27 and the following we only show either the instances with zero or 9 minutes of circulation buffer. Figures 8.27 and 8.28 show a direct comparison of both subsets of instances. Both figures look identical until they reach an initial delay of 8 minutes. When the initial delay exceeds 9 minutes, a difference of class A-G becomes visible. To make an evaluation easier, the performance of all instances is measured in a total delay of all passengers at their destination only.

A final summarizing figure of all three experiments and parameters is Figure 8.29. The figure is showing the relative ranking of ten instances in relation to all 60 instances. Instance $A - 9$ (blue dotted line) for example is ranked 60th in experiment RT-1 with parameter $x = 4$ minutes and ranked 40th with $x = 12$ minutes. Instances A-G follow one specific trend. The instances with nine minutes of circulation buffer seen in the left plot improve when the severity of the disruption is increasing. The cases with zero minutes of circulation buffer seen in the right figure worsen when we increase the severity. This behavior is characteristic of instances A-G. The

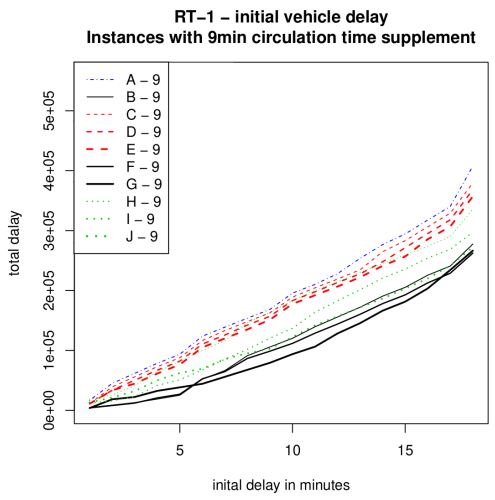


Fig. 8.27: Results of RT-1 instances with 9 minutes of time supplements on circulation edges [Fri+18]

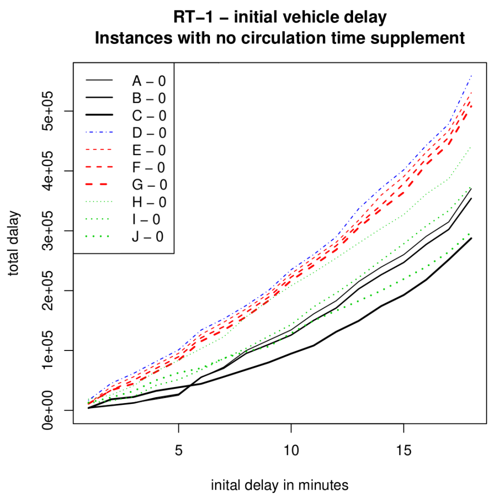


Fig. 8.28: Results of RT-1 instances without time supplements on circulation edges [Fri+18]

cases with zero minutes transfer buffer mainly outperform instances within their class and in classes with a similar amount of short buffer times. When we increase the introduced delays, almost identical instances with circulation buffer will absorb the disruption at the end of their trips, finishing delay propagation. The instances missing circulation buffers will continue their delayed trips much further into the day. Comparing the rightmost points of each line of those classes reveal that the instances with 9 minutes of circulation buffer often outperform classes having a higher degree of time supplements on dwelling arcs. This fact suggests a superiority of circulation buffers over other time supplements on dwelling arc for this network.

Classes I, J, K does not appear to follow the same pattern. A detailed analysis shows that those instances have sufficient time supplements on driving edges to absorb most of the introduced delays rendering their circulation times pointless.

8.4.3 Combining all Three Key Performance Indicators

We understand robustness to be a third key performance indicator for planning in public transport, the two others being operational costs and total travel time. For the task of comparing objects having multiple numeric attributes, we are using radar (or spider) plots. Figure 8.30 is an example showing 3 instances. The center point of each triangle corresponds to the worst solution, while the corners represent the best values of one instance in the respective attribute. Instances having a large blue triangle area can be interpreted as good solutions.

Figure 8.31 shows a comparison of all instances with a time supplement of 9 minutes on circulation edges. Instances (D), (E), (F) have the largest triangle area and can be seen as cases with a good trade-off. Instance (I) is the worst instance shown here, having long initial travel times and high operational cost.

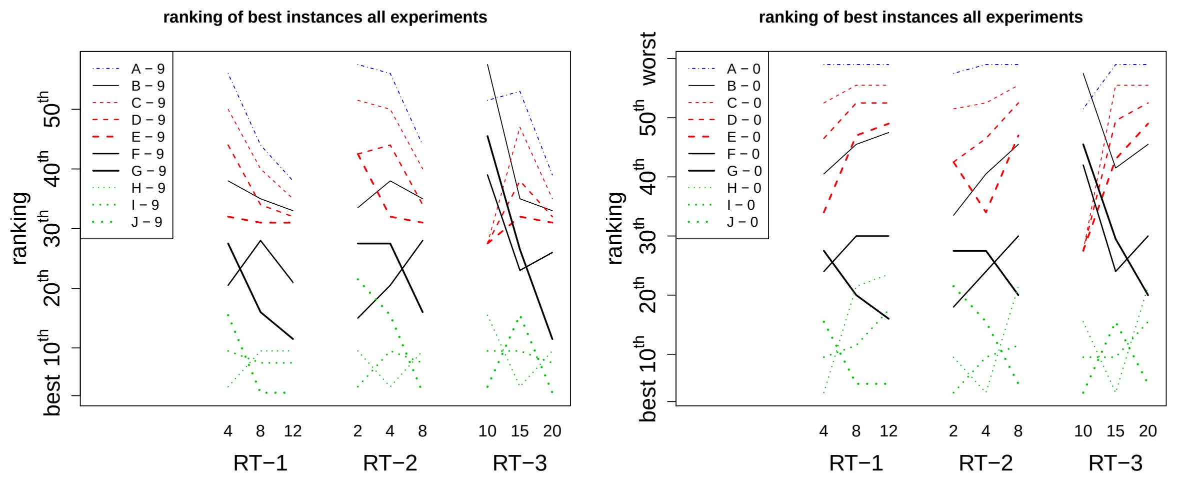


Fig. 8.29: Ranking of all experiments and parameters. Plotted are changes in rankings of instances with 9 minutes circulation buffer(right) an 0 minutes circulation buffer(left) [Fri+18]

This comparison concludes the evaluation of the second set of instances. Before looking at larger networks, we analyze additional questions about the model. First, we want to look at the impact of vehicle capacities on traveling times.

8.4.4 Impact of Realistic Vehicle Capacities

Vehicle capacities are an essential aspect of the planning process. Planners receive a specific passenger demand model and design a schedule that has to satisfies certain conditions. Capacity limitations can divided into hard capacities and nominal capacities (see Section 8.3.4). Both have a significant influence on the validity and level of comfort of a given route.

Planners use nominal capacities for the design of services. In contrast to daily operation, planned vehicles need to have some room to compensate for passenger fluctuation. When research of the FOR2083 group generated plans for the grid network, they had several restrictions. These restrictions included frequencies and hard capacities. The limit for every vehicle was 65 passengers. To see the impact on using hard capacities on an otherwise undisturbed network, we conducted a simulation experiment. In multiple runs of the same day capacities are limited to $cap = 55, 60, 65, 70, 75, 80$ passengers. The fraction of passengers in need of re-routing is then measured. Figure 8.32 shows the results of this experiment. Postulating hard capacities of 80 passengers cause no re-routing passengers. Limiting the capacities further reveals cases where some passengers need re-routing. When we set the limit to 65 (the amount used to create the instances), a fraction close to 0.5% of all passengers is re-routed. This fraction may look insignificant at first sight. This perception changes when looking at the impact of passenger distribution over one day. Figure 2.7 shows that rush-hour traffic is prominent in our data. Between 5:00 and 7:00, the fraction of overcrowded busses exceeds the planner's expectations by far. Limiting the vehicle capacity further shows a substantial increase in re-routed passengers.

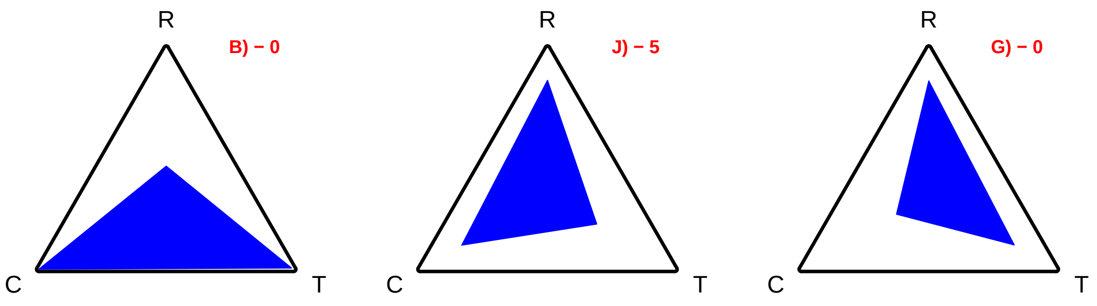


Fig. 8.30: Example of three instances (from the track delay experiment): instance none_0.05_0 with the best cost and best travel-time but poor robustness (left), instance links_0.15_5 with best robustness but medium costs and bad travel-time (center), and the instance stops_0.15_0 where robustness and travel-time are well balanced (right) [Fri+18].

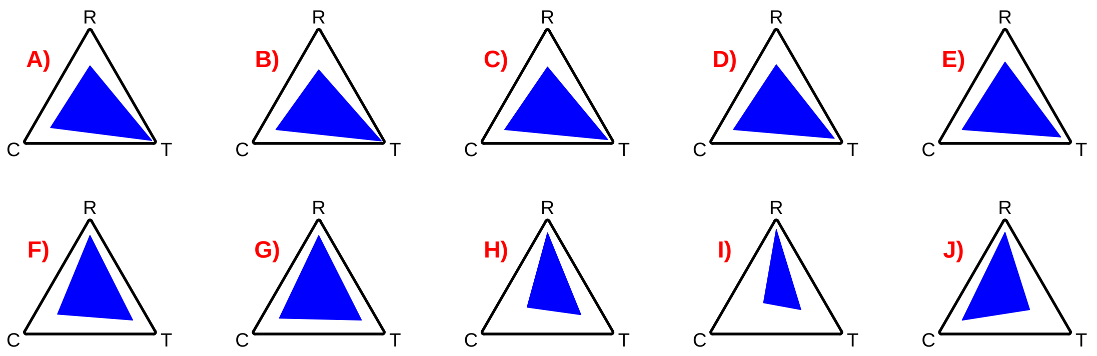


Fig. 8.31: Comparison of operational cost (C), average travel-time (T), and combined robustness concerning robustness tests 1-3 (R) for instances A-J [Fri+18].

LinTim optimized the schedules for costs while satisfying the capacity constraints. This method leads to a schedule with a very high number of average utilization. There is no incentive for LinTim to create a good distribution of passengers among vehicles. In practice, planners for public transport have a large margin between their planned capacities and the hard capacities of a vehicle. Planners for bus schedules typically plan not to exceed the nominal (seating) capacities during non rush hour traffic. In quality control, the number of times passengers claim that there were no free seats are counted. If those claims exceed a certain threshold, planners shift to larger vehicles or higher frequencies.

Using a capacity limitation in the other experiments would have a significant influence on the results. The decision was made not to use capacities for the rest of the experiments to assure that effects are limited to the delayed schedule and not due to the capacities of the whole system.

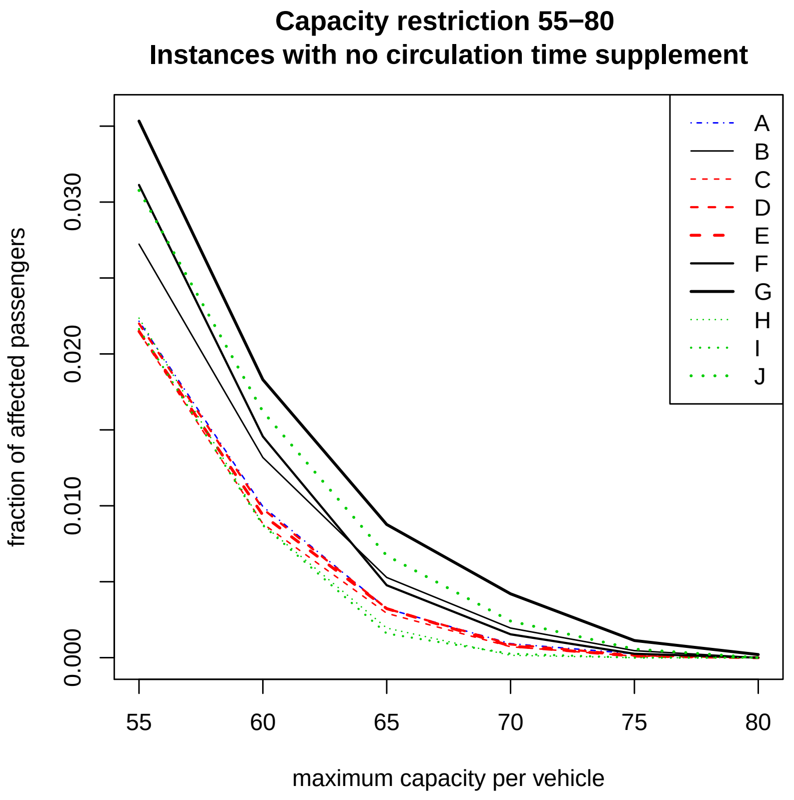


Fig. 8.32: Simulation of a regular day without disruptions limiting the capacity of vehicles.

The next section will cover another detail, influencing the outcome of the simulation: The usage of circulation edges.

8.4.5 Impact of Removing Circulation Edges

In public transport, creating a vehicle and crew schedule is an important process resulting in operational costs for a given timetable. The vehicle schedule defines dependencies between trips. Vehicles arriving with a delay at the end of their trip can cause a delay in the second trip. The same may also be true for necessary crew members scheduled for different trips. Those may have replacements while vehicles typically are not replaced with others due to delays.

While having access to timetables is relatively easy, it is uncommon to export vehicle- and crew schedules for open data purposes. The GTFS format, for example, does not contain any information about the physical vehicles or circulation edges. For studying the impact of neglecting these edges, we conducted an experiment. The set of instances used in this section contains circulation edges with minimum execution times. We conducted the robustness test RT-1 two times. During the first run, we used circulation edges while during the second run, we neglected them. The result is shown in Figure 8.33. There are two distinct differences in those two runs. The first one shows a significantly larger total delay and a steeper incline in

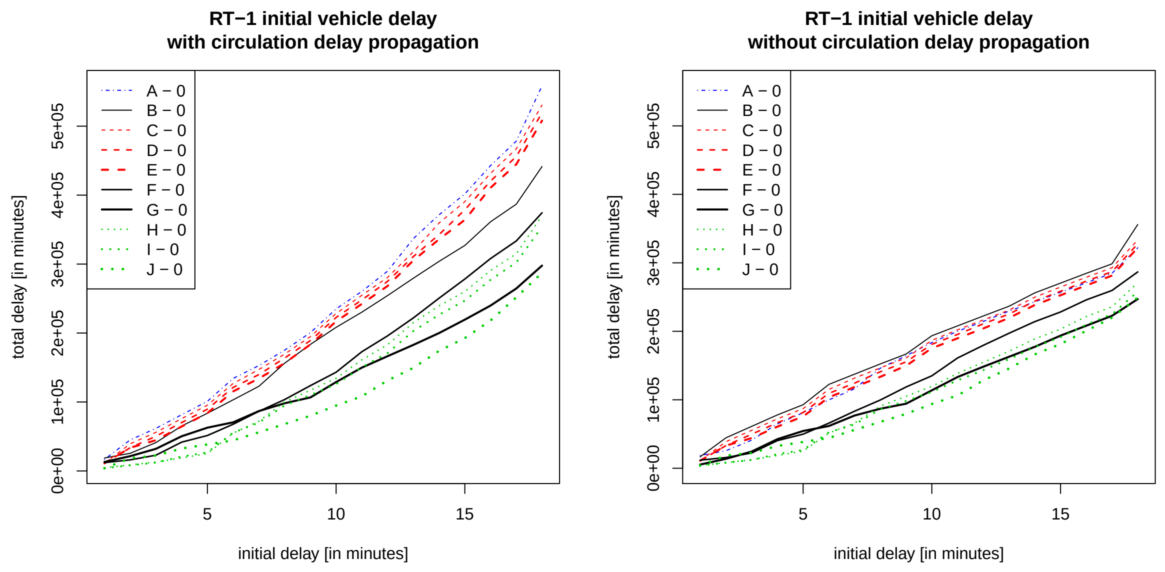


Fig. 8.33: The importance of circulation edges. Plot with circulation edges(left) and without circulation edges(right) [Fri+18].

delay when the initial delay exceeds 7 minutes. While the relative difference of instance C, for example, is marginal with 10 minutes of initial delay we see a 70% increase using the parameter 18 minutes compared to the other run.

The second noticeable difference between the two experiments is a reversal in ranks of instances A and B. Instance A has only a small amount of buffer, while instance B has none. Instance B has a better performance using the less realistic run with the circulation edges disabled. In the more realistic run, it performs worst among all instances.

This experiment demonstrated the importance of circulation edges, especially in cases with substantial delays.

8.4.6 Experiment Set 3 - Different Schedules on the German Long Distance Network

The final set of instances are simplified versions of the German long-distance train network. The authors of [Fri+18] provided us with those instances and passenger flow from Göttingen Univerity using the LinTim software. The schedules include 250 stops, 22 of which are in neighboring countries (see Figure 8.34). The set contains an artificial passenger flow of 380k passengers. This number represents the typical average day of passengers traveling with long-distance trains in Germany.

For this set of robustness tests, four instances have different strategies for the insertion of time supplements. Table 8.4 shows these classes.

We conducted the Robustness test RT-1/2/3 in the same fashion as in the last section. There were some small modifications:

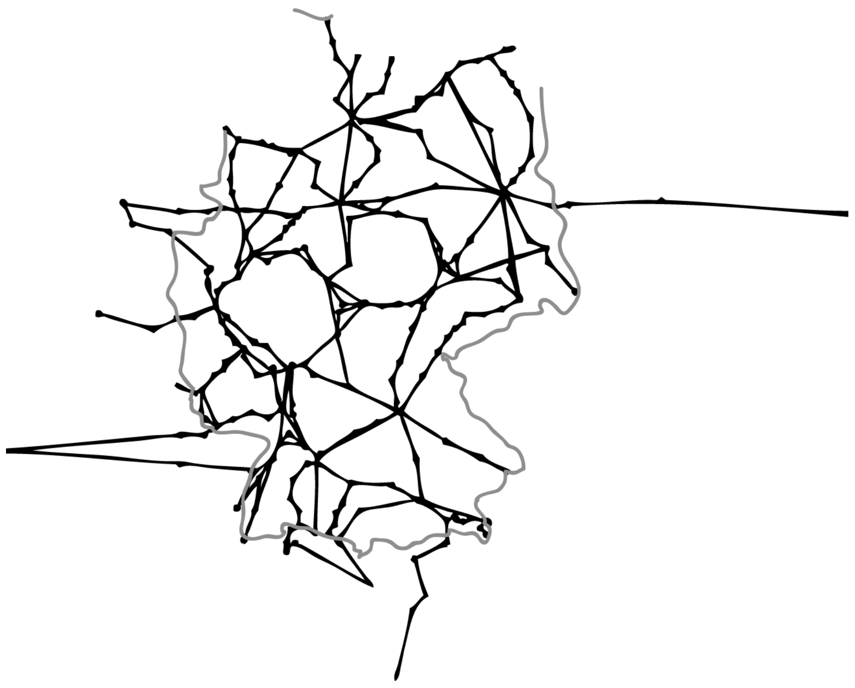


Fig. 8.34: Map of stations and edges used in the DB-instance [Fri+18].

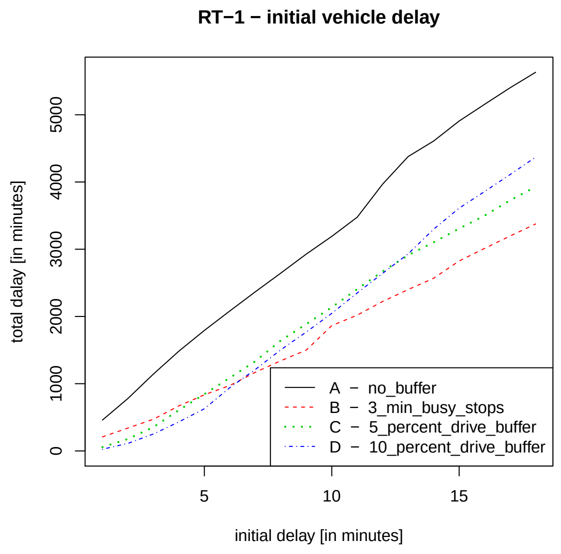
Tab. 8.4: Classes of Instances (timetables) with different time supplements

A	no supplements (“no_buffer”)
B	supplement time is at least 3 minutes at busy stops (“3_min_busy_stops”)
C	supplement time is at least 5% of driving time (“5_percent_drive_buffer”)
D	supplement time is at least 10% of driving time (“10_percent_drive_buffer”)

- the representation in LinTim of the infrastructure is a simplified version of the real infrastructure,
- there are no circulation edges for this experiment,
- the price of an additional transfer (in the routing choice model) is now equivalent to 10 minutes of extra travel time.

The results of the first experiment are shown in Figures 8.35 and 8.36. The total delay of all passengers is rather small for these experiments. Five thousand delay minutes are rather insignificant given a flow of 380k passengers. There is a small gap in performance between instance A and the other instances who behave similarly in this robustness test.

The results of experiment RT-2 is very different in contrast to RT-1 (see Figure 8.37). The disruption is generating a large amount of delay, the difference between instances A and the other instances is massive. The instances maintain their strict ordering of ranks among all tested parameters. The ranking corresponds with the amount of time supplement used in the instances ordered from least to most. The results of RT-3 shown in Figure 8.38 are similar.



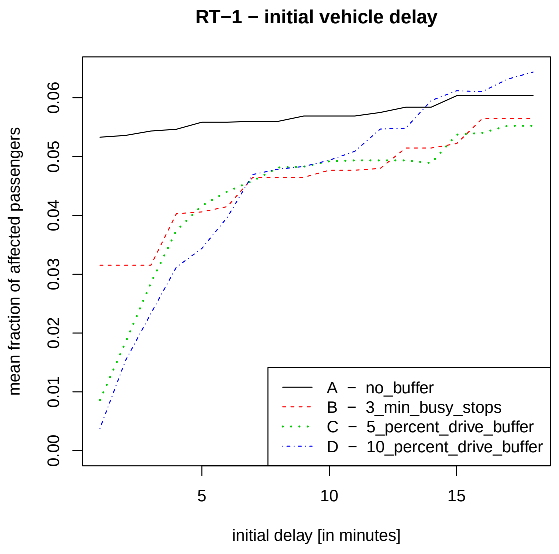


Fig. 8.35: RT-1 on the German long distance instances. Evaluation of total dalay [Fri+18].

Fig. 8.36: RT-1 on the German long distance instances. Evaluation of total mean number of affected passengers [Fri+18].

8.4.7 Random Delay Model

For the long-distance German train network, delay data was provided to us by the Deutsche Bahn. This data was used to create a random delay model for robustness test RT-4 presented in Section 8.2. The data consists of one year of delay data from 2016 to 2017 containing over 28 million events for high-speed trains.

Before introducing the model we use, it is important to analyze the difference between the real network and the generated schedules. Our generated instances include all minimal travel times and minimal dwelling times. For the real network, minimal dwelling times are partially available to us, while minimal travel times are not. The general infrastructure of the real network is not available to us, while the artificial network has only a simplified infrastructure.

Another important aspect of the creation of the model is the way delays are created and counted in the real network. The basic definition of a delay is a positive deviation from the scheduled execution of an event. There is a fundamental difference between delays a model generates and delays counted during operation. The delays generated are primary delays that should be representative of random primary delays occurring in the real network. Delays that are not primary delays should not be introduced during the model creation.

Naturally, primary delays can occur at arrival and departure events. Unfortunately, there are two major problems with counting delays for departure events. The first problem is that a significant number of dispatching decisions (especially small ones) are not well documented. This lack of documentation is a problem, when deciding if a delay is a random occurrence or not.

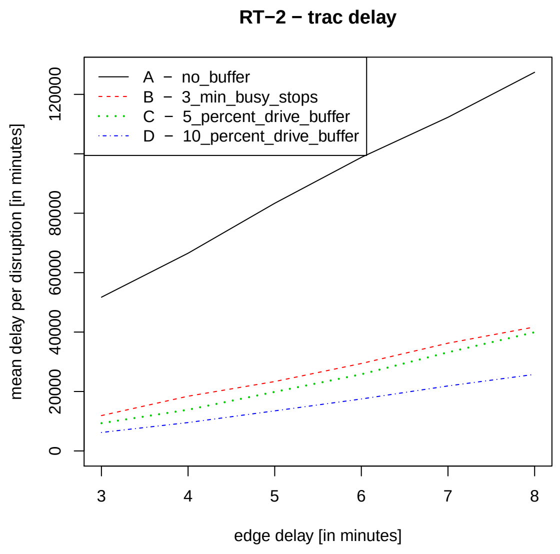


Fig. 8.37: Robustness Test 2 on German long distance instances [Fri+18].

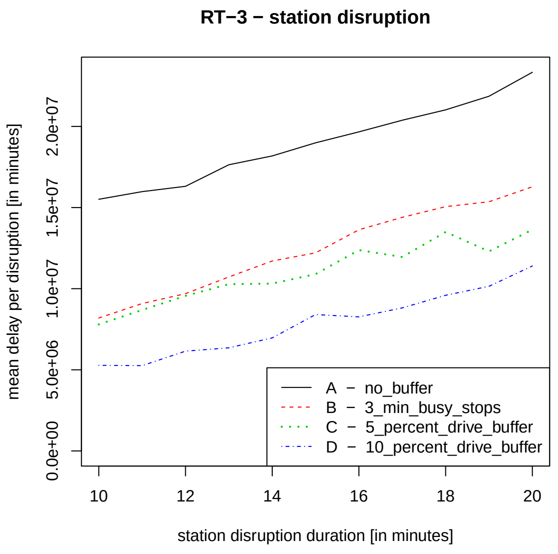


Fig. 8.38: Robustness Test 3 on German long distance instances [Fri+18].

When a disruption generates a primary delay on an edge that contains a buffer, the measured delay is lower by the amount of buffer.

For these two reasons, we decided to create two separate models for analyzing and generating primary delays. Departure edges at the beginning of a new trip have their own probability distributions. Dispatchers rarely included them in waiting decisions. The other distribution is for primary arrival delays accumulated on driving edges. This is because those delays are likely to be the result of a problem rather than a decision. Those distributions were generated using data from the real world data resulting in those shown in Figures 8.39 and 8.40.

Using these distributions, experiment RT-4 was conducted simulating the same day of a schedule in multiple runs. Because of the stochastic nature of this experiment, every trial will result in different delays for the passengers. From several pretests, we learned that 50 runs are sufficient to yield stable means of our test indicators. Our experiments resulted in an average delay of between five and twelve minutes for the average passenger. This gap between the best and the worst solution is relatively large. We now look at this gap seen in the context of inserted times supplements. Figure 8.41 shows a clear trade-off. The dotted diagonal line with slope -1 indicates that instances B and C have a better tradeoff than instances A and D. While instance A is slightly superior in the expected arrival times of a given passenger, public transport planners and passengers would probably prefer schedule D. Here there is a much smaller chance of long delays and re-routings for single passengers.

The expected arrival time for the average passenger is of particular importance to this robustness test. An improvement of the model and input data can lead to a situation where we could better approximate the real world expected delay.

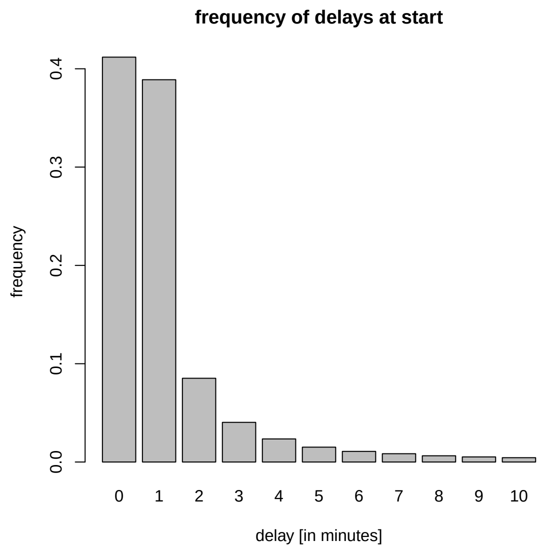


Fig. 8.39: empirical delay distributions of initial departure events of one trip [Fri+18]

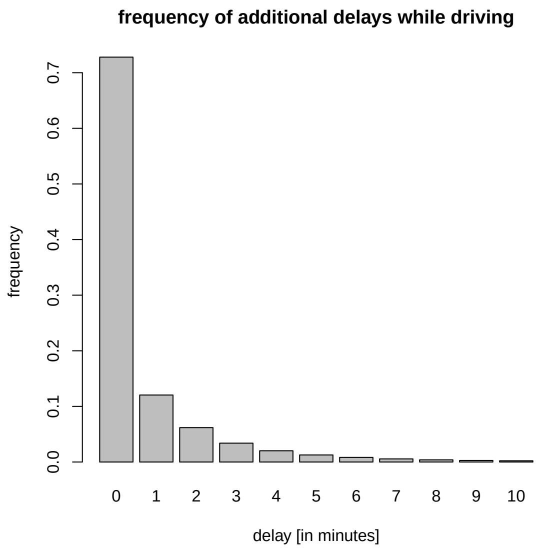


Fig. 8.40: empirical delay distributions of primary delays on driving edges [Fri+18]

8.4.8 Testing Waiting Strategies

In the last section, the usefulness of RT-4 for getting more realistic travel times rather than undisturbed travel times was motivated. A given instance undergoing RT-4 can produce different results when the simulation model is reacting in different ways to delays. A common tool for dispatchers is the usage of *waiting times rules*. These rules should, in theory, give a guideline when operators should keep a connection (assuming there are at least some passengers to be expected on a particular interchange). These rules can be specified for types of trains or even specially defined for certain connections. The next experiment will show if RT-4 can be used to derive a set of simple waiting time rule for trains in instances of the German long distance network.

The experiments consist of several runs of RT-4 using an additional parameter defining the maximum number of minutes trains wait for changing passengers. We introduce the random delays at the scheduled time of departure. This experiment needs the same 50 executions per parameter for a reliable mean delay. The results of this experiments are shown in Figures 8.42 and 8.43. The left figure shows the mean delay in each instance. As the change in the mean delay is relatively small Figure 8.43 is showing the differences in mean compared to the no-waiting simulation. One fundamental feature of the resulting mean delays for each set of waiting time rules is that they have one global minimum. This global minimum marks the best strategy for each instance. If this strategy is applied, the simulated mean delay is decreased to a certain extent serving a lower bound. Using more sophisticated waiting decisions such as the use of the PANDA software can further improve the simulated mean delay.

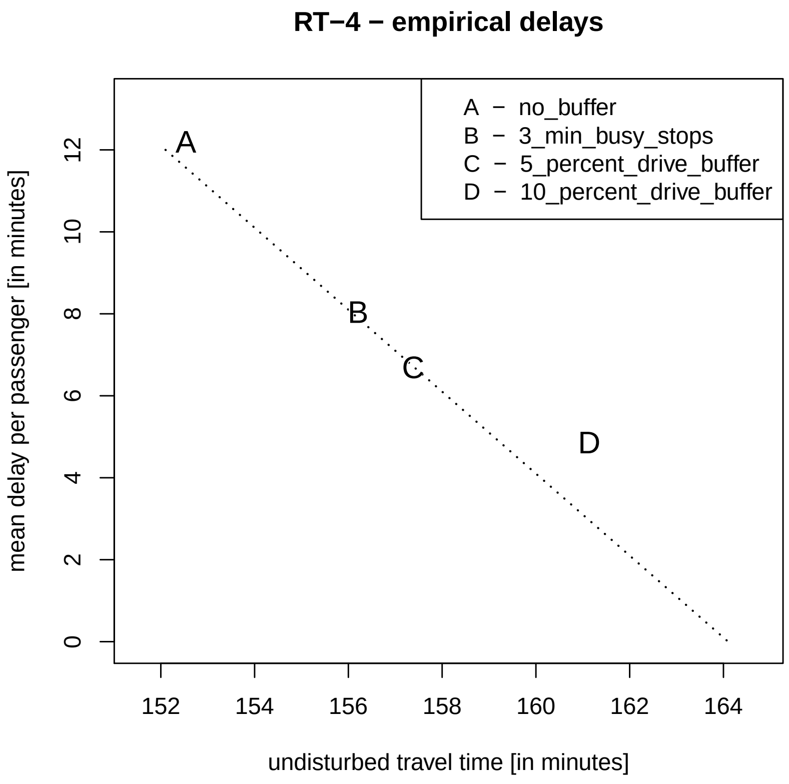


Fig. 8.41: The tradeoff between average travel time in the disturbed and the undisturbed case with inserted time supplements [Fri+18].

Comparing the best waiting time strategies reveals a correlation between used time supplements and the amount of waiting minutes. The no-wait strategy work best concerning instance A, having no time supplements, while the other instance can further improve their simulated mean delay by a significant margin. Instance C can improve the average mean delay by one minute, which would close the gap in trade-off compared to instance A.

8.5 Conclusion and Impact

Our robustness tests offer a new way to evaluate and compare plans and strategies in public transport planning. In this section, we discussed the real-world benefits of the presented tool. The existing software for planning and simulating public transport is primarily focused on two things. The first thing is minimizing cost for public transport and evaluating strategies for dealing with online delay scenarios.

While both aspects are precious themselves, classical optimization in planning lacks in the ability to evaluate robustness metrics for the presented solution. Delay management, on the other side, may never use its full potential of methods discovered if the corresponding plan is

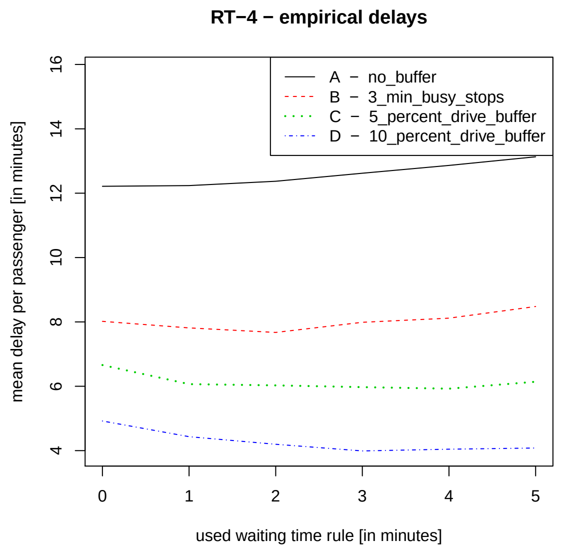


Fig. 8.42: Mean delay using different waiting time rules on the German long distance instances [Fri+18].

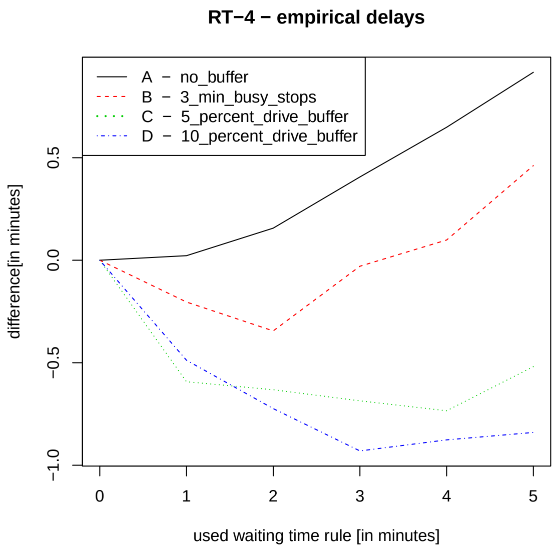


Fig. 8.43: Change in mean delay compared to using 0 minutes as waiting time rule on the German long distance instances [Fri+18].

not flexible enough. These simulation techniques can very easily be adapted to answer multiple sets of what-if questions about instances. Examples:

- What happens to the average passenger satisfaction if the demand increases?
- What paths do passengers use in the network if we remove a specific line/trip?
- Supposing one vehicle breaks down overnight. What cancellation for today will have minimal dissatisfied passengers?

Quickly simulating questions like these add a useful tool to planners and delay managers. As long as planners can not include intricate eventualities into the optimization or uncertainties in daily execution, the presented methods can add value to public transport.

On a final note, we want to point out that RT-4 could be used to predict the expected delay in a network there are two essential features present. The delay distribution should also represent correlations between delays often present in real-life delay scenarios. The second essential improvement would be an even more realistic model for heterogeneous passenger behavior.

Improving the Robustness of Schedules Using Machine Learning | 9

Real stupidity beats artificial intelligence every time.

— **Terry Pratchett, Hogfather [Pra96]**

In the last chapter, we introduced robustness tests and showed their usefulness. They help to find more robust instances among a set created for one of two networks. A natural extension of this work is to use our knowledge and methods to improve the creation of new instances that make them more robust. We first presented the foundation of this work in [Mül+21b], while the more detailed approach is found in [MH+22]. We know that creating plans for public transport that have three objectives (speed, cost, and robustness) naturally has many Pareto-optimal solutions. During one step in optimization, we could only improve speed or cost while the evaluation of robustness is computationally expensive. The addition of any buffer during optimization may yield additional robustness, but we do not know the result until another relatively expensive test is conducted.

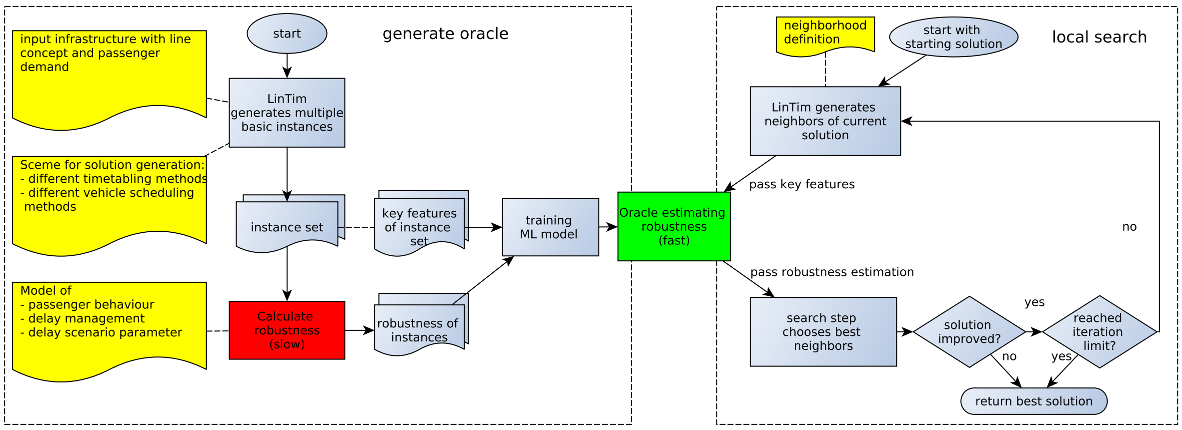
9.1 Goals and Contribution

An integration of robustness during the optimization only has practical relevance once the evaluation of robustness associated with a change in an instance is adequately fast. In many fields of computer science, it is common practice to sacrifice accuracy for running time. We are willing to use a very good approximation for our robustness test for dramatically reduced running time. Here we decided to use machine learning, to learn the robustness from already computed instances and then use the model as an oracle during optimization.

This gives us a clear goal for the continuation of our work improving the robustness of public transportation plans:

1. creating a machine learning based oracle that can predict robustness
2. validation of this oracle
3. including the oracle into the optimization framework of LinTim
4. evaluating if more robust solutions can be found using this technique

For a better graphical understanding of how these steps work into making a better public transport plan see Figure 9.1. The next section will introduce further related work that was not mentioned in Section 8.1.



generate oracle
start
input infrastructure with line concept and passenger demand
LinTim generates multiple basic instances
Sceme for solution generation:
- different timetabling methods
- different vehicle scheduling methods
instance set
key features of instance set
training ML model
Oracle estimating robustness (fast)
Model of
- passenger behaviour
- delay management
- delay scenario parameter
Calculate robustness (slow)
robustness of instances
local search
neighborhood definition
start with starting solution
LinTim generates neighbors of current solution
pass key features
no
pass robustness estimation
search step chooses best neighbors
solution improved?
yes
reached iteration limit?
no
yes
return best solution

Fig. 9.1: Workflow for creating the oracle and using the oracle within local search. Yellow fields denote input or choices of models and methods which are specific for each application. In detail: Left Box: Creating the oracle for estimating robustness of a public transport schedule by training a machine learning model. This step is only done once for each dataset: For a given infrastructure, a line concept and a passenger demand, we generate many different timetables and for each of them many vehicle schedules and corresponding passenger routes for training, validation, and testing. Afterwards the oracle can be used for black-box optimization of public transport schedules. Right Box: Local search as an exemplary application of the oracle. With each iteration through the optimization loop, we try to increase the robustness of the instance. Once the local neighborhood does not contain a more robust solution, the optimization terminates. [MH+22]

9.2 Related Work

Most related publications concerning timetables are already covered in Section 8.1. In the field of machine and deep learning, there is much activity. Methods from machine learning, however, are not currently used in the optimization of public transport schedules. The main uses of machine learning reported in domain review articles include traffic flow forecasting, traffic signal control, automatic vehicle detection, traffic incident processing, travel demand prediction, autonomous driving, and driver behaviors, see Nguyen et al. [Ngu+18], Wang et al. [Wan+19], and Vargehese et al. [VCU20]. Algorithmic approaches like Matos et al. [Mat+21; Mat+18] use reinforcement learning in periodic timetable optimization without robustness. Bauer and Schöbel [BS14a] used machine learning in delay management, predicting up to which delay maintaining a connection is beneficial.

A more common field where machine learning is used is delay prediction. Oneto et al. [One+18] build a data-driven train delay prediction system in an online setting based on learning algorithms for extreme learning machines. Yap and Cats [YC20] consider the problem of predicting disruptions and their impact at specific stations, while Cats and Jenelius [CJ18] predict the impact of a delay after its occurrence. Predicting where in the network delays will occur is considered in Cats et al. [CYv16] or Yap et al. [Yap+18]. In contrast to these papers we do not aim to predict delays, but quantify the overall robustness of a public transport schedule where a large set of delay scenarios is given. The next section will explain the creation of a machine learning-based oracle that can predict the outcome of the robustness test RT-1/2/3/4.

9.3 Creation of Oracle

We want to create a predictor that is specific for each public transport network. In our training dataset, we use a fixed line concept and passenger demand. Similar to the last chapter the combination of timetable and vehicle schedule is called an instance. For all instances contained in the dataset we apply the robustness tests and obtain the corresponding robustness values. The learning task is to predict robustness values for instances.

In contrast to the last chapter, we do not use the total delay of the experiments as robustness values. Using these numbers leads to error rates that are hard to interpret. Instead, we scale the robustness values so the worst-known instance has a robustness value of 100. With this representation, it is easier to compare the quality of the estimations across robustness tests.

The complete data contained in the timetable, the vehicle schedule, and the corresponding passenger routes is huge, partially redundant, and different instances can have different input lengths. These are not desirable attributes for machine learning algorithms. We prefer to work with selected key features instead. These features act as a kind of fingerprint and characterize the instances. The importance of feature selection and feature engineering to the success of machine learning is well recognized, see e.g. [KJ19].

9.3.1 Choosing key features.

Selecting key features representing our public transport instances is challenging because several factors influence different aspects of robustness. A feature selected to help in the prediction of robustness has to serve two main goals. The first goal is to give direct information about data associated with robustness. These features give information like buffer on activities or average rate of occupation on edges. The other goal of including data into the feature set is to provide context that can link features or give a weight during machine learning. For example, the amount of slack on transfers at a specific station is more useful with the context that there is a massive amount of transfers taking place there compared to other stations. During training, the predictors learn the important associations between features. In image processing, e.g., neural networks learn about the context of a 2D pixel even if the 2D image is converted into a 1D vector before processing. Other considerations for the selection process were the following. The features should:

- be numeric or labels that can be mapped to numerical values, see e.g. [Ras15],
- capture non-trivial aspects of the public transport system,
- be chosen so that common information or information that is likely to be similar between instances is under-represented while taking care to include differences, see e.g. [GBC16].

Key features are grouped into ordered sets of fixed size. This means including average slack or buffer per station will create a feature set of size n, where n is the number of stations. This set can be seen as an ordered vector containing the features. In our model one instance produces nine vectors containing key features.

These vectors contain information about the data in a network with n stations and m network edges. Table 9.1 shows a description of these feature vectors. Due to the nature of some features like travel time distribution, there has to be an upper limit. A passenger with a travel time of $\text{traveltime}^{\text{max}}+1$ minutes is counted in the last vector entry. In the experiments conducted for this study, only the distribution for turn around buffer of vehicles produced entries in the last element of such a vector. And in this case, the entry contains the number of turnaround edges with sufficient buffer to absorb any realistic delay. All nine vectors are joined to one large vector as input for the machine learning procedure.

#	description	# elements
1	the avg. occupancy rate of the corr. vehicle in percent for each drive activity	m
2	the number of passenger groups with a perceived travel time of i minutes	$\text{traveltime}^{\text{max}}$
3	the share of passengers with i transfers	$\#\text{transfers}^{\text{max}}$
4	the average buffer on wait activities per station	n
5	the average slack on transfer activities per station	n
6	the share of transfers happening per station	n
7	the average sum of line frequencies per station	n
8	the share of events happening per station	n
9	the number of trips with an outgoing turnaround buffer of i minutes	$\text{turnaround}^{\text{max}}$

Tab. 9.1: Key features of an instance [MH+22].

9.3.2 Choosing suitable ML models.

Using the set of key features and producing multiple values estimating the outcome of the robustness experiments is the task for the machine learning algorithms. We suspect that the relationship between input and output is highly non-linear and to train the algorithm we are using supervised learning. The methods of choice for this situation are artificial neural networks (ANN) and regression-based methods like support vector regression (SVR), or gradient boosting [GBC16]. If our assumptions about the relationship between input and output are correct ANN and SVR with a non-linear kernel should outperform other machine-learning techniques.

Independent of our assumptions, we will evaluate many common models used in similar regression scenarios [Mah+18; Fan+19]. For all of those techniques, we used Python and open-source libraries. The data and scripts for our oracle generation have been publicly available so that the parameters and libraries used are transparent, see [Mül+21a].

The design of the ANN requires a more extensive process than the other methods used. This is because the structure of the ANN needs to be specified. This concerns the number of layers, neurons per layer, and the structure of connections between layers as basic parameters. Other important aspects of the neural network are what activation function is used and how training is done. We use a model with a fixed number of layers and neurons. Every neuron of one layer is connected to every neuron of the previous layer. This is often called a *feedforward neural network* [Sch15]. If this network has multiple layers it is usually called a *multi-layer perceptron* (see Figure 9.2).

9.3.3 Data for Machine Learning

For the experiment with the framework to improve the robustness of instances, we use the following data sets. The artificial benchmark datasets `grid` and `ring` (see [Ptn]), the bus system in Göttingen, Germany (`goevb`) and the regional train network in southern Lower Saxony, Germany (`lowersaxony`). They can be found as part of the open-source software library LinTim, see [Lin]. These datasets contain the infrastructure as well as the passenger demand directly as parameters while the line concept is generated by LinTim using the basic cost-oriented approach of [Sch12]. In contrast to the 5×5-grid of the previous chapter, this version has more details with 9×9 stops using a more detailed demand matrix.

For a graphical representation of the infrastructure of the four datasets, see Figures 9.3-9.6. Considering urban bus networks and rail networks allows us to investigate the usefulness of our approach for different settings. Bus networks typically have a smaller vehicle capacity and a wide range from low to high line frequencies. The `lowersaxony` rail instance has lower frequencies but a significantly higher vehicle capacity. For this, parameters such as transfer weight in the perceived travel time were adapted accordingly.

To find the initial public transport schedules used to train our model, we used state-of-the-art methods from mathematical public transport planning implemented in LinTim to compute several thousand instances for each dataset. As timetabling methods we used solution procedures

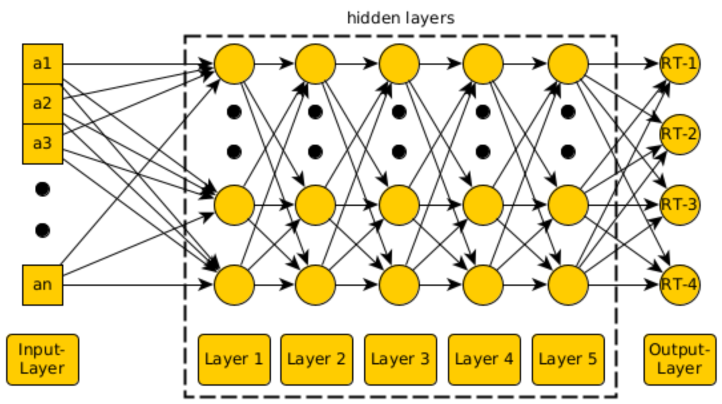


Fig. 9.2: Neural network with five hidden layers [Mül+21b].

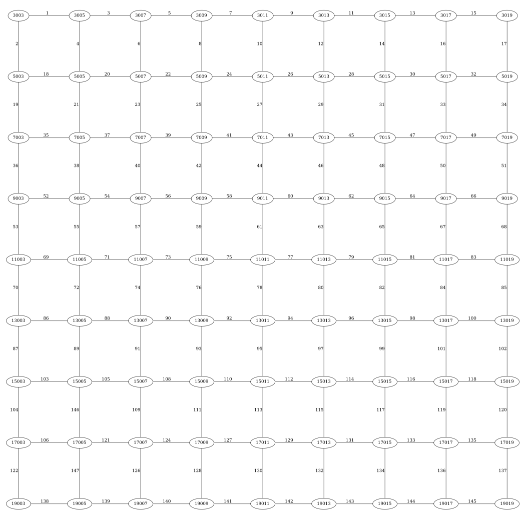


Fig. 9.3: The `grid` network as 9×9 version

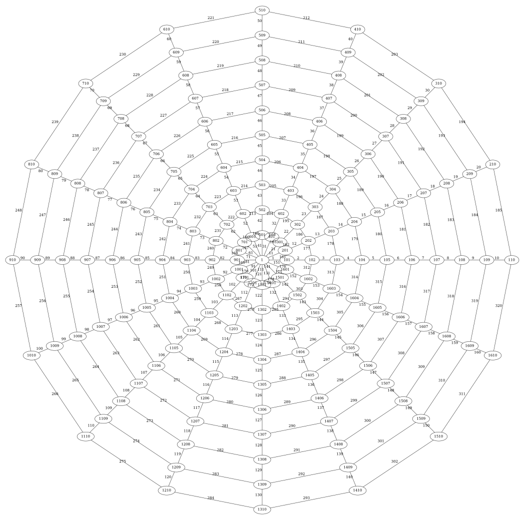


Fig. 9.4: The `ring` network

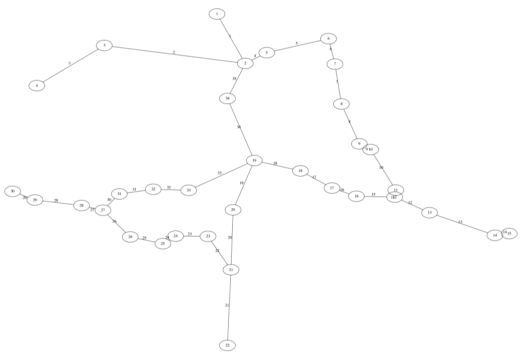


Fig. 9.5: The `lowersaxony` network

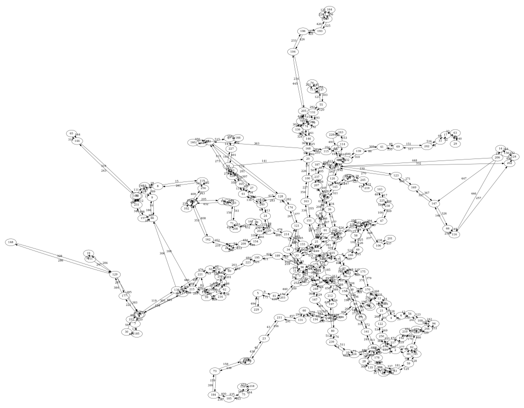


Fig. 9.6: The `goevb` network

for the classic PESP model (as usual in periodic timetabling), in particular, a cycle based integer programming (IP) formulation [PK01] as exact solution method and the fast MATCH heuristic [Pät+17], coupled with different strategies for distributing buffer times such as adding e.g. an exponential buffer distribution on all activities or a proportional buffer on the wait activities in the most used stations. Vehicle scheduling was done by solving the standard IP flow formulation [BK09] which minimizes a weighted sum for the operational costs and uses several adaptions such as fixing possible vehicle schedules before the timetabling step [Pät+17] or different values for the minimal turnaround times of vehicles. To allow reproducibility, all instances created for training along with their corresponding LinTim parameters used for the creation are published, see [Mül+21a]. Afterwards, these instances were evaluated using the robustness tests RT-1/2/3/4. We use these tests to obtain the values to learn from the key features presented in Section 9.3.1. In our experiments, the maximal values were 240 minutes for the maximal travel time $\text{traveltime}^{\max}$, 10 for the maximal number of transfers $\#\text{transfers}^{\max}$, and 30 minutes for the maximal turnaround time $\text{turnaround}^{\max}$. These values are highly dependent on the type of datasets used, e.g., the maximal travel time probably needs to be higher when considering long-distance railway networks. All robustness test results and their corresponding key features can be found at [Mül+21a].

9.3.4 Machine Learning Results

Before the machine learning-based oracle can be used in our framework, we need to determine which models have sufficient quality for this task. To measure the quality of estimation for one robustness test we use the average error and the standard deviation between the real robustness and the estimated robustness. Since we have four robustness tests, this produces four error rates. As a simplification in the evaluation, we will use one error rate as the mean value of the error produced by all four robustness tests. Before presenting the results of our tested models we will introduce all tested models and their parameter settings.

We tested several machine learning methods for estimating the robustness values. The libraries we used for the ANN are NumPy [Oli16], a python library for handling large multidimensional arrays and matrices, TensorFlow [Tea15], a library for machine learning where computations are expressed as stateful dataflow graphs, and Keras [Cho15], a python library for artificial neural networks that acts as an interface for TensorFlow. Additionally, we used several models that are part of the scikit-learn open-source library [Ped+11]. More specifically, we used Logistic Regression, Decision Tree Regression, Gradient Boosting Regression, Bayesian Ridge Regression, Elastic Net Regression, and Support Vector Regression. In addition to scikit-learn we used CatBoost [Pro+19] and XGBRegressor [CG16].

For each of these models, one or more hyperparameters have to be optimized to achieve their best accuracy possible. In principle, one could optimize the hyperparameters of each model independently for each of our four datasets and each of the four robustness tests, possibly yielding 16 different sets of parameters for each model. We refrained from this and kept parameter tuning to a minimum. Unless otherwise stated, we kept the default hyperparameter configuration for the respective method.

A notable exception to this approach is the support vector regression (SVR) where we experimented with different kernel functions. We tried different kernels to test the hypothesis that the connection between input and output is non-linear. The best results are obtained with the radial basis function (RBF), while the polynomial kernel with degree three turns out to be the second-best kernel. After testing several configurations, we find that the combination of 150 neurons per hidden layer and five hidden layers that are fully connected yield the best results.

The quality of the prediction in all our models is evaluated using a separation of the data into training- and testing sets in an 4:1 ratio. For ANN, we chose a separation into training-, validation, and testing sets in an 8:1:1 ratio. For the latter, the training operates in many small runs, called epochs, and in each epoch, the network trains on a subset of the whole training data and is evaluated with the validation set to prevent over-fitting.

Name	network type	$\|S\|$	$\|$instances$\|$	error mean	error sd	$\geq 99\%$ accurate	$\geq 95\%$ accurate
`grid`	artificial	80	8304	0.72	0.85	75 %	99 %
`ring`	artificial	161	2768	0.29	0.87	95 %	96 %
`goevb`	real world	257	4152	0.86	1.30	75 %	98 %
`lowersaxony`	real world	35	5536	0.36	0.77	90 %	99 %

Tab. 9.2: This table shows the basic parameters of the studied infrastructure networks ($|S|$ denotes the number of stations), the number of instances created, and the resulting quality of our robustness predictions for SVR (sd denotes the standard deviation). The mean relative error when predicting the robustness of an instance is always below one percent. The last two columns show the percentage of cases with at least 99% and 95% accuracy. Hence, outliers are also relatively rare [MH+22].

We now present the model with the smallest error rate of all tested models. Table 9.2 shows the performance of SVR on all four test networks.

For graphical representation of how predictions of all four robustness tests relate to the calculated values see Figure 9.7. Instances cluster according to the used strategy in buffer distribution.

It is also interesting to see how the other model compair to the SVM model. Figure 9.8 shows the performance of all tested models. While many of the tested models have a fairly good quality, Bayesian ridge regression (mean error 1.75) and elastic net regression (mean error 2.435) performed significantly worse. These two models both belong to a family of linear models which may explain their poorer performance in comparison to other models. During these test we made use of the characteristic that a ANN can have multiple output neurons compaired to other models that can only predict one outcome at a time. This is why we created one ANN with four distinct models for each robustness test only (ANN four nets) and one integrated model with four output neurons (ANN one net). Interestingly, the integrated approach with four output neurons performed better on the test set. Support Vector Regression (SVR) yields the smallest mean error of all models tested. However, XGBoost comes very close while using fewer parameters. XGBoost, Catboost or SVR show equally good results. Therefore, we applied an additional test. While all performed similarly on the testing data it can not be predicted how they will perform on new data generated outside the cluster of solutions LinTim created. To test instances outside those clusters we generated further instances by the local search procedure described in Section 9.4. The method with the smallest mean error outside the starting clusters was also SVR. This is the reason why this will be the model investigated for the remaining section. As mentioned earlier, these results can be further improved by thorough hyperparameter optimization. Now that the quality of our robustness oracle satisfies our expectations there are only a few other things to consider before using the oracle for our optimization framework.

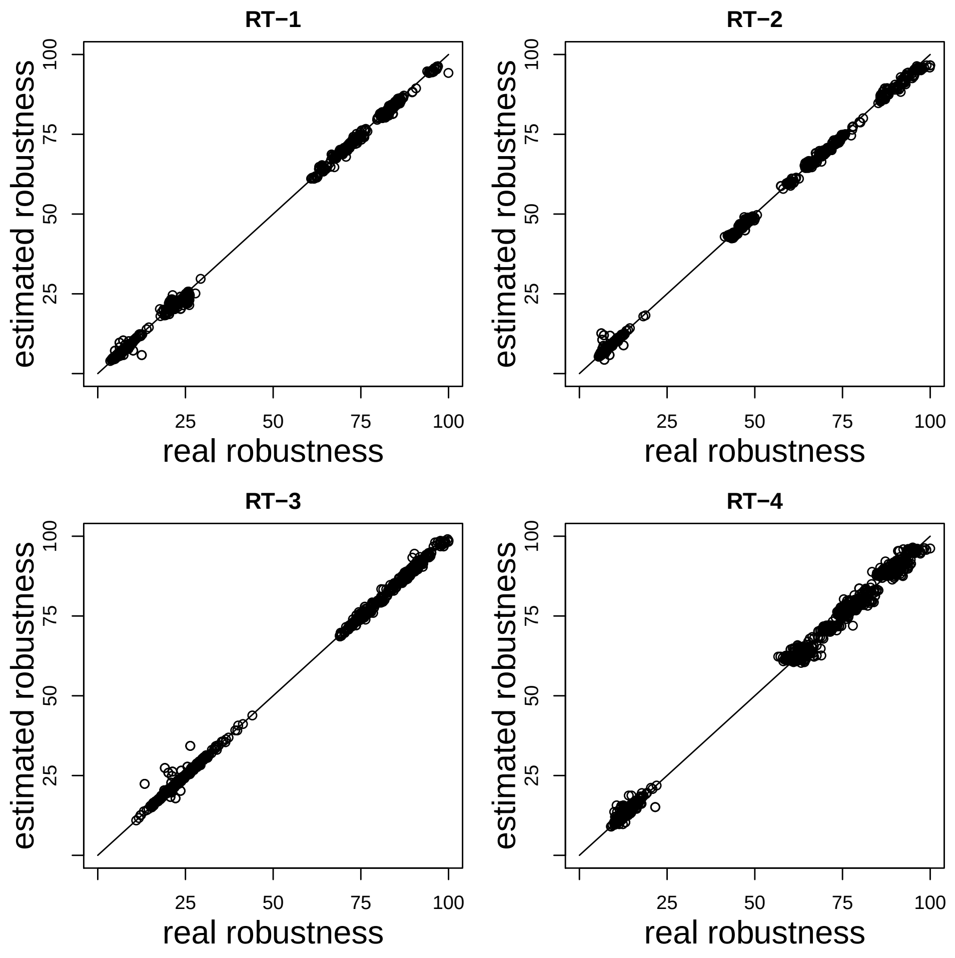


Fig. 9.7: Predictions of all four robustness tests for all instances of the `grid` network. Real robustness refers to the result of a complete simulation.

Because we presented a new approach that uses a set of key features, the impact of those features remains to be investigated, so we do not waste resources on them in the future.

9.3.5 A critical analysis of our key features.

To answer the question of whether a key feature is useful or not we conducted the following set of two experiments. These experiments deliberately contain feature set seven *(the average sum of line frequencies per station)*, although the lines and their frequencies are fixed in all of our experiments. We designed this feature set with the intention of using it in the future, when the line concept may not be fixed any more. In this experiment, it serves as a control feature that shows how an unnecessary feature will behave.

In the first experiment, we trained our SVR model with all but one feature set. The results of tests in which one feature set was omitted in each case are shown in Figure 9.9. The interesting finding is that only neglecting feature set two (the perceived travel time distribution of passengers) and feature set nine (the distribution of the available turnaround buffer between subsequent

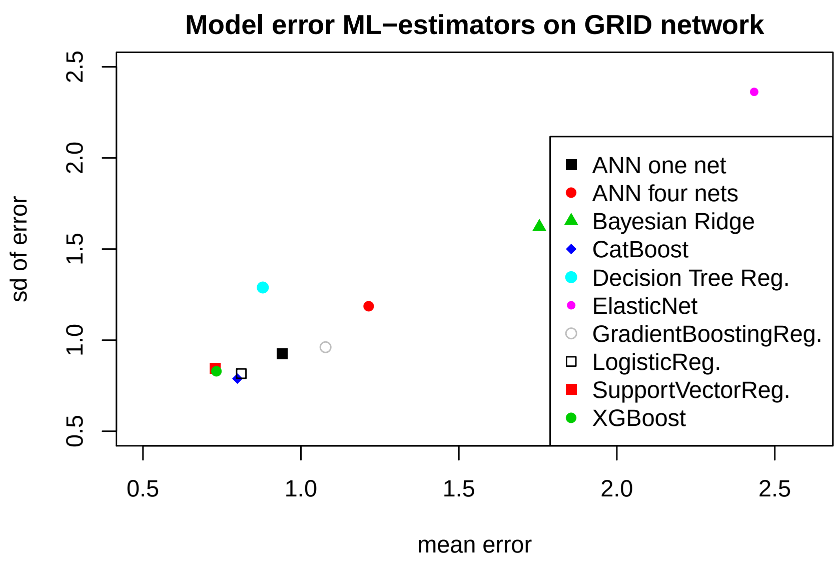


Fig. 9.8: Comparison of multiple regression models to estimate the results of all four robustness tests. Here we look at the mean error and standard deviation for the dataset *grid* [MH+22].

trips) results in a significant performance loss. As anticipated, feature set seven (the distribution of line frequencies) does not influence the result. Neglecting other features results in a very small loss of precision. One reason for this behavior is that there might be some kind of redundancy between the other feature sets.

In the second experiment, we trained our SVR model using only single feature sets, see Figure 9.10. The outcome is very different from the other experiment. The feature set nine, for example, is relatively bad for estimating the robustness by itself while neglecting this feature resulted in a significant loss of accuracy. Surprisingly, feature set eight (the distribution of events to stations) is most accurate in predicting the robustness if taken alone, although it does not contain any information about buffer, occupation, or travel time. This feature set seems to act as a fingerprint of the structure of the instance. It can be used for comparing the similarity of instances. We conclude that feature sets two and nine are essential for a good estimation. Other features (except feature 7) only yield small benefits in this setting.

An evaluation of new instances created in Section 9.3.4 revealed that single feature models result in larger errors in the estimation. Features set eight alone, for example, produced mean errors twice to five times larger compared to the complete model.

9.3.6 How many instances are needed to achieve a good quality?

Training any machine learning model requires data. How many datasets are necessary to achieve a certain degree of performance is a question that should be answered to avoid investing too much time or storage space for training data. In this experiment, we limited the number of instances available for training and tested the performance prediction for the robustness with the remaining instances. Figure 9.11 shows how fast the average error rate drops with an increasing

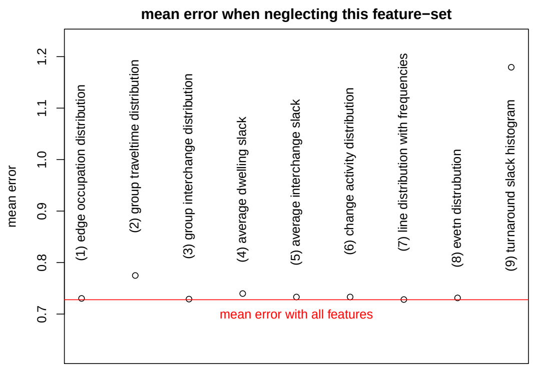


Fig. 9.9: Performance of estimation when one feature set is not used(`grid` network) [MH+22]

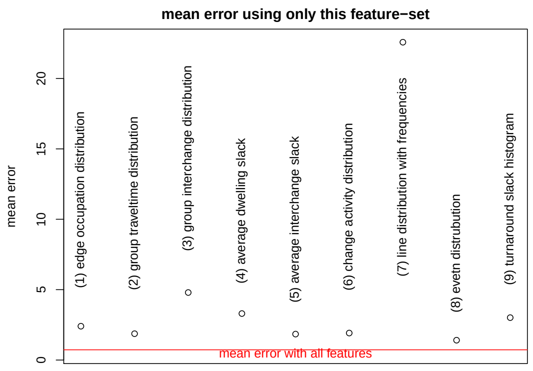


Fig. 9.10: Performance of estimation when only one feature set is used(`grid` network) [MH+22]

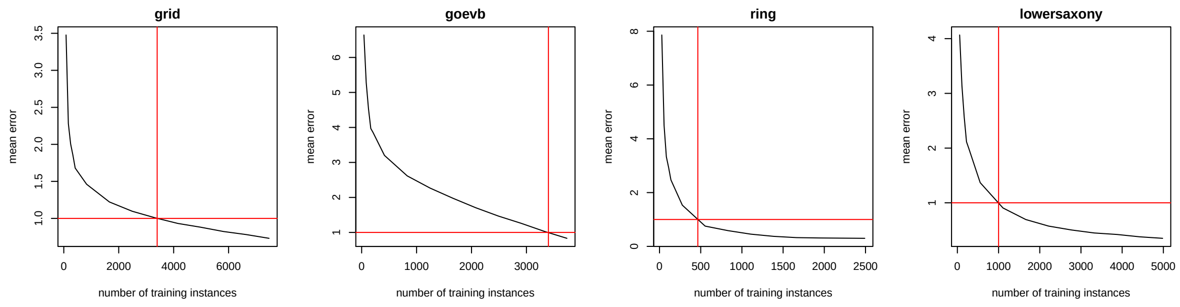


Fig. 9.11: Number of instances necessary until the average error converges for the SVR model [MH+22].

number of training instances. For the instances of the `grid` and `goevb`, we observe that more than 3400 instances are necessary to achieve an average error below 1%, thus for a reliable prediction using our set of key features. The other two smaller networks showed similar behavior, but only needed 500 and 1000 instances. This shows the high initial cost of our proposed machine learning approach which needs to be compensated by a time benefit for later invocations of the model to justify our approach. Having found the preferred model and investigated several aspects of features and training, we now present the application of the oracle inside our local search framework to optimize the robustness of instances.

9.4 Application: A local search framework

With a working robustness oracle, we can now present a first application in which we introduce a simple local search algorithm for robustness optimization. We begin with a starting solution and adapt the timetable with each iteration. As shown in Figure 9.1 the local search terminates when there is no solution found that improves the robustness. We now present a more detailed description of the local search algorithm.

Algorithm 3: Local search using machine learning[MH+22]

```
Data: the starting solution currentSolution
currentValue = evaluateByOracle(currentSolution)
while true do
    bestImprovement = $\emptyset$
    bestValue = $\infty$
    foundImprovement = False
    Compute local neighborhood of currentSolution
    if Rerouting step? then
        Reroute all passengers and update currentSolution
        currentValue = evaluateByOracle(currentSolution)
    end
    for newSolution in local neighboorhood do
        introduceAdditionalBuffer(newSolution)
        value = evaluateByOracle(newSolution)
        if passengerUtility(newSolution) too bad then
            continue
        end
        if value < bestValue then
            bestValue = value
            bestImprovement = newSolution
        end
    end
    if currentValue > bestValue then
        currentValue = bestValue
        currentSolution = bestImprovement
        foundImprovement = true
    end
    if not foundImprovement then
        break
    end
end
```

The algorithm itself is published as part of the open-source software library LinTim, see [Lin]. Algorithm 3 starts with an instance as a starting solution, i.e., some infrastructure and passengers' demand, a line concept, a timetable, a vehicle schedule, and corresponding passengers' routes. Infrastructure, the passengers' demand, and the line concepts are fixed in the following experiments. During the experiments, the timetable, the vehicle schedule, and the corresponding routes are changed. Similar to the previous sections, the combination of an aperiodic timetable, a vehicle schedule and passengers' paths is called a *solution* or *instance*. At the center of the local search is the generation of a neighborhood for the current solution. For this solution, the oracle estimates the robustness of all elements. Only the best element is chosen as the next solution. There may be also solutions with slightly better robustness but a significant loss in perceived traveling times that are neglected.

For a given instance we define a neighborhood of $4N$ elements as follows: We first find the most promising drive, wait, change and turnaround activities by choosing N activities with the smallest current buffer for each activity type. For drive, wait and change activities the buffer is divided by the number of passengers as an additional weight factor. We obtain activities $a_1, \ldots, a_{4N}$.

For each of the chosen activities, $a_1, \ldots, a_{4N}$ we proceed as follows. We increase the buffer of $a_k = (i_k, j_k)$, resulting in a later time for event j_k. This new aperiodic timetable may be infeasible, namely if any of the resulting durations of the activities which start at j_k is smaller than its lower bound. In this case, also the times of the subsequent events have to be increased until the lower bounds of the respective activities are met. This is continued until the increase of the buffer time of a_k is compensated by the sum of buffer times of subsequent activities. Note that the upper bounds of the activities are not checked here, we assume that all events can be postponed arbitrarily.

Afterward, the new public transport schedule is evaluated using the robustness oracle and the best solution in the neighborhood is implemented. Note that we only choose a new solution in our implementation if the passenger utility does not decrease too much in comparison with the start solution. For this, a limit for the amount of passenger utility decrease can be specified by the user, e.g., 10%. For a more thorough investigation about using the oracle within local search algorithms, we refer the reader to [Mül+21b].

For a correct evaluation, a rerouting of the passengers is necessary in every step in order to deliver the correct key features to *oracle* and in order to correctly compute *passengerUtility*. However, to save computation time, the rerouting step is only included every few iterations (in our experiments, we reroute every 10 iterations). In between the rerouting steps, the passengers' routes are assumed to be constant even though the timetable changes.

Apart from this local search variant, there are many other variations possible: the definition of the neighborhood may be changed to improve the local search. The oracle can be improved if the planner has special requirements for the robustness test.

9.5 Results of Optimization Framework

With the implementation of the local search framework and a machine learning-based oracle with high accuracy, we can now present the first results of our framework. There are, however, several things to consider in this evaluation. The starting instance has the most influence on the local search. Parameters like the neighborhood size, and how much additional travel time is acceptable also significantly impact the process. Because the microscopic rerouting during the generation of an instance is expensive this is also only done once in a set interval of optimization steps.

For the first set of experiments, we selected these parameters as follows: neighborhood size of $4N = 80$, a rerouting interval of 10, and a possible travel time increase of 10%

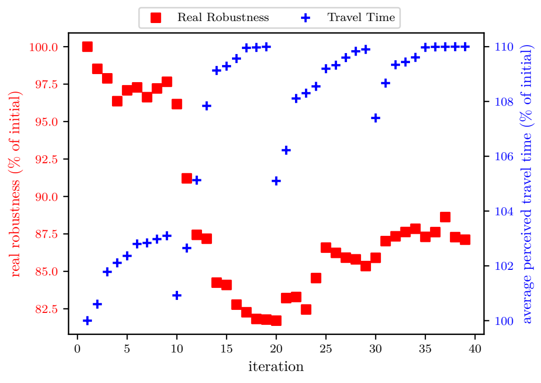


Fig. 9.12: Example instance of the `goevb` network [MH+22]

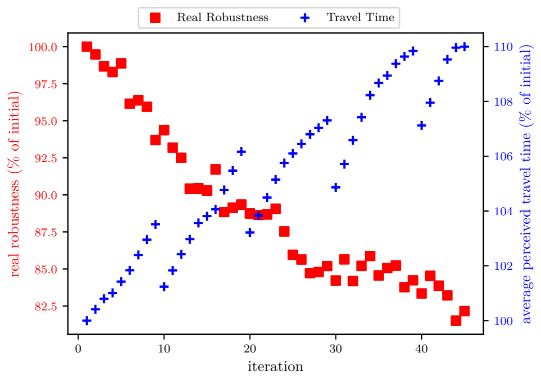


Fig. 9.13: Example instance of the `lowersaxony` network [MH+22]

Typical instances from optimization have good costs and connections for the passengers. These instances often suffer from bad robustness. We used an example for these instances in our experiments. Figure 9.12 shows the behavior of such a starting solution for the local search on the dataset `goevb`. After 40 iterations, the robustness of the instance is improved by 13%, while at the same time the average perceived travel time is increased by 10%. We already mentioned that we are only willing to sacrifice a defined portion of the travel time to gain robustness. This is why the local search finished at this point.

A similar instance from the `lowersaxony` dataset is shown in figure Figure 9.13 with an improvement in robustness of 17.5% while increasing the average perceived travel time again by 10%. Note that we are showing the real robustness here, i.e., the robustness given by simulations of the corresponding instances instead of the predicted value. Furthermore, we see that while the local search only chooses a new solution if the predicted robustness is better than before this must not be the case for the real robustness, i.e., the real robustness observed is not monotone. In both cases, it can be observed the perceived traveling jumps every ten steps. This is because of the rerouting intervals already mentioned. In between, the passenger paths are assumed to be constant even when the public transport schedule changes. This small error, which is corrected every ten steps, is gaining a significant amount of running time.

The local search performs differently when the starting instance already has good robustness. Figures 9.14 (for dataset `grid`) and 9.15 (for dataset `ring`) show such instances. Both instances show, that further improvement is possible. An intriguing behavior can bees in instance `ring` where it was also possible to improve the average travel time in addition to the robustness. The machine learning predictor shows problems once instances get further away from the initial clusters. Since the oracle does not have any real knowledge about the importance to evaluate the quality of estimation. To investigate this effect, we revisit the example from Figure 9.12 and observe not only the reevaluated real robustness but the predicted robustness as well in Figure 9.16. While the real robustness is improved by 13%, the estimated robustness improvement

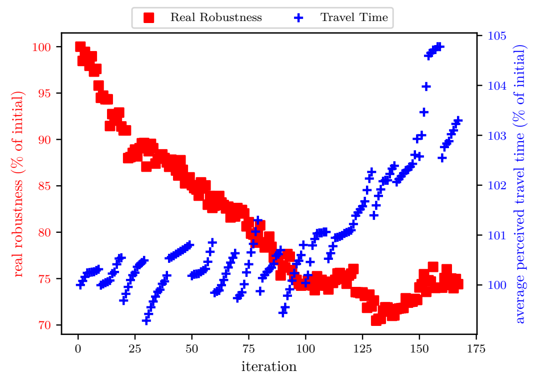


Fig. 9.14: Example instance of the `grid` network [MH+22]

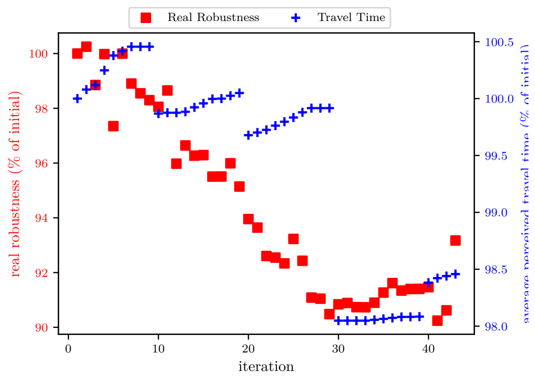


Fig. 9.15: Example instance of the `ring` network [MH+22]

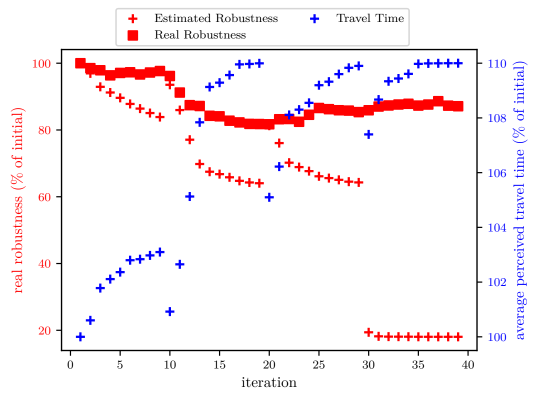


Fig. 9.16: Measuring gap between estimated and real robustness. Example instance of the `goevb` network [MH+22]

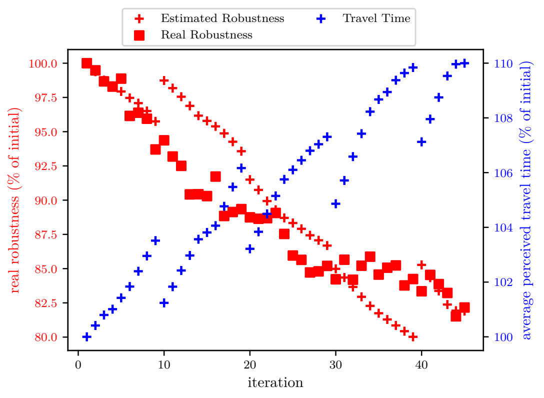


Fig. 9.17: Measuring gap between estimated and real robustness. Example instance of the `lowersaxony` network [MH+22]

is much larger with about 80%. The local search procedure is overestimating the robustness improvement immensely. After the local search is finished it is always important to reevaluate the result using the original robustness test. However this was a rather extreme case, and an average tracking is seen in Figure 9.17, a revisit of Figure 9.13.

Because the re-training of the oracle is relatively fast we can mitigate this effect. In the future, instances containing new structures should be added to the training as soon as their real robustness is known.

Since we are interested in instances that have a good perceived travel time and/or good robustness, there are many instances that are of interest. The final evaluation of the local search shows all starting and finishing solutions in Figure 9.18 and 9.19 where the improvement in the Pareto-front can be seen. Especially for `grid` (Figure 9.18) we are able to produce solutions that

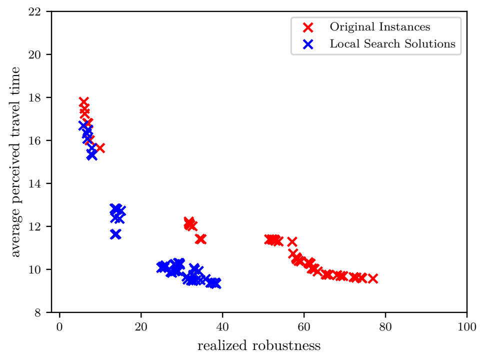


Fig. 9.18: Pareto front of `grid` instances

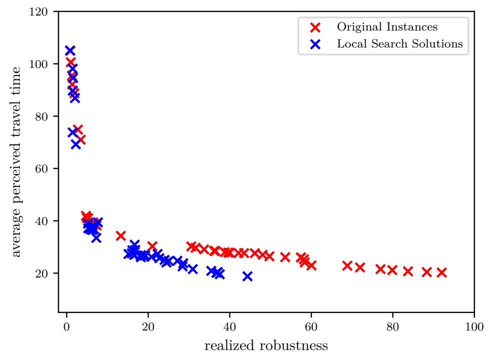


Fig. 9.19: Pareto front of `ring` instances

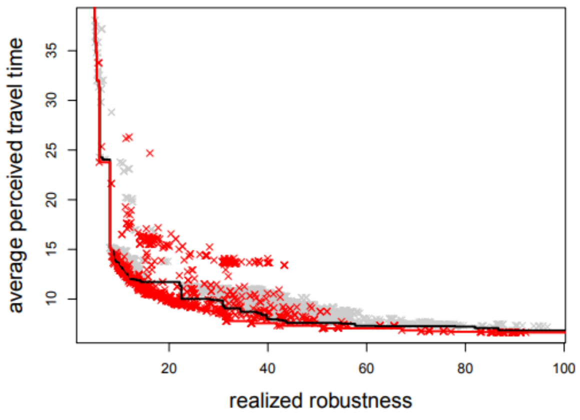
are far better w.r.t. the travel time of the passengers and the robustness. For `ring` (Figure 9.19) the new Pareto-front has a smaller benefit, but a better range of solutions.

In addition to the methods and experiments described in this chapter, we published another set of optimization experiments [Mül+21b]. In this work, we used the *ANN one net* oracle, in contrast, to support vector regression when optimizing our instances. Instead of a deterministic local search approach to improving on starting instances we used a genetic algorithm to get more diverse solutions trying to have more improvements along the whole Pareto-front. Figure 9.20 shows the results of this approach.

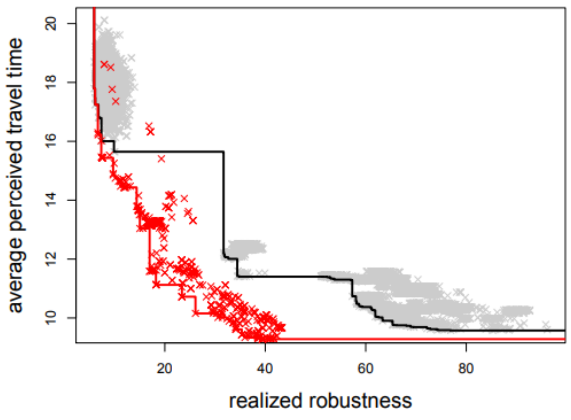
9.6 Conclusion and Impact

In this chapter, we used machine learning to approximate the robustness of a public transport schedule. Approximation was later used to save significant computation time when evaluating robustness during optimization using a local search scheme. Using an SVR model we could indeed predict the robustness of instances near our starting solutions with high accuracy. The optimization using this model when editing the timetable produced more robust solutions while limiting the loss in perceived travel time. Predicting the robustness of instances containing new and different structures compared to the initial clusters of solutions presented a weakness when tracking the robustness.

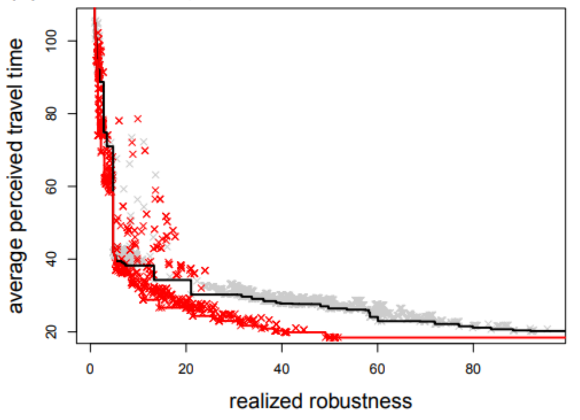
The robustness of a given network may also depend on the delay management strategies. Choosing a different policy compared to a simple no-wait strategy can lead to a different behavior during the local search. With waiting strategies, a robust timetable will have more slack on transfer activities expecting that broken transfers can be fixed by delaying the connecting vehicle. Analyzing the behavior of the oracle and the schedules with a different delay management strategy is a topic for future work. It is also unclear if the oracle can accurately predict the robustness of instances when changes to the line plans are allowed. We know that the line concept affects the robustness as shown in [GSS13; SS13]. Another planning aspect to include



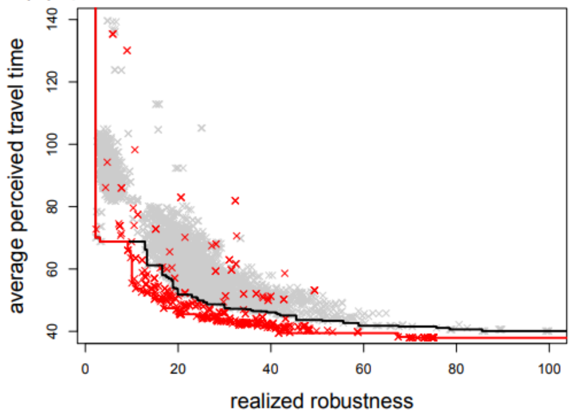
(a) `lowersaxony`.



(b) `grid`.



(c) `ring`.



(d) `goevb`.

Fig. 9.20: Approximated Pareto fronts for the solutions computed by the genetic search. Old instances are grey, new solutions are marked in red [Mül+21b].

would be passenger demand instability, i.e., the influence of the timetable and its robustness on the modal split and therefore on the passenger demand which in turn is an input of the model. There are multiple approaches that can use the oracle (or similar models) as a black box for robustness evaluation. This may include improved local search algorithms but also metaheuristics such as hill-climbing, simulated annealing, or taboo search.

Finally, a promising extension of our approach could be to predict not only the overall robustness of the public transport schedule but additionally to locate where particular weaknesses occur.

Summary and Future Work 10

Look back over the past, with its changing empires
that rose and fell, and you can foresee the future too.

— **Marcus Aurelius [Aur80]**

We now want to gather lessons learned throughout our work. Public transport is an immensely complex system. This system has to be created, executed, maintained, and improved by many different people and processes. We introduced improvements in form of creation and execution using algorithmic approaches. In contrast to other work focusing on runtime and computational complexity alone, our practice is following a slightly different pattern. This pattern is using contemporary methods of algorithms paired with real-world data to gain quantifiable benefits for the passenger. This approach has yielded many exciting insights about improving schedules and operation, which is characterized by the work of dispatchers. The next section will condense our work and findings.

10.1 Combining our Approaches in a Chronological Process

Up until this point, we presented our work in the order of their publications. Now we will go through our improvements in processes concerning public transport in the order they naturally occur during planning and execution. This order is also described in Section 2.1.

10.1.1 Collect, Model, and Create

The only way towards improving the public transport system of the past is to learn from its shortcomings. In addition to using every old data about schedules and infrastructural data, it is vital to study the occurrence and cause of delays so far. We are using this knowledge to our advantage in Section 8.4.7, where one distribution makes the difference between one indicator and a good prediction for the actual average delay. After gathering insights into the past, we then look inside the creation of a new timetable. While the classical paradigm is to create a somehow optimal plan, we introduce a different approach. In Chapter 8, we recommend creating several different solutions, from line plan to vehicle schedule. This creation can be done using sophisticated programs like *LinTim*. A vital part of this process is making sure the data contained essential additional information for simulation tools. We show the level of impact some of that information has on later calculations. This information includes capacities (see Section 8.4.4), the circulation edges contained in the vehicle schedule (see Section 8.4.5), and minimal travel and dwelling times. Those last two are essential because the difference between those times and the actual length of an edge is the time supplement. The variation and use of those time supplements is vital for the robustness of the system (see Section 8.4.2).

10.1.2 Test, Evaluate, Choose, and Repeat

The situation of having several plans to choose from brings us to the second phase of this endeavor; we thoroughly test all solutions. The first step in measuring the robustness is to select a set of tests and parameters. The tests RT-1 to RT-4 may be a good fit for many public transport networks (see Chapter 8). Specialized networks, however, may be treated differently. If the system is lacking stops that resemble hubs RT-3 (the station disruption test) may not produce additional information. Selecting sensible parameters for the tests is also essential. In Section 8.4.2, we explain our choices in a way that can be understood and adapted for other networks.

The evaluation of the results can present quite a challenge. Every run of those robustness experiments produces an enormous amount of data. As an example of this RT-1 in Section 8.4.2 produced 10'000 runs per tested instance (600 runs per parameter). Each run contains delay data and a number of reroutes from every passenger group. We recommend condensing this amount of information to a select number of KPIs representing the desired level of information. Only if the evaluation is representative, understandable, and meaningful, they can support the right choice (see Section 8.4.3). This choice, however, can also lead to the decision to generate another set of instances featuring properties present in outstanding solutions. Starting a new cycle of optimization further improves the robustness as well as knowledge about the system. After a sufficient number of cycles, we end up with a superior solution. Before implementing the plan in the real world, we can prepare for future disruptions.

Chapter 9 shows how we use the knowledge gained from robustness tests to improve solutions during the optimization. There we created a set of training data. This data had special key features and the result of the robustness tests. We implemented a machine learning model that can predict the robustness during optimization with high accuracy as long as the created instance is still similar to other known instances. With these tools, an optimization scheme can improve the robustness without performing time-consuming robustness tests during this process.

10.1.3 Design Policies

Policies like the *waiting time rules* simplify and automate the process of connection dispatching. The creation of a good set of rules is not an easy task. In Section 8.4.8, we show how different instances have very distinct optima for these rules. The process described in that section can be used to come up with good *waiting time rules*. That said, this method might not be detailed enough. An improved technique may produce an individual rule for each connection. Especially in non-rush-our traffic Deutsche Bahn deploys significantly longer connection-specific *waiting time rules*. When these policies are in place, the real-world operation can begin.

10.1.4 Monitor, Decide, and Inform

The daily operation should make use of the technology available to improve the monitoring of the system. The operator should have as much information about the passengers as possible without infringing on their privacy. Reliable data is crucial for any process that involves decisions (see Chapter 6). If this data is available, dispatchers should use software like PANDA to find or support their choices (see Chapter 4).

This system should serve several purposes. Early detection and classification of the situation are one of those. This feature enables possibilities for the information and routing of passengers sure to miss a transfer in the future (see Chapter 5). After the dispatcher safely handles these passengers, he has time to concentrate on the critical decisions. With PANDA, we introduced a comprehensible way to visualize the impact of one choice. We showed that it is also possible to combine one decision with those of its resulting ones (see Chapter 7). Even if those cases are rare, we are now able to decrease the passenger's fear of missing the next train(s).

We hope this process can be used as a template for future guidelines in public transport planning and operation.

10.2 Open Question

However useful our work turns out to be in the coming years, there is more to be desired and hopefully achieved. The biggest problem in the context of the German long-distance railway is the number of primary delays. Reasons for many of those delays are too little staff and maintenance of trains and tracks. Nevertheless, we see at least a little bit of potential for reducing the probability of new delays. We have shown in Section 8.3.6 that crowding caused primary delays. In Section 8.4.4 we discovered that while *LinTim* does not prefer schedules with evenly distributed vehicle utilization the robustness test highlight well-distributed solutions. Combining our findings and methods could, therefore, lead to a small decrease in primary delays.

The biggest shortcomings of our methods and vulnerability for critique is the neglect of infrastructural data. This data has a significant influence on the executability of any dispatching process. We hope to gain access to real-world datasets containing this data in the future.

Another interesting aspect is to seek economic improvements. Quality of service, frequencies of lines, reliability, and operational costs are all factors influencing the price of each service. The changes introduced by us could potentially modify ticket prices. Modified policies for passenger compensation could be an attractive field for investigation. Allowing upgrades or dynamically compensating passengers for switching to a less crowded route could increase overall passenger satisfaction.

In a more recent paper [PRMH22] we integrated another approach dispatchers can use to reduce the overall delay of passengers when one vehicle is heavily delayed. In the paper, we show that it is possible to calculate short turns similar to the PANDA approach. However, we also show the recommendation whether or not a short turn is beneficial or not can be guessed by a machine learning oracle. The oracle achieved a performance of 94% using most information available without performing the actual computation, and a 90% accuracy when using only six features that should be available to most dispatchers.

An apparent extension to the methods described in Chapter 9 is a neural network, which has more detailed information about possible vulnerabilities. To achieve more locally targeted improvements and better robustness optimization we implemented this approach and defined an extensive set of features detailing possible vulnerabilities of the network. While machine learning was able to predict the robustness of know instances with the same quality as the previous model, there was no additional improvement in overall robustness compared to the methods described in Chapter 9.

Another important aspect of rerouting during disruption is the optimal rerouting of passengers in case of (partially) canceled trains. In [MHRS19] we used the RAPTOR algorithm for rerouting and an ILP formulation of the problem as a multi-source multi-commodity unsplittable flow problem to have a prototype where optimal rerouting was possible. Furthermore, we compared the results with a greedy approach to see the difference in performance and quality of the generated solutions. The result was a slightly better quality of the ILP solution while the greedy algorithm runs more than three times faster.

Exploring new methods of public transport will also be a part of future research. Modern technology will give rise to new forms of automated vehicles and processes. Small self-driving busses could dominate future cityscapes. Those technologies should optimize passenger satisfaction as well as minimize their impact on the environment. The reliability of those products should be very high when introduced. This property raises the chance of mass acceptance by the general public.

Bibliography

[AA+11] Rodrigo Acuna-Agost, Philippe Michelon, Dominique Feillet, and Serigne Gueye. „A MIP-based local search method for the railway rescheduling problem“. In: *Networks* 57.1 (2011), pp. 69–86 (cit. on p. 87).

[Ach+08] Tobias Achterberg, Timo Berthold, Thorsten Koch, and Kati Wolter. „Constraint Integer Programming: A New Approach to Integrate CP and MIP“. In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Ed. by Laurent Perron and Michael A. Trick. Berlin, Heidelberg, 2008, pp. 6–20 (cit. on p. 70).

[Aur80] Marcus Aurelius. *Meditations*. 161 - 180 AD. Chap. 9.5 (cit. on pp. 35, 139).

[Bas+16] Hannah Bast, Daniel Delling, Andrew Goldberg, et al. „Route Planning in Transportation Networks“. In: *Algorithm Engineering: Selected Results and Surveys*. Ed. by Lasse Kliemann and Peter Sanders. Vol. 9220. Lecture Notes in Computer Science (LNCS). Cham: Springer International Publishing, 2016, pp. 19–80. URL: `https://doi.org/10.1007/978-3-319-49487-6_2` (cit. on pp. 9, 10).

[Bau+19] Moritz Baum, Valentin Buchhold, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf. „UnLimited TRAnsfers for Multi-Modal Route Planning: An Efficient Solution“. In: *27th Annual European Symposium on Algorithms (ESA 2019)*. Ed. by Michael A. Bender, Ola Svensson, and Grzegorz Herman. Vol. 144. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 14:1–14:16. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/11135` (cit. on p. 13).

[BBOK13] Marco Bender, Sabine Büttner, and Sven O. Krumke. „Online delay management on a single train line: Beyond competitive analysis“. In: *Public Transport* 5 (Oct. 2013), pp. 243–266 (cit. on p. 37).

[Ber+11a] Annabell Berger, Christian Blaar, Andreas Gebhardt, Matthias Müller-Hannemann, and Mathias Schnee. „Passenger Flow-Oriented Train Disposition“. In: *Algorithms – ESA 2011*. Ed. by Camil Demetrescu and Magnús M. Halldórsson. 2011, pp. 227–238 (cit. on pp. 16, 38).

[Ber+11b] Annabell Berger, Andreas Gebhardt, Matthias Müller-Hannemann, and Martin Ostrowski. „Stochastic Delay Prediction in Large Train Networks“. In: *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*. Ed. by Alberto Caprara and Spyros Kontogiannis. Vol. 20. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011, pp. 100–111. URL: `http://drops.dagstuhl.de/opus/volltexte/2011/3270` (cit. on p. 91).

[Bes+16] Nikola Besinovic, Rob M.P. Goverde, Egidio Quaglietta, and Roberto Roberti. „An integrated micro–macro approach to robust railway timetabling“. In: *Transportation Research Part B: Methodological* 87 (2016), pp. 14 –32 (cit. on p. 87).

[Bie06] Claus Biederbick. „Computergestützte Disposition im schienengebundenen Personentransport – ein kundenorientierter Ansatz“. PhD thesis. Universität Paderborn, Dec. 2006 (cit. on pp. 37, 65, 74).

[BK09] Stefan Bunte and Natalia Kliewer. „An overview on vehicle scheduling models“. In: *Public Transport* 1.4 (2009), pp. 299–317 (cit. on p. 126).

[Bor+18] Ralf Borndörfer, Torsten Klug, Leonardo Lamorgese, et al. *Handbook of Optimization in the Railway Industry.* International Series in Operations Research & Management Science. Springer International Publishing, 2018 (cit. on pp. 7, 8, 36).

[Bro] Patrick Brosi. *GTFS.de - Deutschlandweite GTFS Feeds.* URL: `{https://gtfs.de/}` (visited on Sept. 1, 2021) (cit. on p. 30).

[BS07] Claus Biederbick and Leena Suhl. „Decision Support Tools for Customer-Oriented Dispatching“. In: *Algorithmic Methods for Railway Optimization.* Ed. by Frank Geraets, Leo Kroon, Anita Schöbel, Dorothea Wagner, and Christos D. Zaroliagis. 2007, pp. 171–183 (cit. on pp. 37, 65, 74).

[BS14a] Reinhard Bauer and Anita Schöbel. „Rules of Thumb — Practical online strategies for delay management“. In: *Public Transport* 6.1 (2014), pp. 85–105 (cit. on p. 122).

[BS14b] Reinhard Bauer and Anita Schöbel. „Rules of thumb: Practical online-strategies for delay management“. In: *Public Transport* 6 (Apr. 2014), pp. 85–105 (cit. on pp. 37, 50).

[BSS89] Jim Brumbaugh-Smith and Douglas Shier. „An empirical investigation of some bicriterion shortest path algorithms“. In: *European Journal of Operational Research* 43.2 (1989), pp. 216 –224. URL: `http://www.sciencedirect.com/science/article/pii/0377221789902154` (cit. on p. 10).

[CBG15] Alexander Chernev, Ulf Böckenholt, and Joseph Goodman. „Choice overload: A conceptual review and meta-analysis“. In: *Journal of Consumer Psychology* 25.2 (2015), pp. 333–358. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1016/j.jcps.2014.08.002`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1016/j.jcps.2014.08.002` (cit. on p. 18).

[Cc0a] *(CC0 Image - Grafton Train Station, Double Tracking IV.* URL: `{https://commons.wikimedia.org/wiki/File:Grafton_Train_Station,_Double_Tracking_IV.jpg}` (visited on July 3, 2020) (cit. on p. 90).

[Cc0b] *(CC0 Image - track exchange failure snow.* URL: `{https://pixabay.com/photos/track-exchange-failure-snow-224564/}` (visited on July 3, 2020) (cit. on p. 89).

[Cc0c] *(CC0 Image - train railroad rail transport rails.* URL: `{https://pixabay.com/photos/train-railroad-rail-transport-rails-167980/}` (visited on July 3, 2020) (cit. on pp. 91, 92).

[CG16] Tianqi Chen and Carlos Guestrin. „XGBoost: A Scalable Tree Boosting System“. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD ’16. San Francisco, California, USA: ACM, 2016, pp. 785–794. URL: `http://doi.acm.org/10.1145/2939672.2939785` (cit. on p. 127).

[Che09] G. K. Chesterton. *Tremendous Trifles.* 1909 (cit. on p. 1).

[Che+11] Xumei Chen, Shuxia Guo, Lei Yu, and Bruce Hellinga. „Short-term forecasting of transit route OD matrix with smart card data“. In: *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 2011, pp. 1513–1518 (cit. on p. 65).

[Chi18] Chip.de. *DB Zugradar: Deutsche Bahn lässt Live-App sterben*. 2018. URL: {https://www.chip.de/news/DB-Zugradar-Deutsche-Bahn-laesst-Live-App-sterben_150685678.html} (visited on Sept. 1, 2021) (cit. on p. 20).

[Cho15] Francois Chollet. *Keras - Simple. Flexible. Powerful.* 2015. URL: https://keras.io/ (visited on Oct. 10, 2020) (cit. on p. 127).

[Cic+09] Serafino Cicerone, Gianlorenzo D'angelo, Gabriele Di Stefano, Daniele Frigioni, and Alfredo Navarra. „Recoverable robust timetabling for single delay: Complexity and polynomial algorithms for special cases“. In: *Journal of Combinatorial Optimization* 18 (2009), pp. 229–257 (cit. on p. 87).

[CJ18] Oded Cats and Erik Jenelius. „Beyond a complete failure: the impact of partial capacity degradation on public transport network vulnerability“. In: *Transportmetrica B: Transport Dynamics* 6.2 (2018), pp. 77–96 (cit. on p. 122).

[Com07] Wikimedia Commons. *Antwerpen-Noord signal box, during an open day visit. The system shown is EBP (Elektronische Bedien Post), made by Siemens.* 2007. URL: https://commons.wikimedia.org/wiki/File:Antwerpen_Noord_seinhuis.jpg (cit. on p. 36).

[CYv16] Oded Cats, Menno Yap, and Niels van Oort. „Exposing the role of exposure: Public transport network risk analysis“. In: *Transportation Research Part A: Policy and Practice* 88 (2016), pp. 1–14 (cit. on p. 122).

[Dag16] *Dagstuhl Seminar 16171 - Algorithmic Methods for Optimization in Public Transport*. 2016. URL: https://www.dagstuhl.de/en/program/calendar/semhp/?semnr=16171 (visited on May 27, 2019) (cit. on p. 87).

[Db16] *Unofficial Deutsche Bahn Fernverkehr (German Railways Long-Distance Trains) Timetable 2016 GTFS Feed*. Visited on: 2018-10-10. URL: {https://github.com/fredlockheed/db-fv-gtfs} (cit. on p. 30).

[DEL] DELFI. *OpenData ÖPNV - Für Entwickler, die mobil machen!* URL: {https://www.opendata-oepnv.de/} (visited on Sept. 1, 2021) (cit. on p. 30).

[Deu18] Deutsche Bahn. *Ein Jahr nach Start der Schnellfahrstrecke Berlin–München: Bahn ist Verkehrsmittel Nummer 1.* https://www.deutschebahn.com/de/presse/pressestart_zentrales_uebersicht/Ein-Jahr-nach-Start-der-Schnellfahrstrecke-Berlin-Muenchen-Bahn-ist-Verkehrsmittel-Nummer-1-3528088. 2018 (cit. on p. 65).

[Deu21] RedaktionsNetzwerk Deutschland. *Corona-Pandemie: Fahrgastzahlen in Bus und Bahn brechen drastisch ein.* URL: {https://www.rnd.de/wirtschaft/corona-pandemie-fahrgastzahlen-in-bus-und-bahn-brechen-drastisch-ein-VA6OQHKYWX7BSZACGGC3XTYFVY.html} (visited on Apr. 12, 2021) (cit. on p. 96).

[DH11] Twan Dollevoet and Dennis Huisman. „Fast Heuristics for Delay Management with Passenger Rerouting“. In: *Public Transport* 6 (Jan. 2011), pp. 67–84 (cit. on p. 37).

[Dib+13] Julian Dibbelt, Thomas Pajor, Ben Strasser, and Dorothea Wagner. „Intriguingly Simple and Fast Transit Routing“. In: *Experimental Algorithms*. Ed. by Vincenzo Bonifaci, Camil Demetrescu, and Alberto Marchetti-Spaccamela. 2013, pp. 43–54 (cit. on pp. 9, 10).

[Dol+10] Twan Dollevoet, Dennis Huisman, Marie Schmidt, and Anita Schöbel. „Delay Management with Re-Routing of Passengers“. In: *Erasmus University Rotterdam, Econometric Institute, Econometric Institute Report* 46 (Jan. 2010) (cit. on p. 37).

[Dol+18] Twan Dollevoet, Dennis Huisman, Marie Schmidt, and Anita Schöbel. „Delay propagation and delay management in transportation networks“. In: *Handbook of Optimization in the Railway Industry*. Springer, 2018, pp. 285–317 (cit. on p. 37).

[Doy87] Sir Arthur Conan Doyle. *Sherlock Holmes - A Study in Scarlet*. Ward Lock & Co, 1887 (cit. on p. 23).

[DPW12] Daniel Delling, Thomas Pajor, and Renato F. Werneck. „Round-based public transit routing“. In: *In ALENEX*. 2012, pp. 130–140 (cit. on pp. 9, 10).

[Dsb15] *Öffentlicher Personenverkehr 2015: Neuer Höchststand bei Fahr- und Fluggästen*. Visited on: 2018-10-16. URL: `{https://www.destatis.de/DE/PresseService/Presse/Pressemitteilungen/2016/02/PD16_052_461.html}` (cit. on p. 1).

[Eme60] Ralph Waldo Emerson. *The Conduct of Life*. Smith, Elder & Co., 1860 (cit. on p. 73).

[Fan+19] Junliang Fan, Lifeng Wu, Fucang Zhang, et al. „Empirical and machine learning models for predicting daily global solar radiation from sunshine duration: A review and case study in China“. In: *Renewable and Sustainable Energy Reviews* 100 (2019), pp. 186–212 (cit. on p. 124).

[FAZ19] FAZ Manfred Köhler. *Polizei hilft bei überfüllten Zügen*. URL: `{https://www.faz.net/aktuell/rhein-main/frankfurt/deutsche-bahn-polizei-hilft-bei-ueberfuellten-zuegen-1941811.html}` (visited on July 28, 2019) (cit. on p. 21).

[Foc21] Dieter Fockenbrock. *Bahn faehrt Rekordverlust ein und erwartet erst 2022 wieder Gewinn*. URL: `{https://www.handelsblatt.com/unternehmen/handel-konsumgueter/jahresbilanz-2020-bahn-faehrt-rekordverlust-ein-und-erwartet-erst-2022-wieder-gewinn/27040154.html}` (visited on Apr. 12, 2021) (cit. on p. 96).

[FOR2083] *DFG - Forschungsgruppe 2083: Integrierte Planung im öffentlichen Verkehr*. URL: `{https://for2083.mathematik.uni-kl.de/}` (visited on July 11, 2022) (cit. on p. 39).

[Fri+17a] Markus Friedrich, Maximilian Hartl, Alexander Schiewe, and Anita Schöbel. „Angebotsplanung im öffentlichen Verkehr - planerische und algorithmische Lösungen“. In: *Heureka'17*. 2017 (cit. on pp. 14, 15, 98, 100).

[Fri+17b] Markus Friedrich, Matthias Müller-Hannemann, Ralf Rückert, Alexander Schiewe, and Anita Schöbel. „Robustness Tests for Public Transport Planning“. In: *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*. Ed. by Gianlorenzo D'Angelo and Twan Dollevoet. Vol. 59. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 6:1–6:16. URL: `http://drops.dagstuhl.de/opus/volltexte/2017/7890` (cit. on pp. 2, 3, 28, 88, 99, 101–103, 105, 106, 108).

[Fri+18] Markus Friedrich, Matthias Müller-Hannemann, Ralf Rückert, Alexander Schiewe, and Anita Schöbel. „Robustness as a Third Dimension for Evaluating Public Transport Plans“. In: *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*. Ed. by Ralf Borndörfer and Sabine Storandt. Vol. 65. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 4:1–4:17. URL: `http://drops.dagstuhl.de/opus/volltexte/2018/9709` (cit. on pp. 2, 3, 88, 108, 110–112, 114–120).

[Fuh2019] Cornelia Fuhrmann. *Praxisprojekt mit der Deutschen Bahn Anschluss mit Panda*. URL: `{https://www.mz-web.de/mitteldeutschland/praxisprojekt-mit-der-deutschen-bahn-anschluss-mit-panda-3012944}` (visited on July 11, 2019) (cit. on p. 47).

[Gat+04] Michael Gatto, Riko Jacob, Leon Peeters, and Peter Widmayer. „Online Delay Management on a Single Train Line“. In: Jan. 2004, pp. 306–320 (cit. on p. 37).

[Gat+07] Michael Gatto, Riko Jacob, Leon Peeters, and Peter Widmayer. *Online Delay Management on a Single Train Line*. Ed. by Frank Geraets, Leo Kroon, Anita Schoebel, Dorothea Wagner, and Christos D. Zaroliagis. 2007, pp. 306–320 (cit. on p. 37).

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016 (cit. on pp. 123, 124).

[GHS17] Philine Gattermann, Jonas Harbering, and Anita Schöbel. „Line Pool Generation“. In: *Public Transport* 9.1-2 (2017), pp. 7–32. URL: `http://dx.doi.org/10.1007/s12469-016-0127-x` (cit. on pp. 99, 104).

[GM01] Francesca Guerriero and Roberto Musmanno. „Label Correcting Methods to Solve Multicriteria Shortest Path Problems“. In: *Journal of Optimization Theory and Applications* 111.3 (Dec. 2001), pp. 589–613. URL: `https://doi.org/10.1023/A:1012602011914` (cit. on p. 10).

[GS10] Marc Goerigk and Anita Schöbel. „An Empirical Analysis of Robustness Concepts for Timetabling“. In: *10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'10)*. Ed. by Thomas Erlebach and Marco Lübbecke. Vol. 14. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010, pp. 100–113. URL: `http://drops.dagstuhl.de/opus/volltexte/2010/2753` (cit. on pp. 85, 88).

[GS13] Marc Goerigk and Anita Schöbel. „Improving the Modulo Simplex Algorithm for Large-Scale Periodic Timetabling“. In: *Computers and Operations Research* 40.5 (2013), pp. 1363–1370 (cit. on p. 99).

[GSS13] Marc Goerigk, Michael Schachtebeck, and Anita Schöbel. „Evaluating Line Concepts using Travel Times and Robustness: Simulations with the Lintim toolbox“. In: *Public Transport* 5.3 (2013), pp. 267–284 (cit. on pp. 8, 27, 88, 99, 136).

[Hac] *HACON – A Siemens Company - References*. URL: `{https://www.hacon.de/en/company/#c8922}` (visited on Sept. 1, 2022) (cit. on p. 25).

[Haf] *Hafas Rohdaten Format (HRDF)*. URL: `{https://opentransportdata.swiss/de/cookbook/hafas-rohdaten-format-hrdf/}` (visited on Sept. 1, 2022) (cit. on p. 25).

[HGL08] Géraldine Heilporn, Luigi De Giovanni, and Martine Labbé. „Optimization models for the single delay management problem in public transportation“. In: *European Journal of Operational Research* 189.3 (2008), pp. 762 –774 (cit. on p. 87).

[HK15] Luke Haywood and Martin Koning. „The distribution of crowding costs in public transport: New evidence from Paris“. In: *Transportation Research Part A: Policy and Practice* 77 (2015), pp. 182 –201. URL: `http://www.sciencedirect.com/science/article/pii/S096585641500083X` (cit. on p. 96).

[IO21] MobilityData IO. *OpenMobilityData*. URL: `{https://transitfeeds.com/}` (visited on Sept. 1, 2021) (cit. on p. 30).

[KAA21] Masood Jafari Kang, Shervin Ataeian, and Sayyed Mahdi Amiripour. „A procedure for public transit OD matrix generation using smart card transaction data“. In: *Public Transport* 13 (2021), pp. 81–100 (cit. on p. 20).

[Kan+11] Satoshi Kanai, Koichi Shiina, Shingo Harada, and Norio Tomii. „An optimal delay management algorithm from passengers' viewpoints considering the whole railway network“. In: *Journal of Rail Transport Planning and Management* 1.1 (2011). Robust Modelling of Capacity, Delays and Rescheduling in Regional Networks, pp. 25 –37. URL: `http://www.sciencedirect.com/science/article/pii/S2210970611000060` (cit. on p. 41).

[KJ19] Max Kuhn and Kjell Johnson. *Feature Engineering and Selection — A Practical Approach for Predictive Models*. CRC Press, Taylor & Francis Ltd, 2019 (cit. on p. 123).

[KL17] Somkiat Kosolsombat and Wasit Limprasert. „Arrival Time Prediction and Train Tracking Analysis“. In: *Trends in Artificial Intelligence: PRICAI 2016 Workshops*. Ed. by Masayuki Numao, Thanaruk Theeramunkong, Thepchai Supnithi, et al. Vol. 10004. Lecture Notes in Computer Science (LNCS). Cham: Springer International Publishing, 2017, pp. 170–177 (cit. on p. 52).

[Kou+14] Marco Kouwenhoven, Gerard C. de Jong, Paul Koster, et al. „New values of time and reliability in passenger transport in The Netherlands“. In: *Research in Transportation Economics* 47 (2014). Appraisal in Transport, pp. 37 –49. URL: `http://www.sciencedirect.com/science/article/pii/S0739885914000584` (cit. on p. 18).

[Kro+08] Leo Kroon, Gábor Maróti, Mathijn Retel Helmrich, Michiel Vromans, and Rommert Dekker. „Stochastic improvement of cyclic railway timetables“. In: *Transportation Research Part B: Methodological* 42.6 (2008), pp. 553 –570 (cit. on p. 87).

[LC17] Lijuan Liu and Rung-Ching Chen. „A novel passenger flow prediction model using deep learning methods“. In: *Transportation Research Part C: Emerging Technologies* 84 (2017), pp. 74–91. URL: `https://www.sciencedirect.com/science/article/pii/S0968090X17302024` (cit. on p. 20).

[LC20] Nuannuan Leng and Francesco Corman. „The role of information availability to passengers in public transport disruptions: An agent-based simulation approach“. In: *Transportation Research Part A: Policy and Practice* 133 (2020), pp. 214–236. URL: `https://doi.org/10.1016%2Fj.tra.2020.01.007` (cit. on p. 93).

[Lem+14] Martin Lemnian, Ralf Rückert, Steffen Rechner, Christoph Blendinger, and Matthias Müller-Hannemann. „Timing of Train Disposition: Towards Early Passenger Rerouting in Case of Delays“. In: *14th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*. Ed. by Stefan Funke and Matúš Mihalák. Vol. 42. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014, pp. 122–137. URL: `http://drops.dagstuhl.de/opus/volltexte/2014/4757` (cit. on pp. 2, 3, 15, 38, 41, 51, 54, 56–59).

[Lie+09] Christian Liebchen, Marco Lübbecke, Rolf Möhring, and Sebastian Stiller. „The Concept of Recoverable Robustness, Linear Programming Recovery, and Railway Applications“. In: vol. 5868. Oct. 2009, pp. 1–27 (cit. on pp. 2, 85).

[Lin] *LinTim - Integrated Optimization in Public Transportation. Homepage.* see http://www.lintim.net/ (cit. on pp. 27, 99, 107, 125, 132).

[Lin10] Tzu-Ping Lin. „Carbon dioxide emissions from transport in Taiwan's national parks“. In: *Tourism Management* 31.2 (2010), pp. 285 –290. URL: `http://www.sciencedirect.com/science/article/pii/S026151770900051X` (cit. on p. 1).

[LMHR16] Martin Lemnian, Matthias Müller-Hannemann, and Ralf Rückert. „Sensitivity Analysis and Coupled Decisions in Passenger Flow-Based Train Dispatching“. In: *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016).* Ed. by Marc Goerigk and Renato Werneck. Vol. 54. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, 2:1–2:15. URL: `http://drops.dagstuhl.de/opus/volltexte/2016/6526` (cit. on pp. 2, 3, 38, 49–51, 53, 65, 67, 71, 72, 76, 77, 80).

[LS73] Allen L. Soyster. „Technical Note–Convex Programming with Set-Inclusive Constraints and Applications to Inexact Linear Programming“. In: *Operations Research* 21 (Oct. 1973), pp. 1154–1157 (cit. on p. 85).

[Luo+21] Dan Luo, Dong Zhao, Qixue Ke, et al. „Fine-Grained Service-Level Passenger Flow Prediction for Bus Transit Systems Based on Multitask Deep Learning“. In: *IEEE Transactions on Intelligent Transportation Systems* 22.11 (2021), pp. 7184–7199 (cit. on p. 20).

[Mah+18] Mohammad Saeid Mahdavinejad, Mohammadreza Rezvan, Mohammadamin Barekatain, et al. „Machine learning for internet of things data analysis: a survey“. In: *Digital Communications and Networks* 4.3 (2018), pp. 161–175 (cit. on p. 124).

[Mar84] Ernesto Queirós Vieira Martins. „On a multicriteria shortest path problem“. In: *European Journal of Operational Research* 16.2 (1984), pp. 236 –245. URL: `http://www.sciencedirect.com/science/article/pii/0377221784900778` (cit. on p. 10).

[Mat+18] Gonçalo Matos, Luis Albino, Ricardo Saldanha, and Ernesto Morgado. „Solving periodic timetabling problems with SAT and machine learning“. In: *Proceedings of CASPT 2018.* 2018 (cit. on p. 122).

[Mat+21] Gonçalo Matos, Luis Albino, Ricardo Saldanha, and Ernesto Morgado. „Solving periodic timetabling problems with SAT and machine learning“. In: *Public Transport* 13 (2021), pp. 625–648 (cit. on p. 122).

[MH+22] Matthias Müller-Hannemann, Ralf Rückert, Alexander Schiewe, and Anita Schöbel. „Estimating the robustness of public transport schedules using machine learning“. In: *Transportation Research Part C: Emerging Technologies* 137 (2022), p. 103566. URL: `https://www.sciencedirect.com/science/article/pii/S0968090X22000146` (cit. on pp. 2, 3, 121, 122, 124, 128, 130–132, 134, 135).

[MHR17] Matthias Müller-Hannemann and Ralf Rückert. „Dynamic Event-Activity Networks in Public Transportation“. In: *Datenbank-Spektrum* 17.2 (2017), pp. 131–137 (cit. on pp. 2, 3, 18, 40, 43).

[MHRS19] Matthias Müller-Hannemann, Ralf Rückert, and Sebastian S. Schmidt. „Vehicle Capacity-Aware Rerouting of Passengers in Delay Management“. In: *19th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2019)*. Ed. by Valentina Cacchiani and Alberto Marchetti-Spaccamela. Vol. 75. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 7:1–7:14. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/11419` (cit. on pp. 2, 96–98, 142).

[MHS07] Matthias Müller-Hannemann and Mathias Schnee. „Finding All Attractive Train Connections by Multi-criteria Pareto Search“. In: *Algorithmic Methods for Railway Optimization*. Ed. by Frank Geraets, Leo Kroon, Anita Schoebel, Dorothea Wagner, and Christos D. Zaroliagis. Vol. 4359. Lecture Notes in Computer Science (LNCS). 2007, pp. 246–263 (cit. on pp. 10, 38).

[MHS09] Matthias Müller-Hannemann and Mathias Schnee. „Efficient Timetable Information in the Presence of Delays“. In: ed. by Ravindra K. Ahuja, Rolf H. Möhring, and Christos D. Zaroliagis. Vol. 5868. Lecture Notes in Computer Science (LNCS). Springer, 2009, pp. 249–272 (cit. on p. 38).

[MHW01] Matthias Müller-Hannemann and Karsten Weihe. „Pareto shortest paths is often feasible in practice“. In: *Proceedings of the 5th International Workshop on Algorithm Engineering*. Vol. 2141. Lecture Notes in Computer Science (LNCS). Springer. 2001, pp. 185–197 (cit. on p. 10).

[MN18] Joshua Fields Millburn and Ryan Nicodemus. 2018. URL: `https://twitter.com/TheMinimalists/status/981670255423574016` (cit. on p. 7).

[MS98] Paola Modesti and Anna Sciomachen. „A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks1This work has been partially supported by the Italian National Research Council (CNR) Project on Transportation “PFT2”, subproject 4.2.1, Contract N. 96.00112.PF74.1“. In: *European Journal of Operational Research* 111.3 (1998), pp. 495 –508. URL: `http://www.sciencedirect.com/science/article/pii/S0377221797003767` (cit. on p. 95).

[Mül+21a] Matthias Müller-Hannemann, Ralf Rückert, Alexander Schiewe, and Anita Schöbel. *Framework for Generating Machine Learning Models for Robustness . Homepage.* `https://gitlab.rlp.net/for2083/framework-for-generating-machine-learning-models-for-robustness`. 2021 (cit. on pp. 124, 126, 127).

[Mül+21b] Matthias Müller-Hannemann, Ralf Rückert, Alexander Schiewe, and Anita Schöbel. „Towards Improved Robustness of Public Transport by a Machine-Learned Oracle“. In: *21st Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS 2021, September 9-10, 2021, Lisbon, Portugal (Virtual Conference)*. Ed. by Matthias Müller-Hannemann and Federico Perea. Vol. 96. OASIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 3:1–3:20 (cit. on pp. 2, 3, 121, 125, 133, 136, 137).

[Mur+10] Pavankumar Murali, Maged Dessouky, Fernando Ordóñez, and Kurt Palmer. „A delay estimation technique for single and double-track railroads“. In: *Transportation Research Part E: Logistics and Transportation Review* 46.4 (2010), pp. 483–495 (cit. on p. 35).

[Ngu+18] Hoang Nguyen, Le-Minh Kieu, Tao Wen, and Chen Cai. „Deep learning methods in transportation domain: a review“. In: *IET Intelligent Transport Systems* 12 (9 2018), 998–1004(6). URL: `https://digital-library.theiet.org/content/journals/10.1049/iet-its.2018.0064` (cit. on p. 122).

[Oli16] Travis Oliphant. *NumPy - The fundamental package for scientific computing with Python.* 2016. URL: `https://numpy.org/` (visited on Oct. 10, 2020) (cit. on p. 127).

[One+18] Luca Oneto, Emanuele Fumeo, Giorgio Clerico, et al. „Train Delay Prediction Systems: A Big Data Analytics Perspective“. In: *Big Data Research* 11 (2018). Selected papers from the 2nd INNS Conference on Big Data: Big Data & Neural Networks, pp. 54–64. URL: `https://www.sciencedirect.com/science/article/pii/S2214579617300060` (cit. on p. 122).

[Ope19] OpenStreetMap contributors. *Planet dump retrieved from https://planet.osm.org.* `https://www.openstreetmap.org`. 2019 (cit. on p. 24).

[OR90] Ariel Orda and Raphael Rom. „Shortest-path and Minimum-delay Algorithms in Networks with Time-dependent Edge-length“. In: *J. ACM* 37.3 (July 1990), pp. 607–625. URL: `http://doi.acm.org/10.1145/79147.214078` (cit. on p. 14).

[Pät+17] Julius Pätzold, Alexander Schiewe, Philine Schiewe, and Anita Schöbel. „Look-Ahead Approaches for Integrated Planning in Public Transportation“. In: *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017).* Ed. by Gianlorenzo D'Angelo and Twan Dollevoet. Vol. 59. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 17:1–17:16. URL: `http://drops.dagstuhl.de/opus/volltexte/2017/7894` (cit. on pp. 86, 126).

[Ped+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, et al. „Scikit-Learn: Machine Learning in Python“. In: *J. Mach. Learn. Res.* 12 (2011), 2825–2830 (cit. on p. 127).

[PH06] Yonghwa Park and Hun-Koo Ha. „Analysis of the impact of high-speed railroad service on air transport demand“. In: *Transportation Research Part E: Logistics and Transportation Review* 42.2 (2006). Selected papers from the 8th ATRS Conference, pp. 95 –104. URL: `http://www.sciencedirect.com/science/article/pii/S1366554505000748` (cit. on p. 65).

[PK01] Leon Peeters and Leo Kroon. „A cycle based optimization model for the cyclic railway timetabling problem“. In: *Computer-aided scheduling of public transport.* Vol. 505. Lecture Notes in Computer Science (LNCS). Springer, 2001, pp. 275–296 (cit. on p. 126).

[Pra87] Sir Terence David John Pratchett. *Equal Rites.* Discworld: 3rd novel – 1st Witches story. Victor Gollancz in association with Colin Smythe, 1987, p. 82 (cit. on p. 65).

[Pra96] Sir Terence David John Pratchett. *Hogfather.* 20th novel – 4th Death story. Victor Gollancz, 1996 (cit. on p. 121).

[PRMH22] Julian Patzner, Ralf Rückert, and Matthias Müller-Hannemann. „Passenger-Aware Real-Time Planning of Short Turns to Reduce Delays in Public Transport". In: *22nd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2022)*. Ed. by Mattia D'Emidio and Niels Lindner. Vol. 106. Open Access Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 13:1–13:18. URL: `https://drops.dagstuhl.de/opus/volltexte/2022/17117` (cit. on p. 141).

[Pro+19] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. *CatBoost: unbiased boosting with categorical features*. 2019. arXiv: `1706.09516 [cs.LG]` (cit. on p. 127).

[PSS18] Julius Pätzold, Alexander Schiewe, and Anita Schöbel. „Cost-Minimal Public Transport Planning". In: *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*. Ed. by Ralf Borndörfer and Sabine Storandt. Vol. 65. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 8:1–8:22. URL: `http://drops.dagstuhl.de/opus/volltexte/2018/9713` (cit. on p. 86).

[Ptn] *Collection of open source public transport networks by DFG Research Unit "FOR 2083: Integrated Planning For Public Transportation"*. `https://github.com/FOR2083/PublicTransportNetworks`. 2018 (cit. on p. 125).

[Pyr+08] Evangelia Pyrga, Frank Schulz, Dorothea Wagner, and Christos Zaroliagis. „Efficient Models for Timetable Information in Public Transportation Systems". In: *ACM J. Exp. Algorithmics* 12 (June 2008). URL: `https://doi.org/10.1145/1227161.1227166` (cit. on pp. 10, 12, 14).

[Ras15] Sebastian Raschka. *Python Machine Learning*. Birmingham, UK: Packt Publishing, 2015 (cit. on p. 123).

[Rc19] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2019. URL: `https://www.R-project.org` (cit. on p. 97).

[RE09] Andrea Raith and Matthias Ehrgott. „A comparison of solution strategies for biobjective shortest path problems". In: *Computers and Operations Research* 36.4 (2009), pp. 1299 –1331. URL: `http://www.sciencedirect.com/science/article/pii/S0305054808000233` (cit. on p. 10).

[RFID18] *Contactless Smart Card Schemes in the Asia Pacific Region*. Visited on: 2018-10-16. URL: `{https://www.securetechalliance.org/secure/reports/Asia_CSC_Report.pdf}` (cit. on p. 20).

[Ros+11] Bernd Rosenbusch, Kord Simons, Thomas Haberer, and Karl-Heinz Wächter. „Integrierte Fahrgastinformation für den öffentlichen Verkehr: Vernetzung von Defas Bayern und RIS der Deutschen Bahn / Integrated passenger information for public transport". In: *NAHVERKEHR* 29.46 (2011), pp. 35–39 (cit. on p. 35).

[Rot+99] Emilie M. Roth, Nicolas Malsch, Jordan Multer, and Michael Coplen. „Understanding how Train Dispatchers Manage and Control Trains: A Cognitive Task Analysis of a Distributed Team Planning Task". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 43.3 (1999), pp. 218–222. URL: `https://doi.org/10.1177/154193129904300319` (cit. on p. 35).

[Rüc+15] Ralf Rückert, Martin Lemnian, Steffen Rechner, Christoph Blendinger, and Matthias Müller-Hannemann. „PANDA: A software tool for improved train dispatching with focus on passenger flows“. In: *CASPT 2015 (Conference on Advanced Systems in Public Transport)*. Rotterdam, 2015. URL: `http://www.rotterdam2015.caspt.org/proceedings/paper90.pdf` (cit. on pp. 2, 3, 34, 38, 43–45, 60, 61).

[Rö19] Bernd Röder. *Jeder vierte Fernzug war 2018 zu spät*. 2019. URL: `http://www.spiegel.de/reise/aktuell/unpuenktliche-deutsche-bahn-2018-war-jeder-vierte-fernzug-zu-spaet-a-1247350.html` (visited on Jan. 10, 2019) (cit. on p. 1).

[Rü+17] Ralf Rückert, Martin Lemnian, Christoph Blendinger, Steffen Rechner, and Matthias Müller-Hannemann. „PANDA: a software tool for improved train dispatching with focus on passenger flows“. In: *Public Transport* 9 (July 2017) (cit. on pp. 2, 3).

[Sch01] Anita Schöbel. „A Model for the Delay Management Problem based on Mixed-Integer-Programming“. In: *Electronic Notes in Theoretical Computer Science* 50.1 (2001). ATMOS 2001, Algorithmic MeThods and Models for Optimization of RailwayS (Satellite Workshop of ICALP 2001), pp. 1 –10. URL: `http://www.sciencedirect.com/science/article/pii/S1571066104001604` (cit. on pp. 37, 74, 87).

[Sch+05] Anita Schöbel, Jürgen Jacobs, Nicolai Bissantz, et al. „DisKon - Laborversion eines flexiblen, modularen und automatischen Dispositionsassistenzsystems“. In: *Eisenbahntechnische Rundschau* 12 (2005), 809–821 (cit. on p. 37).

[Sch06] Anita Schöbel. *Optimization in public transportation. Stop location, delay management and tariff planning from a customer-oriented point of view*. Optimization and Its Applications. New York: Springer, 2006 (cit. on p. 87).

[Sch07] Anita Schöbel. „Integer Programming Approaches for Solving the Delay Management Problem“. In: *Algorithmic Methods for Railway Optimization*. Ed. by Frank Geraets, Leo Kroon, Anita Schoebel, Dorothea Wagner, and Christos Zaroliagis. Vol. 4359. Lecture Notes in Computer Science (LNCS). Springer, 2007, pp. 145–170 (cit. on p. 37).

[Sch12] Anita Schöbel. „Line planning in public transportation: models and methods“. In: *OR Spectrum* 34.3 (June 2012), pp. 491–510 (cit. on pp. 88, 125).

[Sch13] Marie Schmidt. „Simultaneous optimization of delay management decisions and passenger routes“. In: *Public Transport* 5 (Sept. 2013), pp. 125–147 (cit. on p. 37).

[Sch15] Jürgen Schmidhuber. „Deep Learning in Neural Networks: An Overview“. In: *Neural Networks* 61 (2015), pp. 85–117 (cit. on p. 125).

[Sch16] Gero Scholz. *IT systems in public transport*. dpunkt.verlag, 2016 (cit. on p. 20).

[Sch17] Anita Schöbel. „An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation“. In: *Transportation Research Part C: Emerging Technologies* 74 (2017), pp. 348 –365 (cit. on p. 88).

[Sel+16] Peter Sels, Thijs Dewilde, Dirk Cattrysse, and Pieter Vansteenwegen. „Reducing the passenger travel time in practice by the automated construction of a robust railway timetable“. In: *Transportation Research Part B: Methodological* 84 (2016), pp. 124 –156 (cit. on p. 87).

[Ser87] Paolo Serafini. „Some Considerations about Computational Complexity for Multi Objective Combinatorial Problems“. In: *Recent Advances and Historical Development of Vector Optimization.* Ed. by Johannes Jahn and Werner Krabs. 1987, pp. 222–232 (cit. on p. 9).

[SG13] Michael Siebert and Marc Goerigk. „An experimental comparison of periodic timetabling models“. In: *Computers and Operations Research* 40.10 (2013), pp. 2251 –2259. URL: `http://www.sciencedirect.com/science/article/pii/S0305054813001020` (cit. on p. 27).

[SS10] Michael Schachtebeck and Anita Schöbel. „To Wait or Not to Wait-And Who Goes First? Delay Management with Priority Decisions“. In: *Transportation Science* 44 (Aug. 2010), pp. 307–321 (cit. on p. 37).

[SS13] Anita Schöbel and Silvia Schwarze. „Finding Delay-Resistant Line Concepts using a Game-Theoretic Approach“. In: *Netnomics* 14.3 (2013), pp. 95–117 (cit. on p. 136).

[sta17] statista. *Auslastung der Züge der Deutsche Bahn AG im Fernverkehr in den Jahren 2006 bis 2016.* 2017. URL: `https://de.statista.com/statistik/daten/studie/162886/umfrage/auslastung-der-zuege-der-deutschen-bahn-im-fernverkehr-seit-2006/` (visited on Jan. 3, 2017) (cit. on p. 96).

[Sta17] Statista. *Umfrage zu den Voraussetzungen für den Umstieg vom privaten Pkw auf den öffentlichen Personennahverkehr in Deutschland im Jahr 2017.* 2017. URL: `https://de.statista.com/statistik/daten/studie/892631/umfrage/voraussetzungen-fuer-den-umstieg-vom-pkw-auf-den-oepnv-in-deutschland/` (visited on Nov. 11, 2018) (cit. on p. 85).

[Ste18] *StepStone - MOBILITÄTSREPORT.* Visited on: 2018-10-16. URL: `{https://www.stepstone.de/Ueber-StepStone/wp-content/uploads/2018/04/StepStone_MobilitÃďtsreport_2018-1.pdf}` (cit. on p. 1).

[Sto97] Bram Stoker. *Dracula.* Archibald Constable and Company (UK), 1897 (cit. on p. 85).

[tag19] tagesschau. *So viele Flüge in Deutschland wie nie.* `https://www.tagesschau.de/inland/luftverkehr-107.html`. 2019 (cit. on p. 65).

[TC92] Chi Tung Tung and Kim Lin Chew. „A multicriteria Pareto-optimal path algorithm“. In: *European Journal of Operational Research* 62.2 (1992), pp. 203 –209. URL: `http://www.sciencedirect.com/science/article/pii/0377221792902488` (cit. on p. 10).

[Tea15] Google Brain Team. *Tensorflow.* 2015. URL: `https://www.tensorflow.org/` (visited on Oct. 10, 2020) (cit. on p. 127).

[van+15] Niels van Oort, Marc Drost, Ties Brands, and Menno Yap. „Data-driven public transport ridership prediction approach including comfort aspects.“ English. In: *Proceedings of Conference on Advanced Systems in Public Transport, 19-23 July 2015, Rotterdam.* CASPT, July 2015, pp. 1–13 (cit. on p. 20).

[VCU20] Varun Varghese, Makoto Chikaraishi, and Junji Urata. „Deep Learning in Transport Studies: A Meta-analysis on the Prediction Accuracy“. In: *Journal of Big Data Analytics in Transportation* 2 (2020), pp. 199–220 (cit. on p. 122).

[Ver] DB Vertrieb. *Umfassendere Informationen für Disponenten: Für die beste Entscheidung.* URL: `{https://digitalspirit.dbsystel.de/fuer-die-beste-entscheidung/}` (visited on July 11, 2019) (cit. on p. 47).

[Ver22] DB Vertrieb. *Pünktlichkeitsentwicklung 2022*. 2022. URL: `https://www.deutschebahn.com/de/konzern/konzernprofil/zahlen_fakten/puenktlichkeitswerte-6878476` (visited on Aug. 19, 2022) (cit. on pp. 1, 87).

[Wan+19] Yuan Wang, Dongxiang Zhang, Ying Liu, Bo Dai, and Loo Hay Lee. „Enhancing transportation systems via deep learning: A survey“. In: *Transportation Research Part C: Emerging Technologies* 99 (2019), pp. 144–163 (cit. on p. 122).

[WD44] T. Williams and G. Debusscher. *Tennessee Williams: The Glass Menagerie*. 1944 (cit. on p. 49).

[Yap+18] Menno D. Yap, Niels van Oort, Rob van Nes, and Bart van Arem. „Identification and quantification of link vulnerability in multi-level public transport networks: a passenger perspective“. In: *Transportation* 45.4 (2018), pp. 1161–1180 (cit. on p. 122).

[YC20] Menno Yap and Oded Cats. „Predicting disruptions and their passenger delay impacts for public transport stops“. In: *Transportation* (2020), pp. 1–29 (cit. on p. 122).

[YKS13] Masoud Yaghini, Mohammad M. Khoshraftar, and Masoud Seyedabadi. „Railway passenger train delay prediction via neural network model“. In: *Journal of Advanced Transportation* 47.3 (2013), pp. 355–368 (cit. on p. 91).

[Zei13] Süddeutsche Zeitung. *Deutsche Bahn startet Zugradar*. 2013. URL: `{https://www.sueddeutsche.de/wirtschaft/verspaetungen-in-echtzeit-deutsche-bahn-startet-zugradar-1.1754581}` (visited on Sept. 1, 2021) (cit. on p. 20).

Eidesstattliche Erklärung / Declaration under Oath

Ich erkläre an Eides statt, dass ich die Arbeit selbstständig und ohne fremde Hilfe verfasst, keine anderen als die von mir angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

I declare under penalty of perjury that this thesis is my own work entirely and has been written without any help from other people. I used only the sources mentioned and included all the citations correctly both in word or content.

Datum / Date	Unterschrift des Antragstellers / Signature of the applicant

Ralf Rückert

Lebenslauf

Persönliche Informationen

*27.01.1987 (Merseburg)
deutsch

Ausbildung

2010–2014	**Masters of Science - Bioinformatik**, *Martin-Luther-Universität* , Halle-Wittenberg, Note - *1.5*.
2007–2010	**Bachelor of Science - Bioinformatik**, *Martin-Luther-Universität* , Halle-Wittenberg, Note - *2.7*.

Masters Thesis

Titel	*Großstörungen im Bahnverkehr: Erkennung, Ausbreitung und Vorhersage*
Supervisor	Prof. Dr. Matthias Müller-Hannemann

Berufserfahrung

2014-2018, 2019-Jetzt	**Wissenschaftlicher Mitarbeiter**, MARTIN-LUTHER-UNIVERSITÄT, Halle-Wittenberg, Institut für Informatik. Arbeit an wissenschaftlichen Veröffentlichungen im Bereich öffentlicher Verkehr. Implementierung von Modellen in C++ und Auswertung und Aufbereitung von Ergebnissen in R. Aufbereitung von Planung und Echtzeit-Daten für die Simulationsmodelle aus Datenbanken und Live-Feeds.
2018–2019	**Wissenschaftlicher Mitarbeiter**, MARTIN-LUTHER-UNIVERSITÄT, Halle-Wittenberg, Medizinische Fakultät - Future Care Lab. Arbeit an Ausschreibung für "WIR! – Wandel durch Innovation in der Regionmit dem Projekt TDG - “Translationsregion für digitalisierte Gesundheitsversorgung”.
2019	Arbeit im Future Care Lab an Projekten mit modernen Technologien. Programmierung in Unity für Applikation auf der Microsoft HoloLens. Programmierung in Unity für 3D Lernanwendung für Medizinstudenten.

Awards

2018	tdg.innovate.healthcare Hackathon Preis: Größter Patientennutzen
2018	tdg.innovate.healthcare Hackathon Preis: Beste Geschäftsidee

Kenntnisse

Basiswissen	Microsoft Windows, SQL, Office

Gute Kenntnisse	Maschinelles Lernen, Statistik, Datenbanken, JAVA,C#,PYTHON,LaTeX, Linux, git
Experten-kenntnisse	Algorithmen und Datenstrukturen, Objektorientierte Programmierung, C++,R

Sprachen

Deutsch	**Muttersprache**	
Englisch	**Fachkundige Sprachkenntnisse**	*Halten von Vorträgen auf internationalen wissenschaftlichen Konferenzen*

Publikationen

[1] Ralf Rückert. Großstörungen im Bahnverkehr: Erkennung, Ausbreitung und Vorhersage. Masterthesis, Martin Luther University of Halle-Wittenberg, 2013.

[2] Martin Lemnian, Ralf Rückert, Steffen Rechner, Christoph Blendinger, and Matthias Müller-Hannemann. Timing of Train Disposition: Towards Early Passenger Rerouting in Case of Delays. In Stefan Funke and Matúš Mihalák, editors, *14th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 42 of *OpenAccess Series in Informatics (OASIcs)*, pages 122–137, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[3] Ralf Rückert, Martin Lemnian, Steffen Rechner, Christoph Blendinger, and Matthias Müller-Hannemann. PANDA: A software tool for improved train dispatching with focus on passenger flows. In *CASPT 2015 (Conference on Advanced Systems in Public Transport)*, Rotterdam, 2015.

[4] Martin Lemnian, Matthias Müller-Hannemann, and Ralf Rückert. Sensitivity Analysis and Coupled Decisions in Passenger Flow-Based Train Dispatching. In Marc Goerigk and Renato Werneck, editors, *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*, volume 54 of *OpenAccess Series in Informatics (OASIcs)*, pages 2:1–2:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[5] Ralf Rückert, Martin Lemnian, Christoph Blendinger, Steffen Rechner, and Matthias Müller-Hannemann. Panda: a software tool for improved train dispatching with focus on passenger flows. *Public Transport*, 9, 07 2017.

[6] Matthias Müller-Hannemann and Ralf Rückert. Dynamic event-activity networks in public transportation. *Datenbank-Spektrum*, 17(2):131–137, 2017.

[7] Markus Friedrich, Matthias Müller-Hannemann, Ralf Rückert, Alexander Schiewe, and Anita Schöbel. Robustness Tests for Public Transport Planning. In Gianlorenzo D'Angelo and Twan Dollevoet, editors, *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*, volume 59 of *OpenAccess Series in Informatics (OASIcs)*, pages 6:1–6:16, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[8] Markus Friedrich, Matthias Müller-Hannemann, Ralf Rückert, Alexander Schiewe, and Anita Schöbel. Robustness as a Third Dimension for Evaluating Public Transport Plans. In Ralf Borndörfer and Sabine Storandt, editors, *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*, volume 65 of *OpenAccess Series in Informatics (OASIcs)*, pages 4:1–4:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[9] Matthias Müller-Hannemann, Ralf Rückert, and Sebastian S. Schmidt. Vehicle Capacity-Aware Rerouting of Passengers in Delay Management. In Valentina Cacchiani and Alberto Marchetti-Spaccamela, editors, *19th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2019)*, volume 75 of *OpenAccess Series in Informatics (OASIcs)*, pages 7:1–7:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[10] Matthias Müller-Hannemann, Ralf Rückert, Alexander Schiewe, and Anita Schöbel. Towards improved robustness of public transport by a machine-learned oracle. In Matthias Müller-Hannemann and Federico Perea, editors, *21st Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS 2021, September 9-10, 2021, Lisbon, Portugal (Virtual Conference)*, volume 96 of *OASIcs*, pages 3:1–3:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[11] Matthias Müller-Hannemann, Ralf Rückert, Alexander Schiewe, and Anita Schöbel. Estimating the robustness of public transport schedules using machine learning. *Transportation Research Part C: Emerging Technologies*, 137:103566, 2022.

[12] Julian Patzner, Ralf Rückert, and Matthias Müller-Hannemann. Passenger-Aware Real-Time Planning of Short Turns to Reduce Delays in Public Transport. In Mattia D'Emidio and Niels Lindner, editors, *22nd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2022)*, volume 106 of *Open Access Series in Informatics (OASIcs)*, pages 13:1–13:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Datum / Date

Unterschrift des Antragstellers / Signature of the applicant