

Computing Highly Reliable Train Journeys

Vom Fachbereich Informatik der
Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Dr. rer. nat.

genehmigte Dissertation
von **Mohammad Hossein Keyhani**, M.Sc.
geboren am 21.05.1986 in Teheran, Iran

Referent: Prof. Dr. rer. nat. Karsten Weihe
Korreferent: Prof. Dr.-Ing. Andreas Oetting

Tag der Einreichung: 2. Februar 2017
Tag der Disputation: 25. April 2017

Darmstadt, 2017
Hochschulkenziffer: D-17

Abstract

Millions of people travel daily by public transport in order to reach their destinations. Public transport is often an attractive alternative to traveling by other means such as cars. In daily operation, however, unfortunate delays frequently arise, disrupting the scheduled departure and arrival times of the trains. Even slight delays can result in connection breaks wherein passengers miss their connecting trains because they arrive too late for planned transfers. A considerable number of passengers are consequently faced by delays and their repercussions every day.

This work focuses on the computation of reliable journeys. First, we demonstrate an accurate method of assessing the reliability of train connections. For the assessment of a train connection, we compute the probability of passengers reaching their destinations when taking the train connection; in other words, the probability of the train connection not breaking because of delays. Our method considers the timetable, interdependencies between trains, current delays in the railway network, and stochastic prognoses for the travel times of the trains. Regarding the latter, we use probability distributions which are computed based on tangible historical delay data. The interdependencies between the trains are caused by delay managements; trains wait for the passengers of other delayed trains. Our computational study—which is based on real data—reveals that our reliability assessments are realistic and accurate.

We then address a fundamental problem in planning journeys: arrive in time by train at the destination with a high probability. In addition, to save time, passengers usually desire to commence a journey as late as possible. We present an efficient solution to the described problem. We compute highly reliable train journeys by which the destination can be reached with a high probability of being on time even in case of delays. Such a train journey includes a train connection along with alternative continuations to the destination. The latter are used in case of connection breaks caused by delays. Our optimal approach computes the best choice in enabling the continuance of the journey for each situation that may occur when traveling. Along all possible continuations, the best choice is the continuation with the highest probability of being on time at the destination. The evaluations presented illustrate that the train

journeys computed are highly reliable and attractive to passengers even in terms of both travel time and convenience.

State-of-the-art routing systems provide the search for intermodal, door-to-door connections. Beside public transport, passengers use modes of transport such as taxis, car sharing, bike sharing, and individual cars. We extend our method of assessing the reliability of train connections to intermodal connections. Moreover, we discuss approaches to the computation of reliable intermodal connections.

Last, we address the problem of connection breaks late at night; such a situation is frustrating for passengers, particularly if the destination cannot be reached by public transport prior to the end of operations. In such situations, railway companies must offer compensation to ensure the rights of passengers are upheld. We propose a solution to the problem of finding connections to the destination by taking into account the options of taxi rides or overnight stays in hotels. The main objectives are the satisfaction of passengers and cost reductions for the railway company.

The methods and algorithms presented in this work are designed for real, large train networks such as in Germany with more than a million departure and arrival events per day. We use a fully realistic model that represents the timetable and relevant factors which influence train delays. Thus, we compute realistic train journeys which can be used by passengers in order to reach their destinations. Our computational studies are based on real data from the German railway company, Deutsche Bahn AG. The evaluations demonstrate that our approaches deliver promising results, are practicable, and can be integrated into timetable information systems in order to answer large amounts of passenger queries per day.

Zusammenfassung

Täglich nutzen Millionen Menschen öffentliche Verkehrsmittel, um ihre Reiseziele zu erreichen. Häufig sind öffentliche Verkehrsmittel attraktive Alternativen zu anderen Verkehrsmitteln wie Autos. Allerdings finden jeden Tag Ereignisse statt, die Verspätungen verursachen und die fahrplanmäßigen Abfahrts- und Ankunftszeiten der Züge beeinträchtigen. Bereits kleine Verspätungen können zu Verbindungsbrüchen führen, so dass Passagiere ihre Anschlüsse verpassen. Somit sind viele Menschen täglich mit Verspätungen und deren Folgen konfrontiert.

Diese Arbeit befasst sich mit dem Themengebiet Berechnung von zuverlässigen Bahnverbindungen. Zuerst zeigen wir eine genaue Methode zur Bewertung der Zuverlässigkeit von Bahnverbindungen. Für die Bewertung einer Bahnverbindung berechnen wir die Wahrscheinlichkeit, dass Passagiere mit der Verbindung ihr Ziel erreichen können und diese nicht aufgrund von Verspätungen bricht. Unsere Methode berücksichtigt den Fahrplan, Abhängigkeiten zwischen den Zügen, die aktuellen Verspätungen im Bahnnetz und stochastische Prognosen für Reisezeiten der Züge. Die Abhängigkeiten zwischen den Zügen entstehen durch Wartezeitregelungen: Züge warten auf Passagiere aus anderen verspäteten Zügen. Für die stochastischen Prognosen verwenden wir Wahrscheinlichkeitsverteilungen, die auf realen Verspätungsdaten aus der Vergangenheit basieren. Die Evaluation der Zuverlässigkeitsbewertungen anhand realer Daten zeigt, dass diese realistisch und akkurat sind.

Daraufhin befassen wir uns mit einem grundsätzlichen Problem bei Bahnreisen: der rechtzeitigen Ankunft am Ziel mit sehr hoher Wahrscheinlichkeit. Als sekundäres Kriterium möchten Passagiere ihre Reise so spät wie möglich antreten, um die Reisezeit zu minimieren. Wir präsentieren einen effizienten Algorithmus zur Lösung des beschriebenen Problems. Unser Ansatz berechnet hochzuverlässige Bahnreisen mit denen das Ziel mit sehr hoher Wahrscheinlichkeit auch im Falle von Verspätungen rechtzeitig erreicht werden kann. Eine solche Bahnreise besteht aus einer Bahnverbindung samt Alternativen zum Ziel, auf die Passagiere bei Verbindungsbrüchen im Verspätungsfall ausweichen können. Für jede Situation, die während der Reise auftreten könnte, berechnet unser Ansatz die beste Option, um die Reise fortzusetzen. Die beste Option

maximiert die Wahrscheinlichkeit einer rechtzeitigen Ankunft am Ziel. Die Evaluation des Ansatzes weist nach, dass die berechneten Bahnreisen sowohl hochzuverlässig als auch komfortabel sind und attraktive Reisezeiten haben.

Moderne Auskunftssysteme ermöglichen Reisenden intermodale „Tür zu Tür“ Verbindungen zu berechnen. Neben öffentlichen Verkehrsmitteln nutzen Passagiere bei solchen Verbindungen auch Verkehrsmittel wie Taxi, Carsharing, Bike-Sharing und privates Auto. Für dieses Szenario zeigen wir eine entsprechende Erweiterung unseres Ansatzes, die die Bewertung der Zuverlässigkeit solcher Verbindungen ermöglicht. Zudem beschreiben wir Ansätze zur Berechnung von zuverlässigen intermodalen Verbindungen.

Als letzten Punkt adressieren wir das Problem der Verbindungsbrüche an Tagesrandlagen. Eine solche Situation ist für Passagiere frustrierend, insbesondere wenn das Reiseziel mit öffentlichen Verkehrsmitteln vor Betriebsschluss nicht mehr erreicht werden kann. Dadurch entstehen auch für die Verkehrsunternehmen Kosten zur Sicherung der Fahrgastrechte. Wir schlagen eine Lösung vor, bei der intermodale Verbindungen zum Ziel genutzt werden, die unter anderem Taxifahrten oder Hotelaufenthalte enthalten. Die Optimierungskriterien unseres Ansatzes sind die Zufriedenheit der Passagiere und die Minimierung der für die Verkehrsunternehmen entstehenden Kosten.

Die präsentierten Methoden und Algorithmen sind für reale und große Bahnnetze entworfen. Ein Beispiel hierfür ist das Bahnnetz in Deutschland mit mehr als einer Million Abfahrts- und Ankunftsereignissen am Tag. Wir nutzen ein vollkommen realistisches Modell, das den Fahrplan und relevante Faktoren, die Einflüsse auf Verspätungen haben, repräsentiert. Somit berechnen wir realistische Verbindungen, die von Passagieren genutzt werden können, um ihre Reiseziele zu erreichen. Unsere Evaluationen basieren auf realen Daten der Deutschen Bahn AG. Diese zeigen, dass unsere Ansätze vielversprechende Ergebnisse liefern und praktikabel sind. Sie können in Fahrplanauskunftssysteme integriert werden, um täglich große Mengen an Passagieranfragen zu verarbeiten.

Acknowledgments

First of all, I would like to express my sincere gratitude and appreciation to my supervisor Karsten Weihe who supported me through my research and Ph.D. studies. He always had an open door for me and helped me to overcome difficulties and challenges. I especially thank Mathias Schnee for his indispensable guidance and many constructive discussions regarding my research.

I sincerely thank Sebastian Fahnenschreiber, Felix Gündling, and Hans-Peter Zorn for their cooperation, code implementations, and constructive discussions. I appreciate the work of all students who wrote a thesis and had a contribution to my work: Peter Glöckner [Glö15], Julian Gross [Gro15], Pablo Hoch, Maximilian Müller [Mül16], Marco Plaue [Pla16], Tobias Raffel [Raf15], and Kai Schwierczek [Sch16]. I thank all the members of the Algorithm Engineering group at TU Darmstadt for their help and support.

I would like to thank our cooperation partner Deutsche Bahn AG, particularly Dr. Christoph Blendinger, Steffen Brouër, Dr. Christof Jung, and Nils Passau, for the support of my research, inspiring discussions, and the provision of data: timetables, real-time messages, probability distributions, etc. I also thank DADINA and HEAG mobilo GmbH for the provision of historical delay data.

Last but not least, I would like to thank my loving wife Arezu, my mother, my father, and my sister for their support and love throughout my life.

Contents

Introduction	1
Our Contribution	2
The Structure of This Work	4
1 Model and Setting	5
1.1 Timetable	5
1.2 Train Connections	7
1.3 Delays	9
1.3.1 Real-time Incidents	9
1.3.2 The Model	10
1.4 Intermodal Connections	11
2 Finding Attractive Connections: Basics	15
2.1 Graph Models	15
2.1.1 The Time-Expanded Graph Model	15
2.1.2 The Time-Dependent Graph Model	18
2.2 Pareto Optimal Train Connections	21
2.2.1 Pareto Dominance and Pareto Optimality	22
2.2.2 Pareto Dijkstra	22
2.3 Pareto Optimal Intermodal Connections	24
2.3.1 The First Step: Computing Individual Routes	26
2.3.2 The Second Step: Extending the Graph	27
2.3.3 The Third Step: Pareto Dijkstra	27
2.4 Conclusion	28
3 Reliability Assessment	31
3.1 Introduction	31
3.2 Related Work	32
3.3 Stochastic Model	34
3.3.1 Informal Description	34

3.3.2	Formal Description	35
3.4	Distributions for Prepared Times and Train Move Durations	36
3.4.1	Prepared Times: Learn from Historical Data	37
3.4.2	Train Move Durations: Learn from Historical Data	39
3.4.3	Generate Distributions	41
3.5	Distributions for Random Event Times	42
3.5.1	The Interdependencies Between the Events	42
3.5.1.1	The Dependencies of Departure Events	42
3.5.1.2	The Dependencies of Arrival Events	43
3.5.2	The Calculation of the Probabilities	44
3.5.2.1	The Formula for Departure Events	44
3.5.2.2	The Formula for Arrival Events	45
3.6	The Reliability Assessment of Transfers	46
3.7	The Reliability Assessment of Train Connections	49
3.7.1	The Calculation	49
3.7.2	Deadline Scenario	51
3.8	Processing Events	52
3.8.1	On-Demand Processing	53
3.8.2	Pre-Processing by Dynamic Programming	53
3.8.2.1	The Dynamic Programming Approach	53
3.8.2.2	The Queue	55
3.8.3	Hybrid Approach	55
3.8.4	Parallelization	56
3.9	Incorporation of Real-time Incidents	56
3.10	Asymptotic Complexity	58
3.11	Analysis of the Assumptions of Independence	59
3.12	Cancelations and Reroutings	60
3.13	Implementation	61
3.13.1	Extension of the Graph Model	62
3.13.2	Dependency Graph	63
3.13.3	Negligible Probabilities	64
3.13.4	The Queue	64
3.14	Conclusion	65
4	On-Time Arrival Guarantee	67
4.1	Introduction	67
4.2	State of the Art	69
4.3	Problem Definition	71

4.3.1	Input and Queries	71
4.3.2	Outcomes: Complete Connections	71
4.3.3	Exact Problem Definition	73
4.3.3.1	Latest Departure Subject to Arrival Guarantee	73
4.3.3.2	Maximum Probability of Success	73
4.3.3.3	On-Trip Scenario at a Station	74
4.3.3.4	On-Trip Scenario in a Train	74
4.3.3.5	Minimum Expected Travel Time	74
4.4	Two-Stage Approach	75
5	Optimal Complete Connections	77
5.1	Exact Definition of Complete Connections	78
5.1.1	Initial Departure Event	78
5.1.2	The Instructions to Continue Traveling	79
5.2	The Search Approach	80
5.2.1	The Solution Space	80
5.2.2	Computing the Optimal Complete Connection	81
5.3	Computation of Instructions and Probabilities	83
5.3.1	Set of Successor Events	84
5.3.2	The Recursive Function	84
5.3.3	Selection of the Best Event for Continuation	85
5.3.4	Tie-Breaker: Optimizing the Number of Transfers	85
5.4	Computation Extended by Travel Time Optimization	87
5.4.1	Extension of the Recursive Function	87
5.4.2	Travel Time Optimization	88
5.5	Properties of Complete Connections	90
5.6	Walk Trips in Complete Connections	91
5.7	Translation into Dynamic Programming	93
5.7.1	The Approach	93
5.7.2	Early Termination	95
5.8	Relevant Events	96
5.8.1	Defintion of Relevant Events	97
5.8.2	Discover in Pre-Processing	98
5.9	Discovering and Processing Incrementally	100
5.9.1	Abstract View	100
5.9.2	Detailed View	102
5.10	Asymptotic Complexity	104
5.11	Heuristics	106

5.11.1	Maximum Waiting Time of Passengers	107
5.11.2	Earliest Arrival Time at the Destination	107
5.11.3	Earliest Departure Time	108
5.11.4	Cycles Through the Departure Station	108
5.12	Parallelization	108
5.13	Implementation	108
5.13.1	Accessing the Timetable	109
5.13.2	Relevant Events	109
5.13.3	The Search Labels	110
5.14	Conclusion	111
6	Travel Guidance	113
6.1	Interactive Visualization	114
6.1.1	Web Interface: a Detailed Visualization	114
6.1.2	Mobile Application: a Compact Visualization	116
6.2	Preserving the Optimality of Complete Connections	117
6.2.1	One-Time Interaction: Offline Database	117
6.2.2	Ongoing Interaction: Online Database	118
6.2.2.1	Checking Complete Connections	118
6.2.2.2	Updating Complete Connections	120
6.3	Conclusion	120
7	Alternative Search Approaches	123
7.1	On-Time Arrival Guarantee: Alternative Approaches	123
7.1.1	Disregarding the Probability of Success (<i>DP-L</i> and <i>DP-LP</i>)	124
7.1.2	Minimum Buffer Times for Transfers and Arrival (<i>BT</i>)	124
7.1.3	Connections Extended by Alternatives (<i>CEA</i>)	125
7.2	Reliability as Pareto Criterion	127
7.2.1	Reliability Assessment	128
7.2.2	The Search Algorithm	128
7.3	Conclusion	130
8	Reliable Intermodal Connections	131
8.1	Related Work	132
8.2	Reliability Assessment	132
8.2.1	Predictions Based on Historical Data	133
8.2.2	Reliability Assessment Based on the Predictions	134
8.3	The Search for Reliable Intermodal Connections	136
8.3.1	Retrieving Individual Routes and Their Assessments	137

8.3.2	Excluding Unreliable Rides	139
8.3.3	Reliability as Pareto Criterion	141
8.3.4	Intermodal Connections Extended by Alternatives	143
8.4	Late-Night Connections	144
8.4.1	The Search Approach	145
8.4.2	Retrieving Taxi Transfers	146
8.4.3	The Costs Calculation	147
8.4.4	Modeling Taxi Transfers and Hotel Stays	147
8.4.5	The Pareto Search for Late-Night Connections	149
8.5	Conclusion	151
9	Computational Study	153
9.1	Setup Data	153
9.2	Distributions for Random Event Times	155
9.2.1	Computing the Probability Density Functions	155
9.2.2	Updating the Probability Density Functions	156
9.3	The Quality of the Reliability Assessments	157
9.4	On-Time Arrival Guarantee	161
9.4.1	Objectives of the Computational Study	162
9.4.2	The Evaluated Approaches	162
9.4.3	Classifying the Outcomes	164
9.4.4	Evaluation Setup and Test Queries	165
9.4.5	The Outcomes	165
9.4.6	Relevance of Probability of Success and Optimality	167
9.4.7	Minimum Buffer Times	167
9.4.8	The Price of On-Time Arrival Guarantee	169
9.4.9	Travel Convenience	170
9.4.10	Efficiency and Practicability	172
9.4.11	Heuristics	173
9.4.12	The Discovery of Relevant Events	174
9.5	Late-Night Connections	175
9.5.1	Evaluation Setup	175
9.5.2	Results	179
10	Conclusion and Outlook	183
10.1	Conclusion	183
10.2	Outlook	186

CONTENTS

A Appendices	189
A.1 Reliability of Transfers	189
A.2 Travel Guidance Component	192
A.3 Generated Probability Distributions	192
Nomenclature	197
List of Algorithms	205
List of Figures	207
List of Tables	209
Publications	211
Bibliography	213

Introduction

Public transport is one of the most important means of transport for both short and long distances. In 2015, more than 6 million passengers per day traveled on Deutsche Bahn AG’s trains [Bah16a]. All trains operate according to a timetable unless delays arise and influence their departure and arrival times. In railway networks, delays occur to a significant extent. In 2015, a quarter of all long-distance trains and about 6% of all regional trains of Deutsche Bahn AG had delays of more than five minutes [Bah16a]. Even slight delays can result in connection breaks; passengers miss their connecting trains because they arrive too late for the planned transfers¹. Consequently, a considerable number of passengers are faced by delays every day.

Many countries in the world have extensive railway networks with wide coverages of travel destinations. In such networks, planning train journeys is not a trivial task. For this purpose, passengers usually use timetable information systems. Such systems find attractive train connections among a huge number of possibilities; a train connection contains instructions for the passengers concerning how to travel from a departure station to a destination station in the train network. By way of example, millions of train connections per day are computed by the timetable information system developed by the company HaCon [HaC16]. State-of-the-art systems even take into account the current delays in the railway network when answering passenger queries [Sch09]. However, each train connection delivered is feasible according to the network state that prevails at the moment when the train connection is computed. Afterwards, delays—which are unknown at the moment of the computation—could change the network state, thus breaking the planned transfers.

Hence, *reliability* when faced by disruptions in the timetable caused by delays is an important quality criterion of train connections. An unreliable train connection is not a suitable choice for passengers even if it is attractive in terms of travel time, convenience, and price. Thus, an accurate method of assessing the reliability of train connections as well as promising approaches to computing reliable journeys for passengers are necessary.

¹Leaving a train at a station and entering another one; also called train changes.

The computation of reliable train connections has previously been addressed in different works² which are, however, either inapplicable to large railway networks such as those in Germany or result in unattractive journeys with long travel times. Among these previous approaches, Berger et al. have presented a method for stochastic delay predictions in large train networks [BGMHO11]. In their model, for each departure and arrival event of each train in the railway network, there is a random variable for the unknown real time of the event. For each of these random variables, a probability distribution is computed by taking into account the interdependencies between the trains in the timetable as well as probability distributions for the random durations of the train rides. The resulting probability distributions for the departure and arrival events are both realistic and promising predictions for train delays, on which our approach is based.

Our contribution First, in Chapter 3, we present a method of assessing the reliability of train connections in relation to delays. It is a stochastic method of computing the *probability of the success* of a train connection based on stochastic predictions of train delays. This is the probability that, when taking the train connection, a passenger can reach his/her destination even in the case of delays. In other words, it is the probability that the train connection does not break because of delays. We compute the probability of success quite accurately by taking into account the timetable, interdependencies between the trains in the network, the current delays in the network as known so far, and stochastic prognoses for the travel times of the trains. For the latter, we use probability distributions which are computed based on tangible historical delay data. We present a comprehensive computational study which reveals that our reliability assessments for train connections are relatively realistic and accurate. These evaluations are based on real timetables and delay data provided to us by Deutsche Bahn AG. Our method is efficient and the run time per train connection is on the order of microseconds.

We subsequently introduce highly reliable *complete connections* in Chapter 4. A complete connection comprises a set of instructions; passengers are guided by these instructions in order to reliably travel from a departure station to a destination station. In contrast to conventional train connections—which are chains of trains to be taken successively—a complete connection is a train connection along with alternative continuations to the destination station. For each transfer that could potentially break, a complete connection has fallback solutions.

In complete connections, we address a significant problem in planning journeys: arrive in time by train at the destination with a high probability. Passengers usually desire to commence the journey as late as possible but early enough that arrival no later

²We will discuss related work in Sect. 3.2, 4.2, and 8.1.

than a deadline can be guaranteed with a high probability. Although many thousands of passengers globally are confronted with this problem daily, we are, to the best of our knowledge, the first to address it and provide a promising solution.

In Chapter 5, we present an efficient approach that computes optimal complete connections in relation to the problem introduced. In short, our approach is to compute highly reliable complete connections which contain the best instruction concerning how to continue the journey for each situation that may occur when traveling. Among all available options to continue traveling, the best instruction maximizes the probability of reaching the destination no later than the deadline. Such an instruction could be staying on a train or leaving it in order to transfer to another train. Subject to the guarantee of arriving in time, the complete connections are constructed such that passengers commence their journeys as late as possible in order to save time. Travel time and the number of transfers can be optimized as secondary criteria. Although planning optimal complete connections is very compute-intensive, our algorithm minimizes the computational effort and still delivers optimal results. In the large train network of Germany with more than a million departure and arrival events per day, we require about two seconds on average in order to compute an optimal complete connection. Our extensive computational study reveals that our approach delivers highly reliable complete connections which are still attractive in terms of attributes such as travel time and convenience.

In Chapter 6, we discuss a travel guidance component to accompany passengers while traveling. On the one hand, it provides a user-friendly and comprehensive visualization of complete connections. On the other hand, the travel guidance component is responsible for preserving the optimality of complete connections even after changes to the train network due to delays and further occurrences. Whenever the situation in the train network has changed, a computed complete connection can be updated if desired by the passenger. As a result, promising new options are taken into account: alternative continuations that were unplanned but are possible due to delays which have changed the network state.

In Chapter 8, we extend our reliability assessment method to intermodal, door-to-door connections. Such connections consist of public transport but also individual modes of transport such as taxis, car sharing, bike sharing, individual cars, etc. More precisely, we address a problem of passengers who intend to use car sharing or bike sharing; usually, there is no guarantee that a car or a bike will be available when it is required at some point in the journey. Therefore, we use predictions for the availability of such services based on real, historical data. Furthermore, we discuss methods of computing reliable intermodal connections: we either optimize reliability as a search criterion or omit unreliable rides by individual modes of transport. We also

discuss a method of computing reliable intermodal connections along with alternative continuations to the destination.

Finally, in Sect. 8.4, we address the problem of connection breaks late at night. Such a situation is frustrating for passengers, particularly if the destination cannot be reached prior to the end of the respective day's train operations. Beside passenger dissatisfaction, the railway company must pay compensations to ensure the rights of passengers are upheld. We propose a solution to the issue of finding intermodal connections to the destination by taking into account the options of taxi rides or overnight stays in hotels. The main objectives are the satisfaction of passengers and cost reduction for the railway company. Our approach might support railway companies in providing a better service to their passengers.

The structure of this work In Chapter 1, we explain how a timetable, delays, train connections, and intermodal connections are modeled; this work is based on the definitions introduced in that chapter. In Chapter 2, we discuss the basics of finding attractive connections.

In Chapter 3, we first discuss state-of-the-art methods of assessing the reliability of train connections (cf. Sect. 3.2). We then discuss a stochastic model for delay predictions in a train network. After that, we address the assessment of the reliability of train connections and present our method.

The problem of arriving by train at the destination with a high probability of being on time, as well as the idea of computing train connections along with alternative continuations, is introduced in Chapter 4. There, we also discuss related work (cf. Sect. 4.2). Our optimal approach to solving this problem is detailed in Chapter 5. The travel guidance component is addressed in Chapter 6. After that, in Chapter 7, we discuss alternative approaches to computing reliable train connections and complete connections; in our computational study, we compare our optimal approach to these approaches.

Chapter 8 addresses the assessment of the reliability of intermodal connections, and the computation of reliable intermodal connections. There, we also present our solution to overcome the problem of connection breaks late at night. State-of-the-art intermodal approaches are discussed in Sect. 8.1.

Our computational studies and the evaluations of our implementations are presented in Chapter 9. Finally, in Chapter 10, we conclude this work and discuss future work.

Chapter 1

Model and Setting

In this chapter, we explain how a timetable is modeled along with a wide range of its elements which are relevant to timetable information systems. The model presented here has been essentially established in many works which address timetabling in train networks as presented by [Sch09, MHS09, MHS04, SWW00, DMS08, Gün15]. The work by [Sch09] is the most relevant reference for this chapter; for intermodal connections, we refer to [Gün15]. Parts of the content of this section are taken from our paper [KSW17].

1.1 Timetable

The terminology is illustrated in Fig. 1.1 (on page 6) by way of example. A timetable TT comprises the following elements.

Trains A *train* is a means of transport that operates according to a timetable; this includes long-distance and short-distance trains as well as short-range transit such as streetcars and buses. A train starts its ride at a station, has successive stops at a sequence of further stations, and ends its ride at its terminal station. Each train has a *train category* such as *ICE* (German intercity express) or *bus*. In our model, the set TR comprises all trains in the timetable.

Stations A *station* is a place where a train can load and unload passengers. The set S comprises all stations in the timetable.

Times and validity period Each timetable has a *validity period*; the set of all points in time in the timetable's validity period is denoted by $TIMES = 0, 1, 2, \dots, n$ where 0 and $n \in \mathbb{N}$ each represent the earliest and the latest points in time in the validity period respectively. For all time values, one minute is the indivisible basic unit.

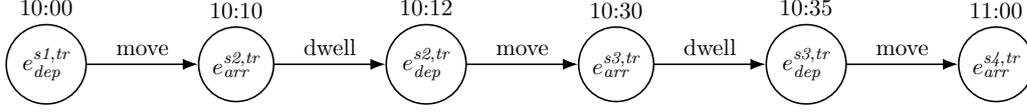


Figure 1.1: The figure illustrates a train trip by the train tr starting at the station $s1$ and ending at the station $s4$. It consists of three train moves with scheduled durations of 10, 18, and 25 minutes respectively. The train trip is via the stations $s2$ and $s3$; the dwell times at these stations are 2 and 5 minutes respectively. Above each event, the scheduled event time is illustrated.

Departure and arrival events After loading and unloading passengers at a station s , a train tr departs from s in order to move to another station $s2$; this event is denoted as the *departure event* $e_{dep}^{s,tr}$. The train tr subsequently arrives at the station $s2$; this subsequent event is denoted as the *arrival event* $e_{arr}^{s2,tr}$. Hence, each departure and arrival event is associated with a train and a station. We denote by E_{dep} and E_{arr} the sets of all departure events and all arrival events in the timetable respectively. The set $E = E_{dep} \cup E_{arr}$ comprises all departure and arrival events in the timetable. For each station $s \in S$, the sets E_{dep}^s and E_{arr}^s comprise all departure and arrival events at s respectively.

Note: For the sake of convenience, in the model presented here, we assume that the tuple of a station, a train, and an event type (departure or arrival) is a unique key specifying an event in the set E of all events. In real timetables, a train could visit a station several times. To model such timetables, we extend the keys used for the events by the scheduled times (see below) of the events.

Scheduled event times Each event $e \in E$ has a *scheduled event time*, denoted by $sched(e) \in TIMES$, which specifies when the event has to take place according to the timetable.

Train moves A *train move* $tm = (e_{dep}^{s,tr}, e_{arr}^{s2,tr})$ is a ride of the train tr starting with the departure at the station s and ending with the very next arrival of tr at the station $s2$. Set TM comprises all train moves in the timetable.

Scheduled duration of train moves Each train move $(e_{dep}^{s,tr}, e_{arr}^{s2,tr}) \in TM$ has a *scheduled duration* that equals

$$min_dur(e_{dep}^{s,tr}, e_{arr}^{s2,tr}) = sched(e_{arr}^{s2,tr}) - sched(e_{dep}^{s,tr}).$$

For each train move $(e_{dep}^{s,tr}, e_{arr}^{s2,tr}) \in TM$, one has $sched(e_{dep}^{s,tr}) \leq sched(e_{arr}^{s2,tr})$.

Dwell activities and times After the arrival at a station s , a train tr waits at s at least for a specific *minimum dwell time* before it departs again. This activity is

denoted as the *dwell activity* $dw = (e_{arr}^{s,tr}, e_{dep}^{s,tr})$. The minimum dwell time of dw is denoted by $min_dur(dw)$, and is given in the timetable. The set DW comprises all dwell activities in the timetable.

Train trips Fig. 1.1 (on page 6) illustrates a train trip by way of example. A *train trip* $tm_1, dw_1, tm_2, dw_2, \dots, tm_{n-1}, dw_{n-1}, tm_n$ is an alternating sequence of $n \in \mathbb{N}_{>0}$ train moves and $n - 1$ dwell activities. The former are successive train moves of the same train. More precisely, a train trip has the following structure:

$$\begin{array}{c} \text{train move} \quad \text{dwell activity} \quad \text{train move} \\ \underbrace{(e_{dep}^{s_1,tr}, e_{arr}^{s_2,tr})}_{\text{train move}}, \underbrace{(e_{arr}^{s_2,tr}, e_{dep}^{s_2,tr})}_{\text{dwell activity}}, \underbrace{(e_{dep}^{s_2,tr}, e_{arr}^{s_3,tr})}_{\text{train move}}, \dots, \\ \underbrace{(e_{dep}^{s_{n-1},tr}, e_{arr}^{s_n,tr})}_{\text{train move}}, \underbrace{(e_{arr}^{s_n,tr}, e_{dep}^{s_n,tr})}_{\text{dwell activity}}, \underbrace{(e_{dep}^{s_n,tr}, e_{arr}^{s_{n+1},tr})}_{\text{train move}}. \end{array}$$

The scheduled times of the events of a train trip are monotonously increasing. The longest trip possible with a train is from the station where the train starts its ride to the station where it terminates.

Walk trips Two stations $s \in S$ and $s' \in S$ may be within short walking distance from each other. For such a pair of stations, there is a *walk trip*, denoted by $walk = (s, s')$, in the set $WALKS$ of all walk trips. Analogously, if the tracks for long-distance trains, regional trains, subways, etc. are in different buildings or on different floors, these buildings or floors are modeled as separate stations with *walk trips*. The duration of each walk trip $walk \in WALKS$ is denoted by $min_dur(walk)$, and is given in the timetable.

1.2 Train Connections

Train connections are used by passengers to move from a station to another one. Fig. 1.2 (on page 8) illustrates a train connection by way of example. A precise definition is presented in the following.

Train connections A *train connection* represents a journey that starts at some station and terminates at another one; it consists of train trips and walk trips. For $n \in \mathbb{N}_{>0}$ and $i = 1, 2, \dots, n$, a *train connection* is a sequence $trip_1, trip_2, \dots, trip_i, \dots, trip_n$ where $trip_i$ is either a train trip or a walk trip. For $i = 1, 2, \dots, n - 1$, the very last station of $trip_i$ and the very first station of $trip_{i+1}$ are the same. We assume that there are never two or more successive walk trips in a train connection.

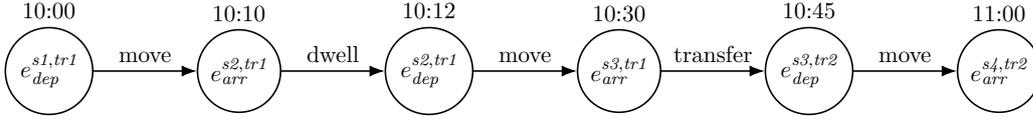


Figure 1.2: By way of example, the figure illustrates a train connection starting at the station $s1$ and ending at the station $s4$. The train connection starts with the train $tr1$; there is a transfer from $tr1$ to $tr2$ at the station $s3$; the destination is reached via the train $tr2$. Above each event, the scheduled event time is illustrated.

Transfer activities and minimum transfer times In a train connection c , for $i = 1, 2, \dots, n - 1$, let $trip_i$ and $trip_{i+1}$ be two train trips of the trains $tr1$ and $tr2$ respectively. At the very last station s of $trip_i$, the passenger—who takes the train connection c —leaves $tr1$ and enters $tr2$; this activity is called a *transfer*. Two events are directly involved in each transfer: the arrival event $e_{arr}^{s, tr1}$ and the departure event $e_{dep}^{s, tr2}$. For these two events, there is a *transfer activity* denoted by $trans = (e_{arr}^{s, tr1}, e_{dep}^{s, tr2})$ in the set of all transfer activities $TRANS$ in the timetable. Hence, the set $TRANS$ comprises each transfer activity that is possible in the timetable.

For each transfer activity $(e_{arr}^{s, tr1}, e_{dep}^{s, tr2}) \in TRANS$, we require

$$sched(e_{arr}^{s, tr1}) + min_dur(e_{arr}^{s, tr1}, e_{dep}^{s, tr2}) \leq sched(e_{dep}^{s, tr2}) \quad (1.2.1)$$

where $min_dur(e_{arr}^{s, tr1}, e_{dep}^{s, tr2})$ is the *minimum time* required for the transfer from $tr1$ into $tr2$ at s . In timetables of Deutsche Bahn AG [Bah16c], the minimum time of a transfer activity generally equals a station-specific minimum transfer time. However, if both trains—which are involved in the transfer activity—arrive and depart at the same platform, a platform-specific time is given that is shorter than the station-specific time.

The set $TRANS$ additionally contains transfers with walk trips, as will be introduced below.

Transfers with walk trips In a train connection with $n \in \mathbb{N}_{>0}$ trips (cf. Sect. 1.2, *Train connections*), let $trip_i$ as well as $trip_{i+2}$ be two train trips and $trip_{i+1}$ a walk trip where $i \in [1, n - 2]$. At the very last station $s1$ of $trip_i$, the passenger leaves the train $tr1$ of $trip_i$, walks to the very first station $s2$ of $trip_{i+2}$, and enters the train $tr2$. The activity $(e_{arr}^{s1, tr1}, e_{dep}^{s2, tr2})$ is denoted as a *transfer with a walk trip*. For such a transfer $(e_{arr}^{s1, tr1}, e_{dep}^{s2, tr2})$, we require that

- $(s1, s2) \in WALKS$ and
- $sched(e_{arr}^{s1, tr1}) + min_dur(s1, s2) \leq sched(e_{dep}^{s2, tr2})$.

The set $TRANS$ contains all *transfers with walk trips*, in addition to all transfers without walk trips as introduced above. For each transfer $trans \in TRANS$ with a walk trip $walk \in WALKS$, we define $min_dur(trans) = min_dur(walk)$.

Note that a train connection can start or terminate with a walk trip; such a walk trip is not a transfer activity.

Each train connection has the following relevant properties.

Scheduled departure time It is the scheduled time of the very first departure event in the train connection. If the train connection starts with a walk trip, the scheduled departure time equals the scheduled time of the very first departure event in the train connection minus the duration of the walk trip.

Scheduled arrival time It is the scheduled time of the very last arrival event in the train connection. If the train connection ends with a walk trip, the scheduled arrival time equals the scheduled time of the very last arrival event in the train connection plus the duration of the walk trip.

Scheduled travel time It is the difference between the scheduled arrival time and the scheduled departure time.

Number of transfers It is the number of the transfer activities in the train connection.

1.3 Delays

In our real-world setting, the departure and arrival times of the trains do not always comply with the timetable. Delays arise because of various reasons, and result in event times later than scheduled. In addition, trains could even arrive earlier than scheduled. In Sect. 1.3.1, we discuss delays and other incidences disrupting the timetable. In Sect. 1.3.2, we discuss how the real-time disruptions are modeled.

1.3.1 Real-time Incidents

The following incidents can disrupt the timetable and change the network state:

Delays A train could be delayed such that its departure and arrival events take place at times which defer from the timetable. The information concerning a new delay is published by the railway company as soon as the respective event has actually taken place. More precisely, railway companies publish the information at which time an event has taken place. We consequently receive this information for an event whether or not it is delayed.

Cancelations A train or a part of it could be canceled; respectively, all or a subset of its train moves become invalid.

Additional trains New trains could be added to the timetable, for example, in order to manage increased passenger traffic.

Reroutings A train could be rerouted; some of its train moves become invalid, and some new train moves are added to the timetable.

We assume that each of the above incidents is known as soon as it occurs. The German railways provides to us this data in a real-time data stream.

1.3.2 The Model

Real departure and arrival times The *real time* of an event is the time when it actually takes place; for each event $e \in E$, this time is denoted by $real(e) \in TIMES$. The time $real(e)$ is consequently unknown unless e actually takes place.

Real durations of train moves The *real duration* of a train move $(e_{dep}, e_{arr}) \in TM$ is defined by $real(e_{arr}) - real(e_{dep})$.

Maximum waiting times For each transfer activity $trans = (e_{arr}^{s,tr}, e_{dep}^{s,tr2}) \in TRANS$, a specific *maximum waiting time* is denoted by $max_wait(trans)$. In order to allow passengers a transfer from tr to $tr2$ at station s , the train $tr2$ waits for tr until

$$\min \{ real(e_{arr}^{s,tr}) + min_dur(trans), sched(e_{dep}^{s,tr2}) + max_wait(trans) \}$$

if $e_{arr}^{s,tr}$ is delayed such that

$$real(e_{arr}^{s,tr}) + min_dur(trans) \in [sched(e_{dep}^{s,tr2}) + 1, sched(e_{dep}^{s,tr2}) + max_wait(trans)].$$

For transfers with walks, the maximum waiting time is zero; no train waits for another train that arrives at another station.

In each timetable, there are rules for the maximum waiting times; these are usually established considering train properties such as train categories.

Feasible transfers Each transfer activity $trans = (e_{arr}, e_{dep}) \in TRANS$ is *feasible according to the timetable* by definition. More precisely, one has

$$sched(e_{arr}) + min_dur(trans) \leq sched(e_{dep}).$$

On the other side, *trans* is *feasible according to the real times* if

$$\text{real}(e_{\text{arr}}) + \text{min_dur}(\text{trans}) \leq \text{real}(e_{\text{dep}}),$$

and *infeasible according to the real times* otherwise. In the latter case, the transfer is called *broken*.

Feasible train connections A train connection is *feasible according to the timetable* by definition since all its transfer activities are feasible according to the timetable. On the other side, a train connection is *feasible according to the real times* if all its transfers are feasible according to the real times; otherwise, it is *infeasible according to the real times* and called *broken*.

Cancelations If train moves of a train have been canceled, we remove them from the set *TM*; additionally, we remove from the timetable all activities associated with those train moves.

Additional trains To model an additional train, we add all of its train moves to the set *TM*. The respective dwell activities are added to the set *DW*. Analogously, the rules for transfer times and maximum waiting times of the additional train have to be added to the respective sets modeling the timetable.

Reroutings The canceled train moves are removed from the timetable as described above for cancelations. The new train moves are added to the timetable as described above for additional trains. The new parts are linked with the existing train via dwell activities such the result is still a valid train.

1.4 Intermodal Connections

Intermodal connections have been discussed by [Gün15, Gün14, Arn14]; the definitions in this section are taken from those works. The content presented in this section is the basis for the approaches presented in Chapter 8.

Passengers usually do not commence and end traveling at public transport stations (which are stations as defined in Sect. 1.1) but they intend moving from a *starting location* to a *destination location*. These locations are not necessarily public transport stations but homes and workplaces of the passengers by way of example. An *intermodal connection*, also called a *door-to-door* connection in the literature, connects a departure location to a destination location. In intermodal connections, passengers use individual modes of transport in addition to public transport.

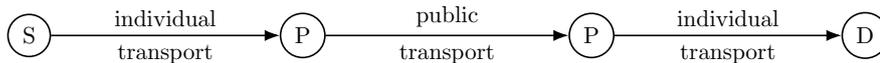


Figure 1.3: The figure illustrates the structure of an intermodal connection. While S and D respectively denote the start location and the destination location of the passenger, public transport stations are denoted by P. First, the passenger moves from his/her starting location to a train station by an individual mode of transport. Then, he/she travels by public transport to a train station in the near of the destination location. Finally, from the train station, the destination location is reached by an individual mode of transport.



Figure 1.4: The figure illustrates the structure of an intermodal connection with station-based bike sharing at the beginning and at the end. While S and D respectively denote the start location and the destination location of the passenger, public transport stations are denoted by P. Bike sharing stations where bikes are collected and returned are denoted by B. Each edge either represents a walking (w) or a ride by bike (b). For station-based car sharing, the structure of an intermodal connection is analogous.

Individual modes of transport Walks, taking a taxi, driving by individual cars and bikes, car sharing, bike sharing, etc. are *individual modes of transport*. In our model, they can be used in order to travel from the starting location to a public transport station and from a public transport station to the destination location. For car sharing and bike sharing, we use a *station-based model* as illustrated in Fig. 1.4. The passenger collects a car or a bike, respectively, at a *car sharing station* or at a *bike sharing station* in the near of the starting location; he/she then returns it at a respective station in the near of a public transport station. While walking as well as individual bikes cause no costs for passengers, modes such as taxi, individual cars, car sharing, and bike sharing have prices.

The structure of an intermodal connection Each intermodal connection consists of three parts as illustrated in Fig. 1.3 (on page 12) and Fig. 2.3 (on page 27). In the *initial part*, the passenger moves by an individual mode of transport from his/her starting location to a public transport station $s \in S$. Then, he/she travels from s to a public transport station $s' \in S$ in the near of the destination location; this middle part is a train connection as defined in Sect. 1.2. In the *end part* of the intermodal connection, the passenger finally moves by an individual mode of transport from s' to his/her destination location.

Note: For the case that the starting location and the destination location are actually public transport stations, the *initial part* and the *end part* would consist of short walks respectively.

For the initial and the end part, an intermodal connection comprises the information which individual mode of transport are used. Moreover, it stores the durations of the rides and the incurred costs. Finally, for station-based bike sharing and car sharing, the intermodal connection contains the information which car and bike stations are involved and when the car and the bike are collected and returned respectively.

Chapter 2

Finding Attractive Connections: Basics

In this chapter, we discuss the *Pareto Dijkstra* algorithm that is applied in state-of-the-art timetable information systems in order to find attractive train connections and intermodal connections. In Chapters 7 and 8, we discuss search approaches which are based on the approach discussed here. The reliability of train connections found by this approach can be assessed by our method that is introduced in Chapter 3.

Note: The contents of this chapter are not contributions of the author of this work but are taken from various works; we list the references in the respective sections.

2.1 Graph Models

In this section, we discuss how the timetable from Sect. 1.1 can be modeled as a directed acyclic graph. An efficient timetable modeling is a prerequisite for efficient algorithms computing train connections. In Sect. 2.1.1 and 2.1.2, we explain the *time-expanded* and the *time-dependent* graph model; they have already been discussed in various works [Sch09, MHS04, DMS08, Gün15].

2.1.1 The Time-Expanded Graph Model

The timetable from Sect. 1.1 can be modeled as a *time-expanded graph* as discussed by [MHS04, Sch09] and illustrated in Fig. 2.1. The timetable is represented by a directed acyclic graph $G = (V, A)$. The set V comprises *arrival*, *departure*, and *change nodes*. The set A comprises *train*, *dwelling*, *walk*, *leaving*, *entering*, *waiting*, and *special-transfer edges*. In the following, we introduce each node and edge type in detail.

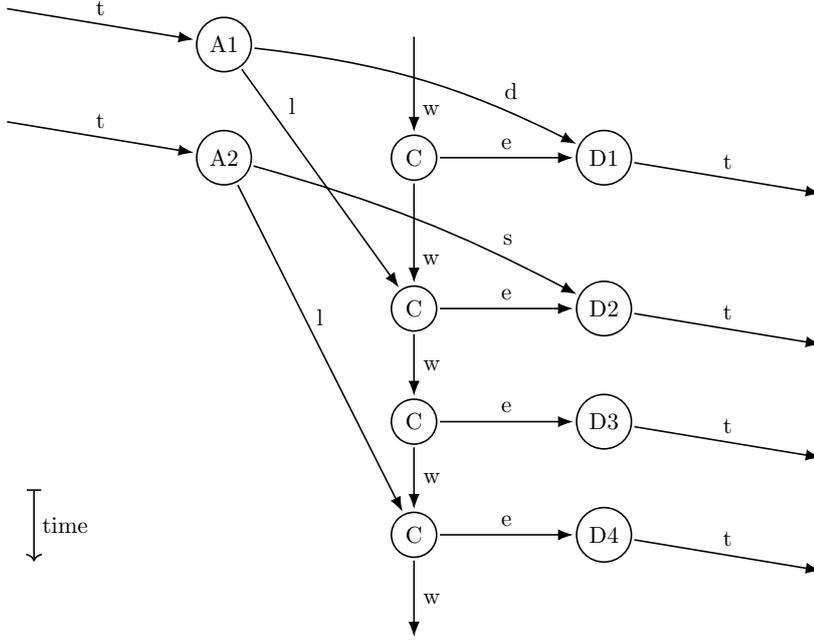


Figure 2.1: The figure illustrates, schematically, the typical situation at a station in a time-expanded graph consisting of arrival (A), departure (D), and change (C) nodes as well as train (t), dwell (d), leaving (l), entering (e), waiting (w), and special-transfer (s) edges. After the arrival at the station via the arrival node A1, a passenger can either stay in the train and continue the journey via D1 or change to one of the trains of D2, D3, D4, or a later departure node. From A2, a passenger can transfer into the train of D2 via the special transfer edge; a transfer into the train of D4 or a later departure node is possible via the leaving edge.

The train layer For each arrival and departure event $e \in E$, there is a *departure* and *arrival node* $v \in V$ respectively. Each departure or arrival node v has a scheduled time that is the scheduled time of the modeled event; we denote this time by $sched(v)$. We define the sets of all departure and arrival nodes as V_{dep} and V_{arr} respectively. The sets V_{dep}^s and V_{arr}^s comprise all departure and arrival nodes at the station s respectively.

Each train move $(e_{dep}, e_{arr}) \in TM$ is modeled by a *train edge* $(v_{dep}, v_{arr}) \in A$ where $v_{dep} \in V_{dep}$ and $v_{arr} \in V_{arr}$ represent e_{dep} and e_{arr} respectively. Analogously, each dwell activity $(e_{arr}, e_{dep}) \in DW$ is modeled by a *dwell edge* $(v_{arr}, v_{dep}) \in A$.

The transfer layer The transfer activities are modeled as follows. Every *change node* has a timestamp $t \in TIMES$. At each station and for each $t \in TIMES$, there is at most one change node with the timestamp t (see also the note below). More precisely, at a station s , a change node with the timestamp t exists if there is at least one departure event $e_{dep} \in E_{dep}^s$ where $t = sched(e_{dep})$.

Waiting edges at each station model waitings of passengers at the station for a departing train in order to continue the journey after the arrival at the station. At

each station, from each change node with the timestamp t_1 , there is a waiting edge to the change node with the timestamp t_2 where

- $t_1 \leq t_2$ and
- there is no other change node with the timestamp t_3 such that $t_1 \leq t_3 \leq t_2$.

There is no waiting edge to the change nodes with the smallest timestamp at each station; analogously, there is no waiting edge from the change node with the largest timestamp.

At each station, the departure and arrival nodes are connected to the change nodes as follows. From each change node at s with the timestamp t , there is an *entering edge* to each departure node $v_{dep} \in V_{dep}^s$ where $t = sched(v_{dep})$. Moreover, from each arrival node $v_{arr} \in V_{arr}^s$, there is a *leaving edge* to the change node at s with the smallest timestamp t such that $t \geq sched(e_{arr}^{s,tr1}) + \delta$ where

- $e_{arr}^{s,tr1} \in E_{arr}^s$ is the event represented by v_{arr} and
- the value δ equals

$$\delta = \max \left\{ \min_dur(e_{arr}^{s,tr1}, e_{dep}^{s,tr2}) \mid (e_{arr}^{s,tr1}, e_{dep}^{s,tr2}) \in TRANS \right\}. \quad (2.1.1)$$

For all transfer activities $(e_{arr}^{s,tr1}, e_{dep}^{s,tr2}) \in TRANS$ where $\min_dur(e_{arr}^{s,tr1}, e_{dep}^{s,tr2}) < \delta$, there is a *special-transfer edge* from v_{arr} to v_{dep} where v_{arr} and v_{dep} are the nodes representing $e_{arr}^{s,tr1}$ and $e_{dep}^{s,tr2}$ respectively. Such special-transfer edges model platform-specific transfer times (cf. Sect. 1.2, *Transfer activities and minimum transfer times*).

Hence, for all transfer activities $(e_{arr}^{s,tr1}, e_{dep}^{s,tr2}) \in TRANS$, the departure node for $e_{dep}^{s,tr2}$ is reachable from the arrival node for $e_{arr}^{s,tr1}$ via the change layer. Each transfer (without a walk trip) consequently starts with a leaving edge from the arrival node of the arriving train; then, a set of successive waiting edges follow which can be empty; last, the transfer is completed by an entering edge to the departure node of the departing train. An exception are transfers which are possible via special-transfer edges.

Note: Alternatively, at each station s , we could create change nodes with timestamps in $[0, 1440[$. The change node with the timestamp $t \in [0, 1440[$ is then connected to each departure event $e_{dep} \in V_{dep}^s$ where $t = sched(e_{dep}) \bmod 1440$ (see [MHS04]). This would reduce the number of change nodes to 1,440 per station at most. However, it requires additional checks to detect transfers which are possible in the graph but infeasible according to Eq. 1.2.1. In our computational study in Chapter 9, we decided for the model with reduced number of change nodes. However, for a better understanding of the model, in Sect. 2.1.1, we have introduced the model with (at most) a change node for each time $t \in TIMES$.

Walks There is a *walk edge* for each $walk \in WALKS$. Walk edges are not added to the time-expanded graph as the other edge types; they are instead used on-the-fly during the computation of train connections. A transfer activity $(e_{arr}^{s1,tr1}, e_{dep}^{s2,tr2}) \in TRANS$ with a walk trip $(s, s') \in WALKS$ always starts with a walk edge from the arrival node $v_{arr}^{s1,tr1}$ —representing $e_{arr}^{s1,tr1}$ — to the change node at $s2$ with the smallest timestamp t where $t \geq sched(v_{arr}^{s1,tr1}) + min_dur(s, s2)$. Then, there is a set of successive waiting edges which can be empty. Last, the transfer is completed by an entering edge to the departure node $v_{dep}^{s2,tr2}$.

Edge costs Each edge has a *transfer cost* and a duration cost; the latter is called the *length* of the edge. Each leaving edge has a transfer cost equal to 1; all other edges have a transfer cost equal to 0.

The length of each train, dwell, walk, and special-transfer edge equals the duration of the respective activity. Each waiting edge has a length equal to the time difference between its end nodes. The length of each leaving edge is equal to δ as defined in Eq. 2.1.1 (on page 17) while the length of all entering edges equals zero.

Outgoing and incoming edges Each node knows its outgoing and incoming edges. A departure node has always an outgoing train edge; it has or has not an incoming dwell edge; it has an incoming entering edge; last, it has a set of incoming special-transfer edges that can be empty. An arrival node has always an incoming train edge; it has or has not an outgoing dwell edge; it has an outgoing leaving edge (unless the end of the leaving edge would be outside the validity period of the timetable); last, it has a set of outgoing special-transfer edges that can be empty.

2.1.2 The Time-Dependent Graph Model

As an alternative to the graph model from Sect. 2.1.1, the timetable from Sect. 1.1 can be modeled as a *time-dependent graph* as discussed by [Sch09, DMS08, Gün15]. The time-dependent graph model is based on *routes*.

Routes All trains that visit exactly the same sequence of stations build a *route*. Let the set R comprise all routes in the timetable. For each train $tr \in TR$, there is a route $r \in R$ that contains tr . The other way round, for each route $r \in R$ there is a subset of trains $TR(r) \subseteq TR$ such that all trains in $TR(r)$ visit the same (ordered) sequence of stations.

In each route, no train overtakes the other. More precisely, for each route $r \in R$ and at each station s visited by the trains in the set $TR(r)$, there is an ordering of

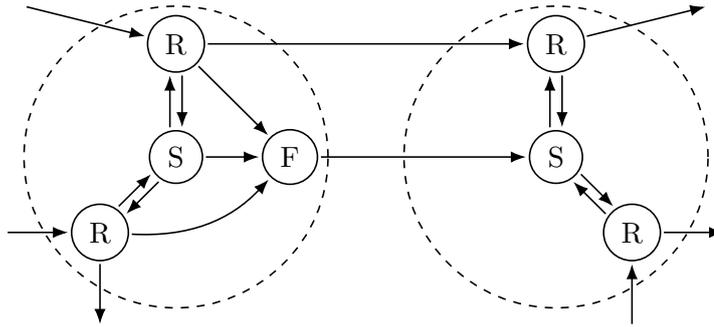


Figure 2.2: The figure illustrates, schematically, the typical situation at two directly connected stations in a time-dependent graph. All nodes inside a dashed circle belong to the same station. Station, route, and foot nodes are labeled with S, R, and F respectively.

the trains in $TR(r)$ according to the scheduled times of their departure events at s in ascending order. The same ordering exists at all stations visited by the trains in $TR(r)$.

Note: The time-dependent graph model is based on the following assumptions for the computation of attractive train connections. First, after the arrival at each station, it is sufficient to consider solely the very next possible departure on each route at that station. Second, transfers between the trains of the same route bring no advantage.

The graph model The timetable is represented by a directed graph $G = (V, A)$ as illustrated in Fig. 2.2. The set V comprises *station*, *route*, and *foot nodes*. The set A comprises *route* and *foot edges*. In the following, we introduce each node and edge type in detail.

Note: For the graph models both discussed in Sect. 2.1.1 and 2.1.2, we use the notation $G = (V, A)$. In the rest of this work, whenever a graph is required, we will explicitly mention the graph model so that the respective notation will be clear to the reader.

The train layer Let s_1, s_2, \dots, s_n denote the sequence of the stations visited by the trains in a route $r \in \mathcal{R}$ where $n \in \mathbb{N}_{>0}$ is the number of the visited stations. In the graph model, each route r is represented by the sequence of *route edges*

$$(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n)$$

where, for $i = 1, \dots, n$, the node v_i is a *route node* associated to the station s_i . The route node v_i of the route r represents the arrival event $e_{arr}^{s_i, tr} \in E_{arr}^{s_i}$ as well as the departure event $e_{dep}^{s_i, tr} \in E_{dep}^{s_i}$ (at the station s_i) of each train $tr \in TR(r)$. For $i = 1, \dots, n-1$, each route edge (v_i, v_{i+1}) represents the train move $(e_{dep}^{s_i, tr}, e_{arr}^{s_{i+1}, tr})$ of each train $tr \in TR(r)$.

The transfer layer Transfer activities without walk trips are modeled as follows. Each station $s \in S$ is represented by a *station node*. There is a *foot edge* from the station node to each route node associated to s . The other way round, from each route node associated to a station s , there is a foot edge to the station node at s . These foot edges facilitate entering and leaving trains.

To model transfer activities with walk trips, for each station $s \in S$, there is a *foot node* associated to s . There is a foot edge from each route node at s to the foot node at s . From the foot node at s there is a foot edge to the station node of each station $s' \in S$ where there is a walk activity $(s, s') \in WALKS$.

Note: If we would represent a walk activity by a direct edge from a station node to another one, for each transfer activity with a walk trip, we would obtain wrong a time cost equal to the station-specific transfer time plus the duration of the walk trip. This is the reason why a foot node is introduced.

Edge costs Each edge has a *transfer cost* and a *duration cost*; the latter is also called the *length* of the edge. Each edge from a route node to a station node or to a foot node has a transfer cost equal to 1. For the other edges, the transfer cost equals 0.

The length of each foot edge that represents a walk activity equals the duration of the walk activity. At each station s , the foot edge from each route node at s to the station node at s has a length equal to the station-specific minimum transfer time defined for s (cf. Sect. 1.2, *Transfer activities and minimum transfer times*). The length of the other foot edges equals 0.

*Note: The time-dependent model, as discussed here, does not support platform-specific transfer times (cf. Sect. 1.2, *Transfer activities and minimum transfer times*).*

The length of each route edge from a station $s \in S$ depends on the time when the edge is asked for its length; that is, the time when the passenger arrives at s (cf. Sect. 2.2). In words, the length models the waiting time for the next departure and the duration of the train move which is taken; more precisely, the length is defined as follows. Let $TM_{(v,w)} \subseteq TM$ comprise all train moves represented by the route edge $(v,w) \in A$. Given a time $t \in TIMES$, the length of a route edge $(v,w) \in A$ equals

$$length_{(v,w)}(t) = (sched(e_{dep}) - t) + min_dur(e_{dep}, e_{arr}) \quad (2.1.2)$$

where

$$(e_{dep}, e_{arr}) = \arg \min \{ sched(e_{dep}) \mid (e_{dep}, e_{arr}) \in TM_{(v,w)}(t) \}$$

and

$$TM_{(v,w)}(t) = \{ (e_{dep}, e_{arr}) \in TM_{(v,w)} \mid t \leq sched(e_{dep}) \}.$$

The length of the route edge (v, w) is set to $+\infty$ if the set $TM(v, w, t)$ is empty.

Note: Given a set of arguments and a function f , the operator “arg min” delivers the argument that minimizes the function output; ties are broken arbitrary.

Outgoing and incoming edges Each node knows its outgoing and incoming edges. Note that each route node has exactly one outgoing edge unless it is the head node of the very last route edge on a route; in the latter case, it has no outgoing edges. Analogously, each route nodes has exactly one incoming edge unless it is the tail node of the very first route edge on a route; in the latter case, it has no incoming edges.

2.2 Pareto Optimal Train Connections

In this section, we discuss the search for *Pareto optimal* train connections as presented by [MHS04, DMS08]. In Chapter 3, we will demonstrate how the reliability of train connections can be assessed; such train connections can be computed with the Pareto Dijkstra algorithm. Furthermore, in Chapters 7 and 8, we will present search approaches basing on the Pareto Dijkstra algorithm.

Given a passenger *query*, the motivation is to find train connections that are attractive to passengers according to a set of criteria such as travel time, number of transfers, etc. A *query* q comprises

- a *departure station* where the passenger starts traveling, denoted by $dep_st(q) \in S$,
- a *destination station* where the passenger ends traveling, denoted by $dest_st(q) \in S$, and
- either a time interval $[begin(q), end(q)]$ or a point in time $ontrip_time(q)$ each for the departure at $dep_st(q)$.

The values $ontrip_time(q)$, $begin(q)$, and $end(q)$ are times in the set $TIMES$. If a time interval $[begin(q), end(q)]$ is given, the passenger seeks attractive train connections from $dep_st(q)$ to $dest_st(q)$ with scheduled departure times (cf. Sect. 1.2) in $[begin(q), end(q)]$. On the other hand, if a time $ontrip_time(q)$ is given, the passenger has already commenced the journey and waits at $dep_st(q)$ at the time $ontrip_time(q)$. He/she consequently seeks attractive train connections from $dep_st(q)$ to $dest_st(q)$ where his/her waiting time at $dep_st(q)$ should be considered when evaluating the criterion *travel time* for each train connection; this waiting time is the time difference between $ontrip_time(q)$ and the scheduled departure time of the train connection.

To solve the above problem, in Sect. 2.2.1, we address the *Pareto* concept as a method of quality measurement for train connections. For each query, we then compute

all *Pareto optimal* train connections using the algorithm *Pareto Dijkstra* as discussed in Sect. 2.2.2.

2.2.1 Pareto Dominance and Pareto Optimality

The *Pareto* concept is applied to measure the quality of train connections according to a set of criteria such as travel time, number of transfers (convenience), price, etc. It is explained in Table 2.1 (on page 23) by way of example; a precise definition follows. A train connection c is called *Pareto optimal* if, in the timetable, there is no other train connection that *dominates* c . Let $n \in \mathbb{N}$ be the number of criteria relevant for the quality measurement; these criteria have to be minimized. Let two train connections c_1 and c_2 be represented as two n -dimensional tuples: $c_1 = (x_1, x_2, \dots, x_n)$ and $c_2 = (y_1, y_2, \dots, y_n)$ where, for $i = 1, \dots, n$, the values x_i and y_i are the values of c_1 and c_2 for the i -th criterion respectively. The train connection c_1 *dominates* c_2 if

$$\begin{aligned} \forall i \in \{1, \dots, n\} : x_i \leq y_i \quad \text{and} \\ \exists i \in \{1, \dots, n\} : x_i < y_i. \end{aligned} \tag{2.2.1}$$

Note: The content of Sect. 2.2.1 is taken from Sect. 2.2 and Sect. 3.3.2 of the works [Sch09] and [Gün15] respectively; we refer to those works for more details.

2.2.2 Pareto Dijkstra

Given a query as specified at the beginning of Sect. 2.2, the *Pareto Dijkstra* algorithm computes all Pareto optimal train connections. This algorithm is presented in Algorithm 2.1 (on page 29); it is taken from Algorithm 7 in [Sch09]. The Pareto Dijkstra algorithm can be applied on a time-expanded as well as a time-dependent graph. In Algorithm 2.1, we use data structures and functions which we define in the following.

Labels Labels are created at nodes in the set V . Each *label* represents a train connection from $dep_st(q)$ to the station at which the label is created and comprises the following data:

- a node to which the label is associated,
- a pointer to the predecessor label (also see *create_label* below),
- a n -dimensional tuple $(cost_1, cost_2, \dots, cost_n)$ of costs where, for $i = 1, \dots, n$, the value $cost_i$ is the cost of the label for the i -th criterion and $n \in \mathbb{N}$ is the number of the criteria relevant to the passenger, and

connection	duration	transfers	Pareto optimal
A	100	2	✓
B	101	1	✓
C	102	0	✓
D	100	3	–
E	103	0	–

Table 2.1: The table lists a set of train connections. The columns “duration” and “transfers” are the travel time and the number of transfers of each of the train connections respectively. The column “Pareto optimal” demonstrates whether a train connection is Pareto optimal according to the criteria travel time and number of transfers or not. While the train connections A, B, and C are Pareto optimal, the train connections D and E are dominated by A and C respectively.

- a timestamp representing the time at which the passenger is in the situation that is modeled by the label. This value is only required for the time-dependent graph model in order to retrieve the edge-cost (cf. time t in Eq. 2.1.2 on page 20).

Each label at a node at $dest_st(q)$ consequently represents a train connection from $dep_st(q)$ to $dest_st(q)$. The train connection can be reconstructed by following the predecessor label pointers recursively.

Creating start labels The function *create_start_labels* creates the initial labels for the search as follows. If a time interval $[begin(q), end(q)]$ is given in the query, we create a new label for each departure event $e_{dep} \in E_{dep}^{dep_st(q)}$ at $dep_st(q)$ where $sched(e_{dep}) \in [begin(q), end(q)]$. Each label is associated to the node $v \in V$ that represents the respective departure event e_{dep} . To support walk trips at the beginning of train connections, we analogously create labels for each departure event $e_{dep} \in E_{dep}$ where e_{dep} is at a station that is reachable from $dep_st(q)$ via a walk trip and $sched(e_{dep}) - \delta \in [begin(q), end(q)]$; the value δ is the duration of the respective walk trip.

On the other hand, in the on-trip scenario,

- in the time-expanded graph model, we create a new label at the earliest change node at $dep_st(q)$ which has a timestamp greater than or equal to $ontrip_time(q)$, and
- in the time-dependent graph model, we create a new label at the station node of $dep_st(q)$; the timestamp of the label is set to $ontrip_time(q)$.

For each new label, the pointer to the predecessor label is set to *void*, and each value in the tuple $(cost_1, cost_2, \dots, cost_n)$ of costs is set to zero. For the case of the on-trip scenario in a time-expanded graph, we set the cost for the criterion *travel time* to the time difference between the timestamp of the change node and $ontrip_time(q)$.

Creating labels during the search Given an existing label \hat{l} at a node $v \in V$ and an outgoing edge $(v, w) \in A$ of v , the function *create_label* creates a new label l at the node w . The label \hat{l} is the predecessor of l , and the timestamp of l is set to the timestamp of \hat{l} plus the duration cost (the length) of (v, w) . The tuple of costs of l is defined as follows. For $i = 1, \dots, n$, the cost $cost_i$ of l equals the cost $cost_i$ of \hat{l} plus the respective cost of the edge (v, w) as defined in Sect. 2.1.1 and 2.1.2 (cf. *transfer cost* and *duration cost*).

Label dominance At the end of the search, the function *filter_labels* retrieves all Pareto optimal labels associated to nodes at the destination station $dest_st(q)$. For this, the rule for Pareto dominance from Eq. 2.2.1 (on page 22) is applied. During the search, at Lines 11 and 17 in Algorithm 2.1 (on page 29), we also apply the dominance rule from Eq. 2.2.1 (on page 22) but with the following restrictions: label l is not allowed to dominate label l' if

- the timestamp of l' is prior to the timestamp of l or
- the scheduled departure time of the train connection represented by l is earlier than the scheduled departure time of the train connection represented by l' .

The above conditions are prerequisites that no Pareto optimal train connection is dominated during the search. For more details, we refer to Sect. 3.3.2 in [Gün15].

The priority queue The *priority queue* maintains labels. The operation *push* adds a new label to the queue. The operation *pop* delivers the label with the minimum costs as follows. The labels are sorted according to their tuples of costs in ascending order; in the comparison of two tuples of costs, the individual cost values in the tuples are compared in a lexicographical order.

This concludes the search for Pareto optimal train connections. In the following, there is a note on the *backward search*.

Backward search For the case that the passenger provides a time interval or a deadline both for the arrival at $dest_st(q)$, we also can apply Algorithm 2.1 (on page 29) but we need to traverse the graph backwards in time. The reader may assume that such queries can be processed analogously to the search queries where a time for the departure at $dep_st(q)$ is given. For more details, we refer to [Sch09].

2.3 Pareto Optimal Intermodal Connections

In this section, we discuss the search for Pareto optimal intermodal connections. The contents presented in this section are the basics for the approaches presented in Chap-

ter 8. Pareto optimal intermodal connections and the search for them in time-dependent graphs have been introduced by [Gün15, Gün14, Arn14, GKSW14]; an application and extension can be found in [FGKS16]. The content presented in this section is taken from those works.

From Sect. 1.4, recall that each intermodal connection consists of a train connection and two parts where individual modes of transport are used. In intermodal connections, the criteria which are relevant to passengers are usually travel time, price, and number of transfers (convenience). Each criteria must be assessed by taking into account the entire connection including the parts where individual modes of transport are used. The search approach, as explained here, computes Pareto optimal intermodal connections for each query such that the intermodal connections are optimized in their entirety.

Note: Alternatively, we could separately optimize the train connection and the parts with individual modes of transport, and, subsequently, merge them to an intermodal connection. However, such intermodal connections are not necessarily optimal—in their entirety—according to the criteria relevant to passengers.

Queries A query q comprises

- a *starting location* where the passenger starts traveling,
- a *destination location* where the passenger ends traveling,
- either a time interval $[begin(q), end(q)]$ or a point in time $ontrip_time(q)$ each for the start time of the journey at the starting location, and
- a list of individual modes of transport which can be used.

A location could be provided for example as GPS coordinates or addresses. The values $ontrip_time(q)$, $begin(q)$, and $end(q)$ are times in the set *TIMES*.

The search approach The search for Pareto optimal intermodal connections has three steps. First, we retrieve all *individual routes* connecting the departure location as well as the destination location with public transport. These are routes where passengers travel by individual modes of transport; they can be used as initial and end parts of intermodal connections. In the second step, we add to the time-dependent graph the individual routes that are computed for the end parts of intermodal connections. The result is a graph representing the timetable for public transport as well as the individual routes for the end parts of intermodal connections. Finally, in the third step, the set of all Pareto optimal connections is computed by applying the Pareto Dijkstra algorithm on the resulting graph. In order to model the individual routes for

the initial parts of intermodal connections, we modify the procedure of creating start labels (cf. Sect. 2.2.2, *Creating start labels*). In Sect. 2.3.1–2.3.3, we discuss each step more precisely.

2.3.1 The First Step: Computing Individual Routes

First, we introduce *individual routes*. Each *individual route* describes how to travel by a specific individual mode of transport either from the starting location to a public transport station or from a public transport station to the destination location; it has a duration (time cost) and a price. Such routes can be computed by routing engines as presented in [LV11].

The first step of the search is illustrated in Fig. 2.3 (on page 27). Given a starting location and a destination location, the first step computes and delivers all *reasonable individual routes* to travel from the starting location to public transport stations (in the near of the starting location) and from public transport stations (in the near of the destination location) to the destination location. Whether a route is reasonable or not depends on its duration. For each individual mode of transport, a specific maximum duration is defined. Routes which have durations longer than this maximum value are not reasonable. These maximum durations, also called *ranges*, are parameters which can be specified by the passengers.

Station-based modes For car sharing and bike sharing, we focus on the station-based model as illustrated in Fig. 1.4 (on page 12); such an individual route consists of three parts as follows. For the initial part of intermodal connections, an individual route begins at the starting location; the passenger walks to a car sharing or a bike sharing station in order to collect the car or the bike respectively. He/she then drives to a car sharing or a bike sharing station in the near of a public transport station, and returns the car or the bike respectively. Last, the passenger walks to the public transport station. For the end part of intermodal connections, the structure of individual routes is analogous.

Modes without stations In contrast, individual routes for modes of transport such as walks, taxis, individual bikes, and individual cars begin at the starting location and end at a public transport station. For the end part of intermodal connections, the structure of individual routes is analogous.

Note: We assume that passengers can park individual cars and bikes in the near of public transport stations.

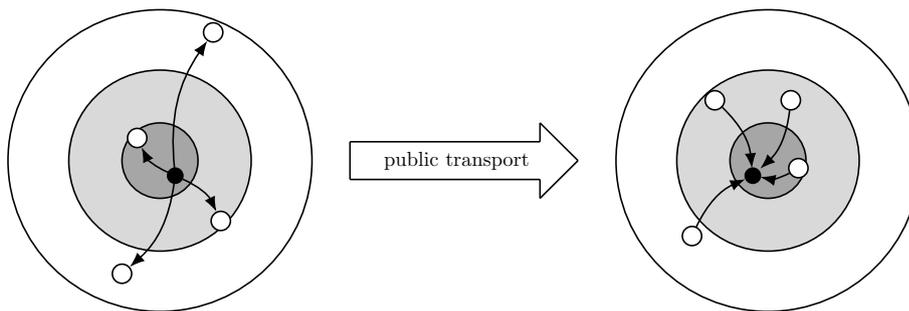


Figure 2.3: The figure illustrates the structure of intermodal connections as well as, respectively, three different ranges of three individual modes of transport around the starting location (left) and around the destination location (right). The thin arrows illustrate the movements by individual modes of transport from the starting location to public stations (left) and from public stations to the destination location (right). By way of example, the inner, middle, and outer circle are the ranges for walk, bike, and car respectively. This figure has been taken from Figure 6.1 in [Gün15].

2.3.2 The Second Step: Extending the Graph

The next step is to extend the time-dependent graph (cf. Sect. 2.1.2) that models the timetable by the individual routes computed for the end parts of the intermodal connections. For this, the graph is extended for each search query, and this modification is rolled back after the query is processed.

Modeling the end part First, we add to the graph a node that represents the destination location of the connections; we denote it by the *destination node*. We then model each individual route from a public transport station to the destination location by an *individual transport edge* from the station node of that station to the destination node. In contrast to route edges (cf. Sect. 2.1.2), the duration of an individual transport edge is not time-dependent; it has a fixed travel time and a fixed price; these values are taken from the respective individual route. The transfer cost equals zero.

Note: To model the start part of the intermodal connections, where individual modes of transport are used, we modify the Pareto Dijkstra algorithm as will be explained in Sect. 2.3.3.

2.3.3 The Third Step: Pareto Dijkstra

In the last step, the Pareto Dijkstra algorithm from Sect. 2.2 is applied to find all Pareto optimal intermodal connections. In order to model the individual routes for the initial parts of intermodal connections, the procedure of creating start labels (cf. Sect. 2.2.2) is modified. Briefly, we create start labels for each individual route that is computed in the first step of the search for the initial parts of intermodal connections

(cf. Sect. 2.3.1); a more precise explanation follows.

Modeling the initial part Each individual route computed for the initial parts of intermodal connections is modeled as follows. Recall that such an individual route ends at a public transport station $s \in S$, and it has a travel time (time cost) as well as a price. In the on-trip scenario where, in the query q , a point in time $ontrip_time(q)$ is given as the start time of the journey at the starting location, we merely create a start label at the station node of the station s . The timestamp of this label is set to $ontrip_time(q)$ plus the travel time of the individual route.

On the other hand, if a time interval $[begin(q), end(q)]$ is given as the start time of the journey, we create a new label for each departure event $e_{dep} \in E_{dep}^s$ at s where

$$sched(e_{dep}) - \delta \in [begin(q), end(q)];$$

the value δ is the travel time of the individual route. The new label is associated to the route node that represents e_{dep} , and the timestamp of the new label is set to $sched(e_{dep})$.

In both cases, the time cost of the start label and its price (if price is a Pareto criterion) are set to the respective values of the individual route. This concludes the search for Pareto optimal intermodal connections. For more details, we refer to [Gün15].

2.4 Conclusion

In this chapter, we discussed the time-expanded and the time-dependent graph model; both can be applied in order to model a timetable. The time-dependent model is more efficient in modeling timetables including short-range local transit such as buses and streetcars. For a timetable without short-range transit, the time-expanded model could be applied as well; advantageously, it correctly models platform-specific interchange times. We then explained the Pareto concept as a quality measurement for train and intermodal connections. We also discussed the Pareto Dijkstra algorithm computing Pareto optimal train and intermodal connections. In the rest of this work, we will use the models and the methods presented in this chapter.

```

Data: timetable  $TT$ ; query  $q$ 
Result: set of all Pareto optimal labels at the destination
1  $DL \leftarrow \emptyset$ ;                                /* set of labels at the destination */
2 foreach  $v \in V$  do
3   |  $L(v) \leftarrow \emptyset$ ;                /* initialize the set of labels at each node */
4 end
   /* Initialize the priority queue with the start labels          */
5  $pq \leftarrow \text{create\_start\_labels}()$ ;
6 while  $pq$  is not empty do
7   |  $\hat{l} \leftarrow pq.\text{pop}()$ ;
8   |  $v \leftarrow$  the node of  $\hat{l}$ ;
9   | foreach  $(v, w) \in A$  do                    /* outgoing edges of  $v$  */
10  |   |  $l \leftarrow \text{create\_label}(\hat{l}, (v, w))$ ;
11  |   | /* check whether  $l$  is dominated          */
12  |   | if  $\nexists l' \in L(w) : l' \text{ dominates } l$  then
13  |   |   | if  $w$  is at the destination then
14  |   |   | |  $DL \leftarrow DL \cup \{l\}$ ;
15  |   |   | else
16  |   |   |   |  $pq \leftarrow pq \cup \{l\}$ ;
17  |   |   |   |  $L(w) \leftarrow L(w) \cup \{l\}$ ;
18  |   |   |   | /* Remove all labels dominated by  $l$           */
19  |   |   |   | foreach  $l' \in L(w) : l \text{ dominates } l'$  do
20  |   |   |   | |  $L(w) \leftarrow L(w) \setminus \{l'\}$ ;
21  |   |   |   | end
22  |   |   | end
23  |   | end
24 end
    $DL$ ;

```

Algorithm 2.1: The Pareto Dijkstra algorithm. Labels and the priority queue as well as the functions `create_label`, `create_start_labels`, `filter_labels`, `pop`, and `push` are as defined in Sect. 2.2.2.

Chapter 3

Reliability Assessment

3.1 Introduction

Timetable information systems basically compute and deliver train connections that optimize the criteria of travel time, number of transfers, and price. The train connections delivered are feasible according to the timetable that is valid at the computation time. In other words, such a train connection allows a passenger to travel from his/her start station to his/her destination station as long as the trains in the train connection adhere to the timetable that is valid at the computation time. In a real-world train network, unfortunately, train delays are common and disrupt the timetable. Moreover, train cancelations and reroutings affect the daily operation of trains.

From Sect. 1.3 (cf. *Feasible transfers*), recall that the feasibility of a transfer activity $(e_{dep}^{s1,tr1}, e_{arr}^{s2,tr2}) \in TRANS$ depends on the real times of the events $e_{dep}^{s1,tr1}$ and $e_{arr}^{s2,tr2}$. If $tr1$ arrives too late, the transfer breaks; the train connection becomes infeasible according to the real times. Delays which result in infeasible train connections are not infrequent.

Probability of success An important characteristic of each train connection is its *reliability* regarding disruptions in the timetable. In our model, the reliability of a train connection c is defined by the *probability* that, when taking c , passengers can travel to their destination even if delays disrupt the timetable; meaning the probability that c will be feasible according to real times (cf. Sect. 1.3, *Feasible train connections*). We denote this probability as the *probability of success*.

The probability of the success of a train connection depends on the transfers in the train connection. Train connections with transfers that tend to break have a lower probability of success compared to train connections with transfers that retain a high probability of feasibility. The definition of *probability of success* also applies to transfers; it is the probability that the transfer will be feasible according to real times (cf. Sect. 1.3, *Feasible transfers*).

The method of assessing the reliability In this chapter, we present a method of assessing the reliability of a train connection; the method computes the probability of the success of the train connection. Our method is based on realistic, stochastic predictions for the real times of the departure and arrival events in the timetable. Each of these predictions is a probability distribution; more precisely, for each event $e \in E$ and for each time $t \in TIMES$, we compute the probability that e takes place at the time t ; that is, the real time of e equals t . These probability distributions are computed by taking into account historical delay data, the current situation in the railway network, and the interdependencies between the trains. The interdependencies between the trains are caused by delay managements; trains wait for the passengers of other delayed trains. Our computational study in Sect. 9.3 demonstrates that our reliability assessments are realistic and accurate.

Note: As discussed in Sect. 3.2, the stochastic predictions have been introduced in [BGMHO11].

Overview In Sect. 3.2, we discuss the state of the art and related methods for reliability assessment. Our method of assessing the reliability is based on a model with stochastic predictions for the real event times; the definition of the model can be found in Sect. 3.3. In Sect. 3.4 and 3.5, we discuss the formulas employed to compute these distributions. In Sect. 3.6 and 3.7, we present our method of computing the probability of success for transfers and train connections. After that, in Sect. 3.8, we present approaches to precomputing the probability distributions of the events in the timetable in order to accelerate the reliability assessment of train connections. We discuss, in Sect. 3.9, how to update the probability distributions for the random event times according to the real-time incidents introduced in Sect. 1.3.1. In Sect. 3.10, we analyze the asymptotic complexity of our reliability assessment method. In Sect. 3.11, we discuss an assumption of independence which is a prerequisite for our reliability assessment method. In Sect. 3.12, we address how train cancelations and reroutings could be incorporated into the reliability assessment. Finally, in Sect. 3.13, we offer details on our implementation of the reliability assessment method presented in this chapter. This chapter is concluded in Sect. 3.14.

3.2 Related Work

The computation of the probability of the success of a train connection, considering (I) the interdependencies between trains which could have effects on the feasibility of the train connection as well as (II) the fact that train connections can consist of multiple transfer activities, has not been addressed so far to the best of our knowledge.

The authors of [MHS09] presented an efficient model to propagate deterministic delays through train networks; a computation of the probability of success is not supported by their model. Stochastic delay propagations and predictions of train delays have been investigated in various works; examples are the works presented by [MM07, BGMHO11, CK94, MDnP10].

A method of assessing the reliability of a train connection is evaluating the buffer times of the train connection's transfer activities; this method is addressed in [Meh07, Sch09]. The buffer time of a transfer activity $(e_{arr}^{s1,tr1}, e_{dep}^{s2,tr2}) \in TRANS$ is the time difference

$$(\text{sched}(e_{dep}^{s2,tr2}) - \text{sched}(e_{arr}^{s1,tr1})) - \text{min_dur}(e_{arr}^{s1,tr1}, e_{dep}^{s2,tr2}).$$

Unfortunately, a mere analysis of the buffer times could result in inaccurate assessments. While a specific buffer time could be even more than necessary for a transfer activity to be reliable, it could be insufficient for another one.

Lemnian et al. proposed a method of computing the probability of the success of a transfer activity [LRR⁺14]. Their model relies on probability distributions for the durations of the train moves (as will be discussed in Sect. 3.3) but does not take into account the interdependencies between the trains which could affect the probability of the success of the train connection; waiting time rules (cf. Sect. 1.3) are only taken into account for the trains which are directly involved in the transfer activity but not for the other departure events in the train connection. Moreover, for the reliability assessment, each transfer in a train connection is evaluated in isolation. In contrast, our method considers the dependencies between the transfer activities in each train connection.

The general idea of modeling the real event times by random variables has already been introduced in [BGMHO11]; the stochastic model from Sect. 3.3 as well as the interdependencies and the formulas discussed in Sect. 3.5.1 and 3.5.2 rely on the work published by [BGMHO11]. However, in our opinion, the terminology and the formulas presented here are easier to read and to understand. Moreover, we designed our model such that we do not require an assumption made by them. In contrast to their model, the real event time of the arrival events can be earlier than their scheduled event time.

Publication of the author In [KSWZ12], we have already presented the formula for the computation of the probability of the success of a transfer activity. We used there a terminology and a formula different than presented here. But both formulas are equivalent as will be proved in Appx. A.1.

3.3 Stochastic Model

In this section, we introduce our stochastic model that allows us to compute stochastic predictions for the real times of the departure and arrival events in the timetable. As explained in Sect. 3.1 (cf. *The method of assessing the reliability*), these stochastic predictions are probability distributions for the events. Our method of assessing the reliability of a train connection is based on these probability distributions. In addition, in Chapters 4–8, we will discuss approaches of computing reliable train connections and *journeys*; these approaches are based on the stochastic model from this section. First, we begin with an informal description of our stochastic model; we then present the formal definition.

3.3.1 Informal Description

From Sect. 1.1 and 1.3.2, recall that each departure and arrival event has a scheduled and a real event time. While the scheduled time may be looked up in the timetable, the real event time depends on unforeseen circumstances, and it is not safely known until the event takes place. Among the reasons are waitings of trains for each other (due to delay management policies from cf. Sect. 1.3, *Maximum waiting times*), signal disturbances, technical malfunctions, human resource problems, emergencies, weather conditions, etcetera. These factors influence the real event time of each departure and arrival event.

Random event times In our model, for each event, there is a random variable modeling its unknown real time; this random variable is called the *random event time* of the event. The delay origins (see above) are modeled by defining stochastic dependencies of random event times. We distinguish between delays arisen from the interdependencies between the events in the timetable and delays from other origins such as technical malfunctions, signal disturbances, etc. The interdependencies between the events are modeled by defining interdependencies between the random times of the events. Thus, the random event time of each event stochastically depends on the random event times of a specific set of events in the timetable as will be detailed in Sect. 3.5.1. To incorporate the other delay origins into our model, we introduce a *random prepared time* for each departure event as well as a *random duration* for each train move.

Note: The interdependencies between the events are detailed in Sect. 3.5.1. In short, they are caused by the waiting times of the trains as defined in Sect. 1.3.2 (cf. *Maximum waiting times*). In addition, each event of each train stochastically depends on the preceding events of that train; for example, a train cannot depart from a station before

it arrives at that station.

Random prepared times For each departure event $e_{dep}^{tr,s} \in E_{dep}$, the *prepared time* denotes the time when the train tr is ready to depart from s . According to the timetable, this time equals $sched(e_{dep}^{tr,s})$. In contrast, the real prepared time is unknown and can be later than $sched(e_{dep}^{tr,s})$ because of a delayed train delivery (only for the very first departure events), technical malfunctions, human resource problems, emergencies, weather conditions, etcetera. Hence, for each departure event, we introduce a random variable modeling the unknown, real prepared time of the departure event; it is denoted as the *random prepared time*. The random event time of a departure event consequently has a stochastic dependency on its random prepared time.

Random train move durations We furthermore introduce, for each train move, a random variable denoted as the *random train move duration*; it models deviations from the scheduled duration of the train move. Delays during a train move could result in a delayed arrival. On the other hand, it is possible that the duration of the train move is shorter than scheduled. The most of the problems resulting in delayed prepared times could occur during a train move as well. In addition, trains could be instructed to wait for a track to be freed.

The real duration of a train move depends on the real event time of its departure event. This dependency may have positive and negative effects and, hence, covers both fundamental cases: the delayed departure e_{dep} causes further delays during the move, or the train operator decides to “step on the gas” along (e_{dep}, e_{arr}) in order to make up for the delayed departure e_{dep} .

A formal description of the random variables introduced above can be found in Sect. 3.3.2.

3.3.2 Formal Description

Each random variables in our model is *discrete*; it has a *discrete probability distribution* with a *discrete probability density function* $f : TIMES \mapsto [0, 1] \subset \mathbb{R}$ where $\sum_{t \in TIMES} f(t) = 1$. For a random variable X with the probability density function f_X , the probability that X assumes the value t equals $P(X = t) = f_X(t)$. The *support* of f_X comprises each time $t \in TIMES$ where $f_X(t) > 0$; it is denoted by $supp(f_X)$.

Random event times For each event $e \in E$, the discrete random variable X_e denotes the *random event time* of e . There is a joint probability distribution with the dimension $|E|$ over all random variables $X_e, e \in E$. For each event $e \in E$, the probability density

function of the marginal distribution of X_e is denoted by $\phi(e)$. The value $\phi(e, t)$ equals $P(X_e = t)$; the probability of the event e taking place at the time t .

In our model, we assume that trains never depart earlier than scheduled. In contrast, arrivals earlier than scheduled are possible.

Random prepared times For each departure event $e_{dep} \in E_{dep}$, the random variable $X_{e_{dep}}^p$ denotes the random time when the train is prepared for the departure. There is a joint probability distribution over all random variables $X_{e_{dep}}^p, e_{dep} \in E_{dep}$ with the dimension $|E_{dep}|$. For each departure event $e_{dep} \in E_{dep}$, the probability density function of the marginal distribution of $X_{e_{dep}}^p$ is denoted by $\xi(e_{dep})$. The value $\xi(e_{dep}, t)$ equals $P(X_{e_{dep}}^p = t)$; the probability that the real prepared time of e_{dep} equals t .

Random durations of train moves The discrete random variable X_{tm} denotes the random duration of the train move $tm = (e_{dep}, e_{arr}) \in TM$. For each time $t_{dep} \in TIMES$, there is a joint probability distribution $\Theta(TM, t_{dep})$ with the dimension $|TM|$ over all random variables $X_{tm}, tm \in TM$. For each $tm \in TM$ and each $t_{dep} \in TIMES$, the probability density function of the marginal distribution of X_{tm} is denoted by $\theta(tm, t_{dep})$. The parameter t_{dep} models the dependency of the random duration of the train move (e_{dep}, e_{arr}) on the real departure time of e_{dep} . For the times $t_{dep}, t_{arr} \in TIMES$ and $tm = (e_{dep}, e_{arr}) \in TM$, we define

$$\begin{aligned} \theta(tm, t_{dep}, t_{arr}) &= P(X_{tm} = t_{arr} - t_{dep} \mid X_{e_{dep}} = t_{dep}) \\ &= P(X_{e_{arr}} = t_{arr} \mid X_{e_{dep}} = t_{dep}); \end{aligned}$$

in words, the value $\theta(tm, t_{dep}, t_{arr})$ is the conditional probability that the real duration of tm equals $t_{arr} - t_{dep}$ given that the departure event e_{dep} takes place at the time t_{dep} .

This concludes our stochastic model. In Sect. 3.4, we focus on the computation of the marginal distributions for random prepared times and random train moves durations. The interdependencies between the random times of the events as well as the computation of their probability distributions are discussed in Sect. 3.5.

3.4 Distributions for Prepared Times and Train Move Durations

The computation of probability distributions for random train move durations and random prepared times (cf. Sect. 3.3) is not on the focus of this work. In our computational study in Chapter 9, for these random variables, we mainly use probability distributions provided to us by the German railway company. However, to make this

work self-contained, in this section, there is an excursion into the computation of probability distributions for these random variables. A reader, who is not interested in this step, may skip this section and continue with Sect. 3.5.

The random variables for random prepared times and random durations of train moves model a broad range of delay reasons. The more realistic their probability distributions are, the more accurate are computations basing on them. We discuss two general ideas to obtain these probability distributions: either to learn them from historical data or to generate realistic ones. While the former is more accurate and convincing, the latter could be a solution if no historical data is available.

Note: In the bachelor's theses by Julian Gross [Gro15] and Marco Plaue [Pla16], we evaluated approaches to the learning of probability distributions.

3.4.1 Prepared Times: Learn from Historical Data

For a timetable TT_{fut} with a validity period in the future, the probability distributions for the random prepared times of the departure events (cf. Sect. 3.3) can be learned from historical data. More precisely, we learn from a given timetable TT_{hist} with a validity period $TIMES_{hist}$ in the past where the real time $real(e)$ of (almost) each event $e \in E_{hist}$ is already known. Events in the set E_{hist} in TT_{hist} with unknown real times (unknown because of lack of data) are excluded from the learning procedure.

We learn probability distributions for random prepared times in two successive steps. In the first step, we divide all departure events in E_{dep} into disjunct subsets; this step is performed equally for TT_{hist} and TT_{fut} . Each of these subsets is denoted as a *class*. Thus, for each of TT_{hist} and TT_{fut} , we obtain a set of classes; these two sets are denoted by $\mathcal{EC}_{dep}^{hist} \subset 2^{E_{dep}^{hist}}$ and $\mathcal{EC}_{dep}^{fut} \subset 2^{E_{dep}^{fut}}$ respectively. This classification has the following background. The random prepared times of all departure events in the same class can be described by the same probability distribution since these departure events are equivalent according to specific *properties* as will be explained below (cf. *First step: classification*). The classification is performed by a *classification function* $c : E_{dep} \mapsto 2^{E_{dep}}$ equally for TT_{hist} and for TT_{fut} . Thus, for each class in $\mathcal{EC}_{dep}^{hist}$, there is a respective class in \mathcal{EC}_{dep}^{fut} and vice versa.

In the second step of the learning procedure, for each class in \mathcal{EC}_{dep}^{fut} , a probability distribution is learned by evaluating the real prepared times of the departure events in the respective class in $\mathcal{EC}_{dep}^{hist}$. The result is a probability distribution that is valid for the random prepared times of each departure event in that class. In other words, this probability distribution describes the random prepared time of each departure event in the respective class in \mathcal{EC}_{dep}^{fut} .

The two steps of the learning procedure are detailed in the following.

First step: classification For the classification of the departure events, we define the classification function $c : E_{dep} \mapsto 2^{E_{dep}}$. A set of properties of the departure events is specified; the classification function is designed such that all departure event sharing the same properties—from the set of properties specified—are assigned to the same class. The set of the properties selected for the classification should be defined depending on the characteristics of the timetable and the railway network. First, the train category (such as intercity express, regional train, streetcar, etc.) usually influences the frequency and the extent of delayed prepared times. For example, for different train categories, different types of vehicles are used. Two further properties potentially influencing the prepared time of each event are the scheduled departure time of the event (e.g. morning, noon, after noon, evening) and the size of the station where the event takes place.

Note: The validity period of TT_{hist} should be as long as it contains sufficient samples for each class. Theoretically, the more crucial properties of the departure events are selected for the classification, the more accurate is the results will be. In practice, for a fine granular view on the data, we are faced by small numbers of samples. The significance and expressiveness of the results consequently suffer from a fine granular view on the data. Therefore, a suitable set of parameters needs to be chosen depending on the timetable TT_{hist} and portion of events in E_{hist} for which data about the real event times are actually available in TT_{hist} .

Note: Historical data contain outliers and potential errors which must be reasonably managed.

Second step: computation Once the departure events in TT_{hist} are classified, for each class in $\mathcal{EC}_{dep}^{hist}$, we compute a *histogram* of the differences between the real and the scheduled event times over all departure events in that class. Then, a probability density function for that class is obtained by normalizing the histogram to 1. These probability density function describes the random prepared time of each departure event in the respective class in TT_{fut} (see below).

Assignment to $\xi(\mathbf{e}_{dep}, \mathbf{t}_{dep})$ Once probability density functions are computed as discussed above, they are assigned to the probabilities $\xi(e_{dep}, t_{dep})$ (cf. Sect. 3.3.2, *Random prepared times*) which describe the random prepared time of each departure event $e_{dep} \in E_{dep}^{fut}$ of the timetable TT_{fut} . More precisely, for each departure event $e_{dep} \in E_{dep}^{fut}$, we apply the classification function in order to find the respective class in $\mathcal{EC}_{dep}^{hist}$. Then, for each time $t_{dep} \in TIMES_{fut}$, the probability for the value $t_{dep} - sched(e_{dep})$ is retrieved—from the probability density function of that class—

and assigned to $\xi(e_{dep}, t_{dep})$ (recall that the probability density function comprise a probability for each deviation from the scheduled event time).

Interdependencies between the events From Sect. 3.3.1, recall that each departure delay could either be caused by the interdependencies between the trains in the timetable or have another origin like technical malfunctions, human resource problems, etc. By random prepared times, we model the latter. Some of the delays in the timetable TT_{hist} could be caused by interdependencies between the events; consequently, they have delay origins which we do not model by random prepared times. For correct results, these delays should be excluded from the learning procedure and the computation of the probabilities.

3.4.2 Train Move Durations: Learn from Historical Data

For a timetable TT_{fut} with a validity period in the future, the probability distributions for the random train move durations (cf. Sect. 3.3.2) can be learned from historical data. More precisely, we learn from a given timetable TT_{hist} with a validity period $TIMES_{hist}$ in the past where the real time $real(e)$ of (almost) each event $e \in E_{hist}$ is already known. Events in the set E_{hist} in TT_{hist} with unknown real times (unknown because of lack of data) are excluded from the learning procedure.

The learning procedure is similar as for random prepared times (cf. Sect. 3.4.1); it has two successive steps. In the first step, we divide all train moves in TM into disjunct subsets; this step is performed equally for TT_{hist} and TT_{fut} . Each of these subsets is denoted as a *class*. Thus, for each of TT_{hist} and TT_{fut} , we obtain a set of classes; these two sets are denoted by $\mathcal{TMC}_{hist} \subset 2^{TM_{hist}}$ and $\mathcal{TMC}_{fut} \subset 2^{TM_{fut}}$ respectively. This classification has the following background. The random durations of all train moves in the same class can be described by the same probability distribution since these train moves are equivalent according to specific *properties* as will be explained below (cf. *First step: classification*). The classification is performed by a *classification function* $c : TM \mapsto 2^{TM}$, equally for TT_{hist} and for TT_{fut} . Thus, for each class in \mathcal{TMC}_{hist} , there is a respective class in \mathcal{TMC}_{fut} and vice versa.

In the second step of the learning procedure, for each class in \mathcal{TMC}_{fut} , a probability distribution is learned by evaluating the real durations of the train moves in the respective class in \mathcal{TMC}_{hist} . The result is a probability distribution that is valid for the random train move durations for each train move in that class. In other words, this probability distribution describes the random duration of each train move in the respective class in \mathcal{TMC}_{fut} .

The two steps of the learning procedure are detailed in the following.

First step: classification For the classification of the train moves, we define the classification function $c : TM \mapsto 2^{TM}$. A set of properties of the train moves is specified; the classification function is designed such that all train moves sharing the same properties—from the set of properties specified—are assigned to the same class. The set of the properties selected for the classification should be defined depending on the characteristics of the timetable and the railway network. For the classification of the train moves, the following properties could be taken into account:

- *Departure delay*: For a train move $tm = (e_{dep}, e_{arr})$, the delayed departure—i.e. the difference between the real and the scheduled event time of e_{dep} —influences the real duration of tm . A delayed departure could cause further delays during the move, or the train operator could decide to “step on the gas” along (e_{dep}, e_{arr}) in order to make up for the delayed departure e_{dep} .
- *Train categories*: The train category usually influences the delays. For example, for different train categories, different types of vehicles are used; as a consequence, they have different capabilities to make up for delays.
- *Scheduled duration*: Short and long rides could be affected from delays to different extents. Train moves over longer distances (resulting in longer scheduled durations) could be more likely to be affected by delays that are caused during train moves. On the other side, over a longer distance, the chance to make up for delays is greater. To model these influences, the scheduled durations of train moves should be incorporated into the definition of the classification function c .
- As further parameters, we could take into account the departure and arrival stations of the train moves, information about the infrastructure of the railway network (e.g. tracks), temporal attributes, etc.

For each train move $tm = (e_{dep}, e_{arr})$ in the timetable TT_{fut} , all properties listed above can be retrieved from TT_{fut} except for the real time of e_{dep} since it is unknown. Consequently, we cannot classify the train moves in TT_{fut} by a classification function that requires the departure delay as a parameter. To overcome this problem, we use a two steps classification. First, we build the classes \mathcal{TMC}_{hist} and \mathcal{TMC}_{fut} by employing a classification function that incorporates all properties specified except for the departure delay. We then perform a *second classification*; we divide all train moves in each class from the set \mathcal{TMC}_{hist} into a set of *subclasses* such that the departure events of all train moves in the same subclass have the same departure delay. Then, we proceed as follows.

Second step: computation Once the train moves in TT_{hist} are classified into \mathcal{TM}_{hist} and subsequently into subclasses according to the departure delays (see above), for each subclass, we compute a *histogram* of the differences between the real and the scheduled durations over all train moves in that subclass. Then, a probability density function for that subclass is obtained by normalizing the histogram to 1. These probability density function describes the random duration of each train move in the respective class in TT_{fut} (see below).

As discussed above, for TT_{fut} , the departure delay is unknown. Therefore, the real departure time t_{dep} of the event e_{dep} of each train move (e_{dep}, e_{arr}) is an additional parameter for the value $\theta(tm, t_{dep}, t_{arr})$; for each departure delay expected, a probability density function is computed.

Assignment to $\theta(tm, t_{dep}, t_{arr})$ Once probability density functions are computed as discussed above, they are assigned to the probabilities $\theta(tm, t_{dep}, t_{arr})$ (cf. Sect. 3.3.2, *Random durations of train moves*) which describe the random duration of each train move $tm \in \mathcal{TM}_{fut}$ of the timetable TT_{fut} . More precisely, for each train move $(e_{dep}, e_{arr}) \in \mathcal{TM}_{fut}$ and two times $t_{dep}, t_{arr} \in \mathcal{TIMES}_{fut}$, we apply the classification function c for (e_{dep}, e_{arr}) and find the respective class in \mathcal{TM}_{hist} . Finally, from the probability density function of the subclass for the departure delay $t_{dep} - sched(e_{dep})$, the probability for the value

$$(t_{arr} - t_{dep}) - (sched(e_{arr}) - sched(e_{dep}))$$

is delivered and assigned to $\theta(tm, t_{dep}, t_{arr})$.

Note: In our computational study in Chapter 9, for the random train move durations, we use probability distributions provided to us by the German railways. These probability distributions are classified by the real departure time, train category, and scheduled train move duration.

Note: The validity period of TT_{hist} should be as long as it contains sufficient samples for each class. Historical data contain outliers and potential errors which must be reasonably managed.

3.4.3 Generate Distributions

Learning probability distributions requires historical data. For the case that historical delay data is not available, realistic probability distributions for prepared times and train move durations could be generated. In our computational study in Sect. 9.4, for a subset of all train moves, we use probability distributions provided to us by Deutsche

Bahn AG. For the rest of the train moves, we generate realistic probability distributions using the Cauchy distribution [NIS17a]. The probability density function of a Cauchy distribution is characterized by a location parameter and a scale parameter. The former defines the position of the function's peak on the x-axis; the latter stretches out the graph [NIS17a]. To generate a probability density function describing the random duration of a train move, we define the location and the scale parameter depending on the scheduled duration of the train move and the delay of its departure event. The scale parameter is defined such that we obtain wider distributions for train moves with longer durations. The location parameter is defined such that in case of a delayed departure the probability that the real duration is shorter than the scheduled duration increases; we assume that the train makes up for the delayed departure. In Appx. A.3, we present some examples.

3.5 Distributions for Random Event Times

In Sect. 3.3.2, we introduced random event times; for each event $e \in E$, there is a probability density function $\phi(e)$ describing the random event time of e . Given a timetable and the probability density functions for prepared times as well as train move durations (as introduced in Sect. 1.1 and 3.3.2), for each $e \in E$, the probability density function $\phi(e)$ can be determined. In Sect. 3.5.1, we explain the interdependencies between the events and their random event times. We then discuss, in Sect. 3.5.2, the formulas to compute the probabilities $\phi(e, t)$ for each event $e \in E$ and each time $t \in TIMES$. Our reliability assessment methods for transfers and train connections (cf. Sect. 3.6 and 3.7) are based on those formulas.

3.5.1 The Interdependencies Between the Events

Fig. 3.1 (on page 43) demonstrates an illustrative example of dependencies between departure and arrival events. In the following, we precisely explain these interdependencies.

3.5.1.1 The Dependencies of Departure Events

After the arrival of a train at a station and prior to its subsequent departure, the minimum dwell time has to be obeyed according to the definition of dwell times from Sect. 1.1 (cf. *Dwell activities and times*). In addition, a train does not depart until it is prepared for the departure as introduced in Sect. 3.3 (cf. *Random prepared times*). Furthermore, each train potentially waits for other delayed trains as introduced in Sect. 1.3 (cf. *Maximum waiting times*).

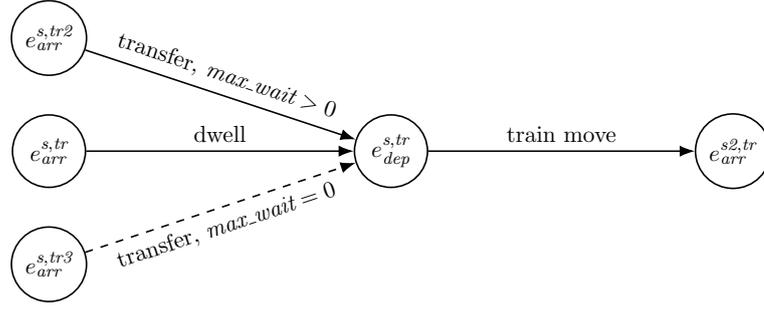


Figure 3.1: The figure illustrates a section of a train network. The departure event $e_{dep}^{s,tr}$ potentially waits for the arrival event $e_{arr}^{s,tr2}$ but not for the arrival event $e_{arr}^{s,tr3}$. Thus, it depends on the arrival events $e_{arr}^{s,tr}$ and $e_{arr}^{s,tr2}$. The arrival event $e_{arr}^{s2,tr}$ only depends on $e_{dep}^{s,tr}$.

Set $E_{arr}^{depend}(e_{dep}^{s,tr})$ Hence, the random event time of each departure event $e_{dep}^{s,tr} \in E_{dep}$ stochastically depends on a set of arrival events; we denote this set by $E_{arr}^{depend}(e_{dep}^{s,tr})$. It comprises the following arrival events. First, if $e_{dep}^{s,tr}$ is not the very first departure event of the train tr , the arrival event $e_{arr}^{s,tr}$ exists and is a member of the set $E_{arr}^{depend}(e_{dep}^{s,tr})$. Second, the set $E_{arr}^{depend}(e_{dep}^{s,tr})$ comprises each arrival event $e_{arr}^{s,tr2} \in E_{arr}^s$ where $max_wait(e_{arr}^{s,tr2}, e_{dep}^{s,tr}) > 0$ (cf. Sect. 1.3, *Maximum waiting times*). Last, in order to take into account the random prepared time of $e_{dep}^{s,tr}$, we add to the set $E_{arr}^{depend}(e_{dep}^{s,tr})$ an artificial event $e_{artific}^{s,tr}$. For $e_{artific}^{s,tr}$, we introduce an artificial dwell activity $dw = (e_{artific}^{s,tr}, e_{dep}^{s,tr})$ where $min_dur(dw) = 0$. Moreover, we define $\phi(e_{artific}^{s,tr}, t) = \xi(e_{dep}^{s,tr}, t)$ for $t \in TIMES$.

Consequently, the probability density function $\phi(e_{dep})$ of a departure event $e_{dep} \in E_{dep}$ is fully determined by the probability density functions of the arrival events in the set $E_{arr}^{depend}(e_{dep})$.

3.5.1.2 The Dependencies of Arrival Events

The random time of each arrival event $e_{arr}^{s,tr} \in E_{arr}$ stochastically depends on the random time of the departure event $e_{dep}^{s2,tr}$ of the train move $tm = (e_{dep}^{s2,tr}, e_{arr}^{s,tr})$. The probability density function $\phi(e_{arr}^{s,tr})$ is fully determined by the probability density function $\phi(e_{dep}^{s2,tr})$ and the probability distribution of the random duration of the train move tm (cf. Sect. 3.3.2).

Note: Our network is similar to a Bayesian network [Dar09] in the following sense. We have a directed acyclic graph, where random variables X_e , $e \in E$, build the nodes of the network. Each arrival event e_{arr} of a train move (e_{dep}, e_{arr}) has one predecessor, the event e_{dep} . Set $E_{arr}^{depend}(e_{dep})$ comprises the predecessors of each departure event e_{dep} . There is an arc from each predecessor of a node to the node itself. Each random variable depends on its predecessors. However, in contrast to Bayesian networks, we do

not explicitly create a conditional probability table for the random variables. Instead, for each random variable X_e , we have a one-dimensional probability distribution.

3.5.2 The Calculation of the Probabilities

In this section, we introduce the formulas to compute the probabilities $\phi(e, t)$ for each event $e \in E$ and each time $t \in TIMES$.

3.5.2.1 The Formula for Departure Events

According to Sect. 3.5.1, the dependency of $e_{dep}^{s,tr}$ on some arrival event $e_{arr}^{s,tr2} \in E_{arr}^{depend}(e_{dep}^{s,tr})$ is based either on the minimum transfer time (in case $tr \neq tr2$) or on the minimum dwell time (in case $tr = tr2$). Both of these times are described by $min_dur(e_{arr}^{s,tr2}, e_{dep}^{s,tr})$ as defined in Sect. 1.1 and 1.2. Additionally, we define the latest real event time of $e_{dep}^{s,tr}$ that can be caused by $e_{arr}^{s,tr2} \in E_{arr}^{depend}(e_{dep}^{s,tr})$ as follows:

$$t_{dep}^{latest}(e_{arr}^{s,tr2}, e_{dep}^{s,tr}) = \begin{cases} sched(e_{dep}^{s,tr}) + max_wait(e_{arr}^{s,tr2}, e_{dep}^{s,tr}) & \text{if } tr \neq tr2, \\ +\infty & \text{otherwise.} \end{cases} \quad (3.5.1)$$

The latter case models the trivial fact that each train waits for its own arrival without any limit.

Informal description The stochastic event that a departure event $e_{dep} \in E_{dep}$ takes place at the time $t_{dep} \in TIMES$ may be described as follows. First of all, e_{dep} may take place at t_{dep} only if e_{dep} does not have to wait for any arrival event $e_{arr} \in E_{arr}^{depend}(e_{dep})$ up to some time $t'_{dep} > t_{dep}$ according to the value $t_{dep}^{latest}(e_{arr}, e_{dep})$. For $t_{dep} = sched(e_{dep})$, this necessary condition is also sufficient since we assume that a train never departs earlier than its scheduled departure time. On the other hand, for $t_{dep} > sched(e_{dep})$, an additional condition must be fulfilled: e_{dep} has to wait exactly up to t_{dep} for at least one $e_{arr} \in E_{arr}^{depend}(e_{dep})$. These informal descriptions may be formalized as follows.

The formulas For each departure event $e_{dep} \in E_{dep}$ and each time $t_{dep} \in TIMES$ where $t_{dep} \geq sched(e_{dep})$, let $\psi(e_{dep}, t_{dep})$ denote the probability that the necessary condition (see above) is fulfilled. More precisely, $\psi(e_{dep}, t_{dep})$ is the probability that e_{dep} does not have to wait for any arrival event in $E_{arr}^{depend}(e_{dep})$ up to some time

$t'_{dep} > t_{dep}$. For the time $t_{dep} \geq sched(e_{dep})$, we obtain:

$$\begin{aligned} \psi(e_{dep}, t_{dep}) = & \prod_{e_{arr} \in E_{arr}^{depend}(e_{dep})} \left[P(X_{e_{arr}} + min_dur(e_{arr}, e_{dep}) \leq t_{dep}) \right. \\ & \left. + P(X_{e_{arr}} + min_dur(e_{arr}, e_{dep}) > \right. \\ & \left. \max\{t_{dep}, t_{dep}^{latest}(e_{arr}, e_{dep})\}) \right]. \end{aligned} \quad (3.5.2)$$

For $t_{dep} < sched(e_{dep})$, we define $\psi(e_{dep}, t_{dep}) = 0$ since we assume that a train never departs earlier than its scheduled departure time. Hence, for a time $t_{dep} \geq sched(e_{dep})$, we obtain

$$\phi(e_{dep}, t_{dep}) = P(X_{e_{dep}} = t_{dep}) = \psi(e_{dep}, t_{dep}) - \psi(e_{dep}, t_{dep} - 1). \quad (3.5.3)$$

The smallest value for $t_{dep} \in supp(\phi(e_{dep}))$ is $sched(e_{dep})$. Let $t_{max}(supp(\phi(e)))$ denote the largest time in $supp(\phi(e))$ for each event $e \in E$. More precisely, we define

$$t_{max}(supp(\phi(e))) = \max\{sched(e_{dep}), \max_{e_{arr} \in E_{arr}^{depend}(e_{dep})} \tilde{t}_{dep}^{max}(e_{dep}, e_{arr})\}$$

where the value $\tilde{t}_{dep}^{max}(e_{dep}, e_{arr})$ denotes the latest delayed departure time of e_{dep} that may be caused by e_{arr} when taking into account the function $\phi(e_{arr})$; it equals

$$\begin{aligned} & \text{cf. Eq. 3.5.1 (on page 44)} \\ \tilde{t}_{dep}^{max}(e_{dep}, e_{arr}) = & \min \left\{ \overbrace{t_{dep}^{latest}(e_{arr}, e_{dep})}^{\text{cf. Eq. 3.5.1 (on page 44)}}, \right. \\ & \left. t_{max}(supp(\phi(e_{arr}))) + min_dur(e_{arr}, e_{dep}) \right\}. \end{aligned}$$

Assumption of independence The formula to compute $\psi(e_{dep}, t_{dep})$ is based on the assumption that all random variables $X_{e_{arr}}$ are stochastically independent for $e_{dep} \in E_{dep}$ and each $e_{arr} \in E_{arr}^{depend}(e_{dep})$. This assumption is discussed in Sect. 3.11.

3.5.2.2 The Formula for Arrival Events

For each arrival event $e_{arr} \in E_{arr}$ of the train move $tm = (e_{dep}, e_{arr})$ and for each time $t_{arr} \in TIMES$, the probability $\phi(e_{arr}, t_{arr})$ is given by the convolution of $\phi(e_{dep})$ with the probability density functions for the real durations of tm :

$$\begin{aligned} \phi(e_{arr}, t_{arr}) &= P(X_{e_{arr}} = t_{arr}) \\ &= \sum_{t_{dep}=sched(e_{dep})}^{t_{arr}} \phi(e_{dep}, t_{dep}) \cdot \theta(tm, t_{dep}, t_{arr}). \end{aligned} \quad (3.5.4)$$

From Sect. 3.3.2, recall that $\theta(tm, t_{dep}, t_{arr})$ is the probability that the train move tm arrives at the time t_{arr} given that the departure is at the time t_{dep} . The range of the values in $\text{supp}(\phi(e_{arr}))$ is defined by the smallest and largest values for t_{arr} such that $\phi(e_{dep}, t_{dep}) \cdot \theta(tm, t_{dep}, t_{arr}) > 0$ for some time t_{dep} . Recall that values earlier than $\text{sched}(e_{arr})$ can be in $\text{supp}(\phi(e_{arr}))$.

Bayes' theorem Our formula to compute the probabilities $\phi(e_{arr})$ is based on Bayes' theorem. More precisely, we apply the *law of total probability* [NIS17b]; for stochastic events A and $B_1, B_2, \dots, B_n, n \in \mathbb{N}$, one has

$$P(A) = \sum_n P(A | B_n) \cdot P(B_n).$$

3.6 The Reliability Assessment of Transfers

In this section, we demonstrate our method of assessing the reliability of a transfer; that is, the computation of the *probability of the success* of the transfer activity.

The Probability of the Success of a Transfer For a given transfer activity $(e_{arr}^{s1, tr1}, e_{dep}^{s2, tr2}) \in TRANS$, the *probability of success* is the probability that the transfer from $tr1$ —arriving at $s1$ —into $tr2$ —departing from $s2$ —is feasible according to the real times of $e_{arr}^{s1, tr1}$ and $e_{dep}^{s2, tr2}$, as defined in Sect. 1.2 (cf. *Feasible Transfers*). Formally, it is the following probability:

$$P(X_{e_{arr}^{s1, tr1}} + \text{min_dur}(e_{arr}^{s1, tr1}, e_{dep}^{s2, tr2}) \leq X_{e_{dep}^{s2, tr2}}).$$

Note: From Sect. 1.2, recall that $s1 \neq s2$ if $(e_{arr}^{s1, tr1}, e_{dep}^{s2, tr2})$ is a transfer activity with a walk trip.

The calculation The probability of the success of a transfer activity $(e_{arr}, e_{dep}) \in TRANS$ equals

$$\begin{aligned} & P(X_{e_{arr}} + \text{min_dur}(e_{arr}, e_{dep}) \leq X_{e_{dep}}) \\ &= \sum_{t_{dep} \in \text{supp}(\phi(e_{dep}))} P(X_{e_{arr}} + \text{min_dur}(e_{arr}, e_{dep}) \leq t_{dep} \\ & \quad \cap X_{e_{dep}} = t_{dep}). \end{aligned} \tag{3.6.1}$$

The value $P(X_{e_{arr}} + \text{min_dur}(e_{arr}, e_{dep}) \leq t_{dep} \cap X_{e_{dep}} = t_{dep})$ in Eq. 3.6.1 is the probability that the transfer is feasible if e_{dep} takes place at the time t_{dep} ; in other words, the *joint* probability that e_{arr} takes place no later than $t_{dep} - \text{min_dur}(e_{arr}, e_{dep})$ and e_{dep} takes place at the time t_{dep} (recall that $X_{e_{dep}}$ potentially depends on $X_{e_{arr}}$). This probability is defined as follows:

$$\begin{aligned} & P(X_{e_{arr}} + \text{min_dur}(e_{arr}, e_{dep}) \leq t_{dep} \cap X_{e_{dep}} = t_{dep}) \\ &= \begin{cases} \vartheta_1 & \text{if } t_{dep} \leq \text{sched}(e_{dep}) + \text{max_wait}(e_{arr}, e_{dep}), \\ \vartheta_2 & \text{otherwise} \end{cases} \end{aligned} \quad (3.6.2)$$

where

$$\begin{aligned} \vartheta_1 &= P(X_{e_{arr}} + \text{min_dur}(e_{arr}, e_{dep}) \leq t_{dep}) \cdot \psi_{e_{arr}}(e_{dep}, t_{dep}) \\ &\quad - P(X_{e_{arr}} + \text{min_dur}(e_{arr}, e_{dep}) \leq t_{dep} - 1) \cdot \psi_{e_{arr}}(e_{dep}, t_{dep} - 1) \end{aligned}$$

and

$$\begin{aligned} \vartheta_2 &= P(X_{e_{arr}} + \text{min_dur}(e_{arr}, e_{dep}) \leq t_{dep}) \\ &\quad \cdot [\psi_{e_{arr}}(e_{dep}, t_{dep}) - \psi_{e_{arr}}(e_{dep}, t_{dep} - 1)]. \end{aligned}$$

The probability $\psi_{e_{arr}}(e_{dep}, t_{dep})$ is computed with the formula defined for $\psi(e_{dep}, t_{dep})$ in Eq. 3.5.2 (on page 45) but with one difference: the product in Eq. 3.5.2 is computed over all arrival events in the set $E_{arr}^{depend}(e_{dep}) \setminus \{e_{arr}\}$ instead of the set $E_{arr}^{depend}(e_{dep})$.

Assumption of independence In Sect. 3.5.2.1 (cf. *Assumption of independence*), we already introduced our assumption of independence for the computation of the functions $\phi(e)$, $e \in E$. Analogously, to compute $P(X_{e_{arr}} + \text{min_dur}(e_{arr}, e_{dep}) \leq X_{e_{dep}})$, we assume that the random event times of the arrival event e_{arr} and the arrival events in the set $E_{arr}^{depend}(e_{dep}) \setminus \{e_{arr}\}$ are stochastically independent. This assumption will be discussed in Sect. 3.11.

Explanations Obviously, Eq. 3.6.2 (on page 47) is a modification of the formula from Eq. 3.5.3 (on page 45) employed to compute $\phi(e_{dep})$ for a departure event e_{dep} . By introducing $\psi_{e_{arr}}(e_{dep}, t_{dep})$, the arrival event e_{arr} is omitted from the computation of the product in Eq. 3.5.2 (on page 45). Instead, the arrival event e_{arr} is incorporated into the formula individually. The reason is that, according to Eq. 3.6.1 (on page 46), we need to compute the probability for the case that the transfer activity (e_{arr}, e_{dep}) is feasible; for real event times of e_{arr} later than $t_{dep} - \text{min_dur}(e_{arr}, e_{dep})$, the transfer is infeasible.

Moreover, a distinction is required between the cases where e_{dep} waits for e_{arr}

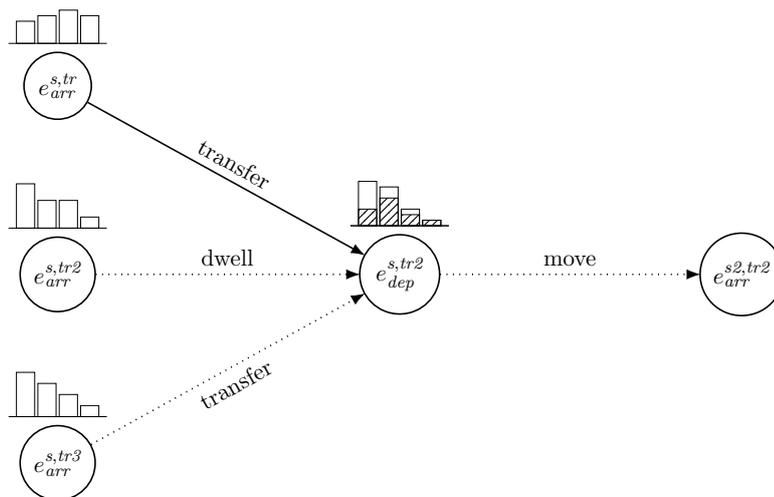


Figure 3.2: The figure illustrates a section of a train network. The reliability of the transfer activity $(e_{arr}^{s,tr}, e_{dep}^{s,tr2}) \in TRANS$ should be assessed. There is a dwell activity $(e_{arr}^{s,tr2}, e_{dep}^{s,tr2}) \in DW$. The departure event $e_{dep}^{s,tr2}$ potentially waits for $e_{arr}^{s,tr}$ and $e_{arr}^{s,tr3}$. The chart bars above the events represent the probability density functions $\phi(e)$ for each event e . In addition, for $e_{dep}^{s,tr2}$ and each $t_{dep} \in \text{supp}(\phi(e_{dep}^{s,tr2}))$, the probability $P(X_{e_{arr}^{s,tr}} + \min_dur(e_{arr}^{s,tr}, e_{dep}^{s,tr2}) \leq t_{dep} \cap X_{e_{dep}^{s,tr2}} = t_{dep})$ is illustrated as a dashed part of the respective bar.

or where not. For $t_{dep} \leq \text{sched}(e_{dep}) + \text{max_wait}(e_{arr}, e_{dep})$, a delayed arrival time of e_{arr} influences the real event time of e_{dep} . In contrast, for $t_{dep} > \text{sched}(e_{dep}) + \text{max_wait}(e_{arr}, e_{dep})$, the departure of e_{dep} at t_{dep} is independent from the arrival time of e_{arr} .

If e_{dep} does not wait for e_{arr} at all, $X_{e_{dep}}$ does not depend on $X_{e_{arr}}$. In that case, the probability from Eq. 3.6.2 (on page 47) equals

$$P(X_{e_{arr}} + \min_dur(e_{arr}, e_{dep}) \leq t_{dep}) \cdot \phi(e_{dep}, t_{dep}).$$

This also applies if (e_{arr}, e_{dep}) is a transfer activity with a walk.

Note: For each transfer activity $(e_{arr}, e_{dep}) \in TRANS$ and each time $t_{dep} \in TIMES$, observe that

$$P(X_{e_{arr}} + \min_dur(e_{arr}, e_{dep}) \leq t_{dep} \cap X_{e_{dep}} = t_{dep}) \leq \phi(e_{dep}, t_{dep}).$$

An illustrative example Fig. 3.2 demonstrates an illustrative example of the dependencies between departure and arrival events which play a role when computing the probability of the success of a transfer. In the example, the real event time of the

departure event $e_{dep}^{s,tr2}$ depends on the arrival events $e_{arr}^{s,tr}$, $e_{arr}^{s,tr2}$, and $e_{arr}^{s,tr3}$. Thus, the set $E_{arr}^{depend}(e_{dep}^{s,tr2})$ (as introduced in Sect. 3.5.1) comprises these three arrival events. The support of $\phi(e)$ for each event e in the example is limited to zero till three minutes delay. We assume that a passenger arrives with the train tr at the station s and plans to transfer into the train $tr2$. To assess the probability of the success of the transfer activity $(e_{arr}^{s,tr}, e_{dep}^{s,tr2}) \in TRANS$, the formula presented in Eq. 3.6.1 (on page 46) is applied.

3.7 The Reliability Assessment of Train Connections

In this section, we introduce the reliability assessment of train connections. To assess the reliability of a train connection, we compute its *probability of success*.

The Probability of the Success of a Train Connection This is the probability that the train connection will be *feasible* according to the real times (cf. Sect. 1.3.2). In other words, it is the probability that the train connection does not break because of delays.

As defined in Sect. 1.2, a train connection is a sequence of train and walk trips; there is a transfer activity between each pair of train trips which appear in the train connection successively. For a train connection c with $k \in \mathbb{N}_0$ transfers, let $(e_{arr}^j, e_{dep}^j) \in TRANS$, $j = 1, \dots, k$, denote the transfer activities in c . Formally, the probability of the success of the train connection c equals

$$P\left(\bigcap_{j=1,\dots,k} X_{e_{arr}^j} + min_dur(e_{arr}^j, e_{dep}^j) \leq X_{e_{dep}^j}\right). \quad (3.7.1)$$

In the following, we discuss how the above probability is calculated.

3.7.1 The Calculation

To compute the probability from Eq. 3.7.1 for a train connection c , first, we introduce a stochastic event.

The stochastic event F_i Let e_1, e_2, \dots, e_n denote the sequence of all departure and arrival events in the train connection c where $n \in \mathbb{N}_{>0}$ is the number of all events in c . For $i \in 1, 2, \dots, n$, we introduce F_i ; it denotes the stochastic event that the part of c from e_1 to e_i is feasible according to real times (cf. Sect. 1.3.2) if the events e_1, e_2, \dots, e_i take place. According to Eq. 3.7.1, we obtain

$$P\left(\bigcap_{j=1,\dots,k} X_{e_{arr}^j} + min_dur(e_{arr}^j, e_{dep}^j) \leq X_{e_{dep}^j}\right) = P(F_n)$$

where e_n is the very last event in the train connection c ; the values j , k , and (e_{arr}^j, e_{dep}^j) are defined as in Eq. 3.7.1. We now discuss how the probability $P(F_n)$ can be computed. Observe that

$$P(F_n) = \sum_{t \in \text{supp}(\phi(e_n))} P(F_n \cap X_{e_n} = t)$$

The probability $P(F_i \cap X_{e_i} = t)$ is defined recursively.

Recursion anchor The recursion anchor is at the very first event e_1 in c ; it is a departure event. For each $t \in \text{supp}(\phi(e_1))$, we define

$$P(F_1 \cap X_{e_1} = t) = \phi(e_1, t).$$

For $i \in 1, 2, \dots, n$ and each $t \notin \text{supp}(\phi(e_i))$, we define $P(F_i \cap X_{e_i} = t) = 0$. For $i > 1$ and each time $t \in \text{supp}(\phi(e_i))$, the recursive step depends on whether e_i is an arrival or a departure event.

Recursive step for arrivals If e_i is an arrival event, we define

$$\begin{aligned} P(F_i \cap X_{e_i} = t) \\ = \sum_{t_{dep} = \text{sched}(e_{i-1})}^t P(F_{i-1} \cap X_{e_{i-1}} = t_{dep}) \cdot \theta(tm, t_{dep}, t) \end{aligned} \quad (3.7.2)$$

where $tm = (e_{i-1}, e_i)$ is a train move in the train connection c .

Recursive step for departures If e_i is a departure event, we proceed as follows. Let $t_{dep}^{latest}(e_{i-1}, e_i)$ be the latest event time of e_i that can be caused by e_{i-1} as precisely defined in Eq. 3.5.1 (on page 44). For $t \in \text{supp}(\phi(e_i))$, we define

$$P(F_i \cap X_{e_i} = t) = \begin{cases} \tilde{\vartheta}_1 & \text{if } t \leq t_{dep}^{latest}(e_{i-1}, e_i), \\ \tilde{\vartheta}_2 & \text{otherwise} \end{cases} \quad (3.7.3)$$

where

$$\begin{aligned} \tilde{\vartheta}_1 &= P(F_{i-1} \cap X_{e_{i-1}} \leq t - \text{min_dur}(e_i, e_{i-1})) \cdot \psi_{e_{i-1}}(e_i, t) \\ &\quad - P(F_{i-1} \cap X_{e_{i-1}} \leq (t - \text{min_dur}(e_i, e_{i-1})) - 1) \cdot \psi_{e_{i-1}}(e_i, t - 1) \end{aligned}$$

and

$$\begin{aligned} \tilde{\vartheta}_2 &= P(F_{i-1} \cap X_{e_{i-1}} \leq t - \text{min_dur}(e_i, e_{i-1})) \\ &\quad \cdot [\psi_{e_{i-1}}(e_i, t) - \psi_{e_{i-1}}(e_i, t - 1)]. \end{aligned}$$

The values $\psi_{e_{i-1}}(e_i, t)$ and $\psi_{e_{i-1}}(e_i, t - 1)$ are computed as defined in Eq. 3.5.2 (on page 45) with the difference that the event e_{i-1} is excluded from the set $E_{arr}^{depend}(e_i)$. Finally, for each event e_i , $i = 2, \dots, n$, and each time $t \in \text{supp}(\phi(e_i))$, observe that

$$P(F_i \cap X_{e_i} \leq t) = \sum_{t'=\text{earliest}}^t (F_i \cap X_{e_i} = t') \quad (3.7.4)$$

where *earliest* is the smallest time in $\text{supp}(\phi(e_i))$.

Explanations The formula from Eq. 3.7.2 (on page 50) is based on the formula from Eq. 3.5.4 (on page 45). The difference is that the probability $\phi(e_{i-1}, t_{dep})$ —used in Eq. 3.5.4—is substituted by $P(F_{i-1} \cap X_{e_{i-1}} = t_{dep})$; instead of the probability that e_{i-1} takes place at the time t_{dep} , we require the probability that the passenger—who takes the train connection c —can depart with the train of e_{i-1} at the station of e_{i-1} at the time t_{dep} . Analogously, Eq. 3.7.3 (on page 50) is derived from Eq. 3.6.2 (on page 47).

Furthermore, We make the following observations.

- For each train move or dwell activity (e_{i-1}, e_i) in c where $i = 2, \dots, n$, one has $P(F_{i-1}) = P(F_i)$.
- For each transfer activity (e_{i-1}, e_i) in c where $i = 2, \dots, n$, one has $P(F_{i-1}) \geq P(F_i)$.
- For each time $t \in \text{TIMES}$ and $i = 1, \dots, n$, one has $P(F_i \cap X_{e_i} = t) \leq \phi(e_i, t)$.

These observations are illustrated in Fig. 3.3, by way of example.

Assumptions of independence By taking into account the above observations, the assumptions of independence from Sect. 3.5.2.1 and Sect. 3.6 (cf. *Assumptions of independence*) are sufficient for the computation of the probabilities $P(F_i \cap X_{e_i} \leq t)$ for all events e_i in c and all times $t \in \text{TIMES}$. For more details see Sect. 3.11.

3.7.2 Deadline Scenario

A passenger often has a *deadline* for the arrival at the destination station; an arrival later than the deadline would be too late. In such a scenario, the definition of the probability of the success of a train connection is as follows: the probability that the passenger reaches the destination no later than the deadline (and the train connection does not break because of delays). Let $deadline \in \text{TIMES}$ denote the deadline, e_n denote the very last arrival event of the train connection (at the destination), and the

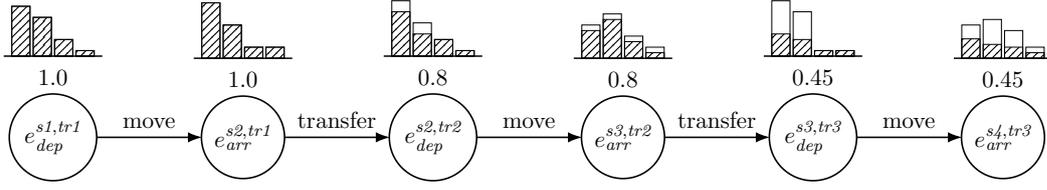


Figure 3.3: The figure illustrates a train connection with two transfers and six events. For each event e_i in the train connection, where $i = 1, \dots, 6$, the chart bars above the events represent the probability density function $\phi(e_i)$ where $\text{supp}(\phi(e_i)) = \{\text{sched}(e_i) + d \mid d = 0, \dots, 3\}$. The probabilities $P(F_i \cap X_{e_i} = t)$, $t \in \text{supp}(\phi(e_i))$, are illustrated as dashed areas within the chart bars. The probability $P(F_i)$ is listed below the chart bar. The probability of the success of the train connection equals 0.45. For the sake of convenience, we did not illustrate the existing dependencies of the departure nodes on other arrival nodes.

stochastic event F_n be defined as in Sect. 3.7.1. The probability of the success of a train connection by taking into account a deadline is defined as follows:

$$\begin{aligned}
 & P\left(\bigcap_{j=1, \dots, k} X_{e_{arr}^j} + \text{min_dur}(e_{arr}^j, e_{dep}^j) \leq X_{e_{dep}^j}\right. \\
 & \quad \left. \cap X_{e_n} \leq \text{deadline}\right) \\
 & = P(F_n \cap X_{e_n} \leq \text{deadline}) \\
 & = \sum_{\substack{t \in \text{supp}(\phi(e_n)) \\ t \leq \text{deadline}}} P(F_n \cap X_{e_n} = t).
 \end{aligned}$$

The probability $P(F_n \cap X_{e_n} = t)$ is computed as discussed in Sect. 3.7.1.

3.8 Processing Events

In daily operation as well as for diverse evaluations, timetable information systems are used to compute large sets of train connections. If additionally a reliability assessment of each train connection is desired, it has to be efficient; the assessment should not slow down the timetable information system.

As introduced in Sect. 3.6–3.7, our reliability assessment of train connections is based on probability distributions for departure and arrival events. Whenever we compute the probability of the success of a train connection c , the probability distributions of a specific set of departure and arrival events are required. More precisely, we require the probability density function $\phi(e)$ for each event e in a set of events that we denote by $E_{req}(c) \subset E$ and define as follows. The set $E_{req}(c)$ comprises the very first departure event of c according to the recursion anchor of our recursive function from Sect. 3.7. Moreover, for each departure event e_{dep} in the train connection c , the set $E_{req}(c)$ comprises each arrival event $e_{arr} \in E_{arr}^{depend}(e_{dep})$.

In Sect. 3.8.1–3.8.3, we discuss three different approaches to *processing* the events in $E_{req}(c)$; that is, to compute the functions $\phi(e)$, $e \in E_{req}(c)$, which are required for the reliability assessment of c . In Sect. 3.8.4, we briefly discuss how all these approaches can be accelerated by a parallelization. The approaches presented here will be evaluated in Sect. 9.2.1.

3.8.1 On-Demand Processing

The first approach is to compute the probability density functions of the events in the set $E_{req}(c)$ *on demand*, that is, whenever the probability of the success of the train connection c should be determined. Since the events in $E_{req}(c)$ recursively depend on further events (see Sect. 3.5.1 and 3.5.2), this approach potentially results in a cascade of events for which the probability density functions need to be computed. This is an expensive procedure that would be performed each time the reliability of a train connection is assessed. As explained at the beginning of Sect. 3.8, in daily operation, the reliability assessment is usually performed for large sets of train connections. Thus, given a large set of train connections, we would perform many redundant calculations if, for each train connection, the events in $E_{req}(c)$ are processed on demand. This approach would be very inefficient and time-wasting.

3.8.2 Pre-Processing by Dynamic Programming

As opposed to the on-demand computation, the probability density function $\phi(e)$ of each event $e \in E$ can be computed in a pre-processing step prior to the reliability assessment of any train connection. After that, for every train connection c , the probability density functions of the events in the set $E_{req}(c)$ will be available. This approach is detailed in Sect. 3.8.2.1–3.8.2.2.

3.8.2.1 The Dynamic Programming Approach

Our pre-processing approach to computing the probability density function $\phi(e)$ for each event $e \in E$ is presented in Algorithm 3.1 (on page 54). It uses a queue that is specified in Sect. 3.8.2.2.

For all events $e \in E$, we compute the probability density functions $\phi(e)$ via dynamic programming. Briefly, the idea of dynamic programming is that a complex problem is broken down into subproblems, each subproblem is solved once, and its solution is available when it is required (see [CLRS01], Chapter 15, Page 323). Translated to our problem, we compute every probability density function once, and every time we compute the $\phi(e)$ function, for an event e , the probability density functions of all events

```

Data: Timetable  $TT$ ; probability distributions describing random prepared
times and random train move durations
Result: Set  $\Phi$  of all functions  $\phi(e)$ ,  $e \in E$ 
1  $\Phi \leftarrow \emptyset$ ;
2 Create empty queue;
  /* Push into the queue the very first train move of each train */
3 foreach  $(e_{dep}, e_{arr}) \in TM$  do
4   if  $\neg[\exists e'_{arr} \in E_{arr}, (e'_{arr}, e_{dep}) \in DW]$  then
5     | queue.push $((e_{dep}, e_{arr}))$ ;
6   end
7 end
8 while queue is not empty do
9    $(e_{dep}, e_{arr}) \leftarrow \text{queue.pop}()$ ;
  /* Compute the probability density functions */
10   $\phi(e_{dep}) \leftarrow \text{process}(e_{dep})$ ;
11   $\phi(e_{arr}) \leftarrow \text{process}(e_{arr})$ ;
12   $\Phi \leftarrow \Phi \cup \{\phi(e_{dep}), \phi(e_{arr})\}$ ;
  /* Push into the queue the succeeding train move */
13  if  $\exists e'_{dep} \in E_{dep}, (e_{arr}, e'_{dep}) \in DW$  then
14    | queue.push $((e'_{dep}, e'_{arr}))$ ;
15  end
16 end

```

Algorithm 3.1: The algorithm computes the probability density function $\phi(e)$ of each event $e \in E$. The queue is as specified in Sect. 3.8.2.2.

on which e depends are available. For this computation, the following condition needs to be satisfied.

Condition 3.8.1 Process all events in E in an order such that

1. for each train move $(e_{dep}, e_{arr}) \in TM$, the event e_{dep} is processed prior to e_{arr} , and
2. no departure event $e_{dep} \in E_{dep}$ is processed unless all arrival events in $E_{arr}^{depend}(e_{dep})$ have been processed.

To satisfy Condition 3.8.1, we could introduce a topological ordering of the events (see [CLRS01], Chapter 22, Page 549), and process the events in a topological order. A topological ordering exists for the events in our timetable since they can be represented as a directed acyclic graph as presented in Sect. 2.1. However, to save the additional effort of a topological ordering, the procedure presented in Algorithm 3.1 (on page 54) fulfills Condition 3.8.1 as follows.

The first part of Condition 3.8.1 is fulfilled by Lines 10–11 in Algorithm 3.1. We

now discuss the second part. From Sect. 3.5.1, recall that, for each departure node $e_{dep}^{s,tr}$, the set $E_{arr}^{depend}(e_{dep}^{s,tr})$ comprises the arrival event $e_{arr}^{s,tr}$ (if it exists) and all arrival events for which $e_{dep}^{s,tr}$ potentially waits; the probability density function for $e_{artific}^{s,tr}$ is given and need not be considered here. The second part of Condition 3.8.1 is obeyed for $e_{arr}^{s,tr}$ since the train move starting with $e_{dep}^{s,tr}$ is added to the queue only after the train move ending with $e_{arr}^{s,tr}$ is processed. On the other hand, the fulfillment of the second part of Condition 3.8.1 for the arrival events of other trains—for which $e_{dep}^{s,tr}$ potentially waits—is left to the queue which is specified in Sect. 3.8.2.2.

3.8.2.2 The Queue

The queue from Algorithm 3.1 is a data structure to manage train moves. The operation *push* adds a new train move to the queue. The operation *pop* removes a train move from the queue and delivers it. Our queue delivers the train moves sorted by the scheduled event times of their departure events in ascending order. This ordering fulfills the second part of Condition 3.8.1 (on page 54) for each arrival event $e_{arr}^{s,tr2} \in E_{arr}^{depend}(e_{dep}^{s,tr1})$ where $tr1 \neq tr2$; this claim is based on the following realistic assumption.

Assumption 3.8.1 We assume that there is no pair of train moves of two different trains such that the departure events of them have the same scheduled event time and one of them depends on the another one. Formally, we assume:

$$\begin{aligned} \nexists (e_{dep}^{s1,tr1}, e_{arr}^{s2,tr1}), (e_{dep}^{s2,tr2}, e_{arr}^{s3,tr2}) \in TM, \\ sched(e_{dep}^{s1,tr1}) = sched(e_{arr}^{s2,tr1}) = sched(e_{dep}^{s2,tr2}) \\ \wedge e_{arr}^{s2,tr1} \in E_{arr}^{depend}(e_{dep}^{s2,tr2}) \\ \wedge tr1 \neq tr2. \end{aligned}$$

This assumption is realistic for real timetables in Germany that we evaluate in our computational study in Chapter 9. For a timetable which violates this assumption, instead of the approach presented in Algorithm 3.1, we propose to process the nodes in a topological order (see [CLRS01], Chapter 22, Page 549).

In summary, Algorithm 3.1 with the above queue specification satisfies Condition 3.8.1 (on page 54). In Sect. 3.13.4, we discuss how a queue that fulfills the specification above can be implemented.

3.8.3 Hybrid Approach

As already announced, the on-demand approach from Sect. 3.8.1 could be very inefficient according to the dependencies between the trains in the timetable; the reliability

assessment of each train connection would be very time-consuming. On the other hand, the pre-processing approach from Sect. 3.8.2 computes and stores the probability distributions for all events in the timetable. Consequently, the system requires a longer set-up time and more main memory.

We propose a hybrid approach that combines both approaches and is more efficient compared to them. It is straight-forward: in a pre-processing step as discussed in Sect. 3.8.2, we compute probability density functions *solely* for the events of each train for which at least one other train *waits* at some station. The probability density functions of the events of the other trains are computed on-demand as discussed in Sect. 3.8.1.

Compared to the pre-processing approach from Sect. 3.8.2, this hybrid approach requires less computation time and memory during the pre-processing step. On the other side, the on-demand computations are limited to manageable numbers of events. For the timetables in Germany including local public transport such as buses and streetcars, this hybrid approach is very efficient. The pre-processing step omits the events of buses and streetcars which usually do not wait for other trains, buses, or streetcars.

3.8.4 Parallelization

In his master's thesis [Glö15], Peter Glöckner developed and evaluated two approaches to computing the functions $\phi(e)$, $e \in E$ in parallel. First, for each event $e \in E$ and each pair of times $t, t' \in TIMES$ where $t \neq t'$, the probabilities $\phi(e, t)$ and $\phi(e, t')$ can be computed separately and in parallel. The reason is that the computation of the probability $\phi(e, t)$ does not depend on the computation of the probability $\phi(e, t')$.

Second, for two events $e, e' \in E$, the events e and e' can be processed in parallel if the real event time of e does not depend on e' and vice versa. The both events should also not depend on each other transitively. This approach requires a topological ordering of the events in order to ascertain whether an event depends on another.

In short, by a parallelization, the computations can be accelerated. However, the parallelization is not in the focus of this work, and we refer to [Glö15] for more details.

3.9 Incorporation of Real-time Incidents

In Sect. 1.3.1, we listed incidents disrupting the timetable and changing the network state. As soon as the real time $real(e)$ of an event $e \in E$ is known, we define

$$\phi(e, t) = \begin{cases} 1 & \text{if } t = real(e), \\ 0 & \text{otherwise.} \end{cases} \quad (3.9.1)$$

Hence, for the events with known real times, we use the definition from Eq. 3.9.1 instead of the definition from Sect. 3.5.2.

Updating precomputed probabilities In Sect. 3.8, we presented different approaches to computing the functions $\phi(e)$, $e \in E$. All functions computed in pre-processing must be *updated* according to the real-time incidents. First, as soon as the real time $real(e)$ of an event $e \in E$ is known, the function $\phi(e)$ is updated as in Eq. 3.9.1. Second, for each event e' that depends on e (cf. Sect. 3.5.1), the function $\phi(e')$ must be recomputed. Recursively, we must update the functions of the events which depend on e' , and so on.

The recursive updates would consequently result in a cascade of recalculations. In order to avoid unnecessary recalculations, we proceed as follows. If an event $e \in E$ has been updated, the events depending on e are updated recursively if and only if the update of e was *significant*. Let $\phi_{before}(e)$ and $\phi_{after}(e)$ denote the probability density functions for the random event time of e before and after the update respectively. The update of e is called *significant* if

$$\max_{t \in TIMES} \{ | \phi_{before}(e, t) - \phi_{after}(e, t) | \} > \epsilon$$

where the value ϵ is a tolerance and part of our computational setting (cf. Sect. 9.2.2).

Update procedure The procedure of updating the probabilities—as described above—is presented in Algorithm 3.2. We execute the update procedure whenever there are new real-time incidents or the real times of some events have become known (cf. Sect. 1.3.1). The input for the algorithm is the set TM_{Input} which is defined as follows. First, we add to TM_{Input} each train move $(e_{dep}, e_{arr}) \in TM$ where $real(e_{dep})$ or $real(e_{arr})$ has become known. Furthermore, for each train move $(e_{dep}, e_{arr}) \in TM$ that has been canceled because of train cancelations or reroutings (cf. Sect. 1.3.1), we add to TM_{Input} each train move $(e'_{dep}, e'_{arr}) \in TM$ where $e_{arr} \in E_{arr}^{depend}(e'_{dep})$ (cf. Sect. 3.5.1.1). The latter step must be performed before removing—from the timetable—the canceled train moves and their associated activities (cf. Sect. 1.3.1); after the set TM_{Input} is initialized, the canceled train moves are removed from the timetable; finally, Algorithm 3.2 is executed. The probability density functions for the additional trains can be computed on-demand (cf. Sect. 3.8.1).

Discussion The update of the probabilities, as described in this section, is sufficient for the reliability assessment of the train connections from Sect. 3.7. It is also sufficient for our approach to computing optimal complete connections presented in Chapter 5.

```

Data: Set  $TM_{Input}$ ; timetable  $TT$ ; all functions  $\phi(e)$ ,  $e \in E$ ; probability
distributions describing random prepared times and random train move
durations
Result: Set  $\Phi$  of all updated functions
1  $\Phi \leftarrow \emptyset$ ;
2 Create empty queue;
3 foreach  $(e_{dep}, e_{arr}) \in TM_{Input}$  do
4 |   queue.push( $(e_{dep}, e_{arr})$ );
5 end
6 while queue is not empty do
7 |    $(e_{dep}, e_{arr}) \leftarrow$  queue.pop();
8 |   /* Compute the probability density functions */
9 |    $\phi(e_{dep}) \leftarrow$  process( $e_{dep}$ );
10 |   $\phi(e_{arr}) \leftarrow$  process( $e_{arr}$ );
11 |   $\Phi \leftarrow \Phi \cup \{\phi(e_{dep}), \phi(e_{arr})\}$ ;
12 |  if update of  $e_{arr}$  was significant then
13 |  |   if  $\exists e'_{dep} \in E_{dep}, (e_{arr}, e'_{dep}) \in DW$  then
14 |  | |   queue.push( $(e'_{dep}, e_{arr})$ );
15 |  |   end
16 |  |   foreach  $(e_{arr}, e'_{dep}) \in TRANS$  do
17 |  | |   if  $max.wait(e_{arr}, e'_{dep}) > 0$  then
18 |  | | |   queue.push( $(e'_{dep}, e_{arr})$ );
19 |  | |   end
20 |  |   end
21 end

```

Algorithm 3.2: The above algorithm recalculates the probability density function $\phi(e)$ of each event $e \in TM_{Input}$ and all dependent events.

For approaches basing on Pareto Dijkstra (cf. Sect. 2.2), an update of the graph model is necessary as addressed by [FMHS08], [Sch09], and [Key09].

3.10 Asymptotic Complexity

In this section, we analyze the asymptotic complexity of the procedures of computing the functions $\phi(e)$, $e \in E$, and assessing the reliability of train connections.

In the following, let the values c_m and c_a denote the costs of a multiplication and an addition (or a subtraction) of two probabilities respectively. Moreover, we define $n_T = \max_{e \in E} |supp(\phi(e))|$ and $n_D = \max_{e \in E_{dep}} |E_{arr}^{depend}(e_{dep})|$. The value n_T is a constant number; in practice, the supports of the distributions are limited as will be evaluated in Sect. 9.2.1 (cf. *Negligible probabilities*).

Probability density functions An upper bound for the number of multiplications and additions which are necessary to compute the probability density functions of all events in the timetable, when employing the approach from Sect. 3.8.2, is

$$\frac{1}{2} \cdot |E| \cdot n_T \cdot \left[\underbrace{c_a + 2 \cdot n_D \cdot (c_a + c_m)}_{\substack{\text{for departures} \\ \text{(Eq. 3.5.3 on page 45)}}} + \underbrace{n_T \cdot c_m}_{\substack{\text{for arrivals} \\ \text{(Eq. 3.5.4 on page 45)}}} \right].$$

In the worst case, each event depends on all other events in the graph such that $n_D = |E|$; then, the asymptotic complexity would be $\mathcal{O}(|E|^2)$. However, in practice, the value n_D is a small number such that the probability density functions of all events in the timetable can be computed in $\mathcal{O}(|E|)$.

Reliability assessment for train connections For a given train connection c , let n_c denote the number of events in c . An upper bound for the number of multiplications and additions necessary to assess the reliability of c , by employing the formula from Eq. 3.7.1 (on page 49), is

$$\frac{1}{2} \cdot n_c \cdot n_T \cdot \left[\underbrace{c_a + 2 \cdot \left(\underbrace{n_T \cdot c_a}_{\substack{\text{summation from} \\ \text{Eq. 3.7.4 (on page 51)}}} + c_m + \underbrace{n_D \cdot (c_a + c_m)}_{\psi_{e_{i-1}}} \right)}_{\text{for departures}} + \underbrace{n_T \cdot c_m}_{\text{for arrivals}} \right].$$

In the worst case, if $n_D = |E|$, the asymptotic complexity would be $\mathcal{O}(n_c \cdot |E|)$. However, in practice, it equals $\mathcal{O}(n_c)$.

3.11 Analysis of the Assumptions of Independence

In this section, we discuss our assumptions of independence from Sect. 3.5.2.1, 3.6, and 3.7.1 (cf. *Assumption of independence*). For the computations of the functions $\phi(e)$, $e \in E$, and the probability of the success of a transfer activity or a train connection, we make the following assumption of independence:

- All random variables $X_{e_{arr}}$ are stochastically independent for $e_{dep} \in E_{dep}$ and $e_{arr} \in E_{arr}^{depend}(e_{dep})$; this assumption has been introduced in [BGMHO11].
- For each transfer activity $(e_{arr}, e_{dep}) \in TRANS$, the random event times of the arrival event e_{arr} and the arrival events in the set $E_{arr}^{depend}(e_{dep}) \setminus \{e_{arr}\}$ are stochastically independent.

We investigated this assumption of independence within the scope of a bachelor's thesis of Kai Schwierczek [Sch16]. A brief summary of that thesis follows; for more details, we refer to that work.

Theoretically, there could exist dependencies between the events addressed above. It is possible to construct constellations where, by the assumption of independence, a significant error is introduced into the computed probabilities. Fortunately, the computational study in [Sch16] demonstrated that, in real timetables in Germany, the error introduced by this assumption is negligible. First, compared to the number of all events in the timetable, the number of events which are affected by this assumption is negligible. Second, the error introduced into the probabilities is negligible; in other words, the deviation of the computed probabilities from the exact, correct values is negligible.

Independently, our computational study in Sect. 9.3 reveals that, for timetables in Germany, our reliability assessments are quite accurate despite this assumption of independence.

3.12 Cancelations and Reroutings

In Sect. 1.3, we discussed train cancelations and reroutings. Moreover, in Sect. 3.9, we explained how to incorporate them into the computation of the probability density functions $\phi(e)$, $e \in E$. The method of assessing the reliability of train connections from Sect. 3.7 does not consider train cancelations and reroutings. The reason is that, in real timetables, the probability of a train getting canceled is very low so that its influence on the probability of success is negligible. For the sake of completeness, in this section, we discuss how train cancelations and rerouting could be incorporated into the computation of the probability of success.

For each train $tr \in TR$, let X_{tr} denote the stochastic event that tr gets canceled. Let $TR(c) \subset TR$ denote the set of all trains in a train connection c . The probability that no train in the train connection c gets canceled equals

$$1 - P\left(\bigcup_{tr \in TR(c)} X_{tr}\right). \quad (3.12.1)$$

Let us assume that there is no stochastic dependency between the cancelations of the trains; that is, for each pair of trains $tr1$ and $tr2$, the stochastic events X_{tr1} and X_{tr2} are independent. In this case, the probability from Eq. 3.12.1 equals

$$\prod_{tr \in TR(c)} [1 - P(X_{tr})].$$

On the other hand, if a stochastic dependency exists, we need to apply the inclusion-exclusion principle (see Sect. III.7, page 207 in [FS09]) in order to compute the probability from Eq. 3.12.1. For example, for a train connection c where $TR(c) = \{tr1, tr2, tr3\}$,

we obtain

$$\begin{aligned} P(X_{tr1} \cup X_{tr2} \cup X_{tr3}) &= P(X_{tr1}) + P(X_{tr2}) + P(X_{tr3}) \\ &\quad - P(X_{tr1} \cap X_{tr2}) - P(X_{tr1} \cap X_{tr3}) - P(X_{tr2} \cap X_{tr3}) \\ &\quad + P(X_{tr1} \cap X_{tr2} \cap X_{tr3}). \end{aligned}$$

If we assume that cancelations do not stochastically depend on the random event times, the probability from Eq. 3.12.1 can be multiplied with the probability from Eq. 3.7.1 (on page 49) in order to obtain the probability of the success of a train connection.

This concludes the incorporation of cancelations into the computation of the probability of success. For train reroutings, the approach is analogous. This section was an excursion; in our computational study in Chapter 9, we do not incorporate train cancelations or reroutings into the computation of the probability of success.

3.13 Implementation

In Chapter 3, we presented a method of assessing the reliability of train connections; in Chapters 4 and 7, we will present approaches to the search for reliable train connections. For an efficient reliability assessment as well as efficient searches for reliable connection, we require

1. a fast access to the probabilities for random prepared times and random train move durations (cf. Sect. 3.3),
2. an efficient determination of the interdependencies between the events (cf. Sect. 3.5.1),
3. a fast access to the precomputed (cf. Sect. 3.8) probability density functions $\phi(e)$, $e \in E$, and
4. an efficient memory usage.

Issue 1: As discussed in Sect. 3.4, the train moves and the departure events are classified according to specific properties, and for each class there is a probability distribution associated to all elements in that class. Given a train move or a departure event, we need an efficient access to the respective class. For this purpose, we use multi-dimensional arrays as follows. For example, for random train move durations, if the train category, the departure delay, and the scheduled duration of the train moves are the classification criteria, we use a three dimensional array where the entry with the index (i, j, k) contains the probability distribution for the train category $i \in \mathbb{N}$ (the train category names can be represented as unique indices), the departure delay $j \in \mathbb{N}$, and

the scheduled train move duration $k \in \mathbb{N}$. For random preparation times the procedure is analogous.

Issues 2–4: For the latter three questions (see above), we propose two solutions. Each of them has its weaknesses and strengths; the suitable method should be chosen depending on the use case as will be discussed. The first solution is extending the graph model that represents the timetable (cf. Sect. 2.1); we discuss this solution in Sect. 3.13.1. The second solution is introducing a separate *dependency graph* as explained in Sect. 3.13.2.

In Sect. 3.13.3, we discuss dropping negligible probabilities in order to reduce the width of the probability distributions. Finally, in Sect. 3.13.4, we discuss the implementation of the queue specified in Sect. 3.8.2.2.

3.13.1 Extension of the Graph Model

As discussed in Sect. 2.1, the timetable can be represented as a time-expanded or a time-dependent graph. Each of these graph models can be extended in order to provide an access to the functions $\phi(e)$, $e \in E$, as well as to the interdependencies between the events; the access can be facilitated in constant time. This approach is designed to be combined with the approach from Sect. 3.8.2 to compute all functions $\phi(e)$, $e \in E$, in pre-processing. The memory usage by this approach is higher compared to the approach that will be discussed in Sect. 3.13.2. However, it is the best solution if a fast access to the functions $\phi(e)$, $e \in E$, is required as for the approach from Chapter 5. In the following, we explain how we extend the time-expanded graph model. Such an extension is possible for the time-dependent graph model as well but it is not in the focus of this work.

Extending the time-expanded graph model Each departure and arrival node, representing an event $e \in E$, additionally stores the function $\phi(e)$ so that this function can be accessed in constant time.

Moreover, we facilitate an efficient determination of the interdependencies between the events (cf. Sect. 3.5.1) as follows. Each arrival node v_{arr} has an incoming train edge (v_{dep}, v_{arr}) , and, therefore, it knows the departure node v_{dep} on which it depends. On the other side, each departure node $v_{dep}^{s,tr} \in V_{dep}$ knows the arrival node $v_{arr}^{s,tr}$ via the incoming dwell edge. These dependencies can be efficiently extracted from the time-expanded graph model.

On the other side, for each departure node $v_{dep}^s \in V_{dep}$, we need to determine via the change layer at the station s the set of the arrival events of other trains for which v_{dep}^s waits (cf. the set $E_{arr}^{depend}(e_{dep})$ in Sect. 3.5.1.1). In order to facilitate an access from a departure node v_{dep} to the arrival nodes for which v_{dep} waits, we add to the graph *feeder edges*. Let v_{arr}^s and v_{dep}^s represent the events $e_{arr}^s \in E_{arr}$ and $e_{dep}^s \in E_{dep}$

respectively. We create a feeder edge from v_{arr}^s to v_{dep}^s if $max_wait(e_{arr}^s, e_{dep}^s)$ is greater than zero. The feeder edges of a departure node are stored as its incoming edges so that they can be accessed from the departure node in constant time. Hence, given an event, its dependencies can be looked up in constant time. We evaluate this approach in Sect. 9.2.1.

3.13.2 Dependency Graph

We now discuss a method with a significantly lower memory usage compared to the graph extension from Sect. 3.13.1. The access to the functions $\phi(e)$, $e \in E$, and the interdependencies between the events can be performed using a hash structure or a binary search tree. This method is designed to be combined with the hybrid approach to computing probability density functions as presented in Sect. 3.8.3. We recommend this combination for very large timetables where only a small proportion of the events has complex interdependencies. According to the timetables for the German train network including local public transport, this hybrid approach is suitable. The pre-processing step ignores a large number of events of buses and streetcars which usually do not wait. We will evaluate this approach in Sect. 9.2.1. Below, we present the details.

Note: The dependency graph approach from this section can also be combined with the method of computing the probability density functions in pre-processing (cf. Sect. 3.8.2); however, in this case, it would result in a memory usage higher than the graph extension method (cf. Sect. 3.13.1), and, therefore, it would be pointless.

Definition of the dependency graph [MHS09] presented the *dependency graph* model; based on this idea, we introduce a dependency graph that stores the functions $\phi(e)$, $e \in E$, and the interdependencies between the events. It is defined as follows.

Let the set $E' \subseteq E$ comprise the following events:

- all events of each train $tr \in TR$ where there is at least another train $tr' \in TR$ which waits for tr at some station (according to the maximum waiting times from Sect. 1.3), and
- all events of each train which waits for at least one other train at some station.

In our dependency graph, for each event $e \in E'$, there is a node in the graph. The respective node for the event e stores the function $\phi(e)$.

In the dependency graph, for $e, e' \in E'$, there is an edge from the node of e to the node of e' if the random event time of e' depends on the random event time of e as discussed in Sect. 3.5.1. More precisely, for each $e_{arr} \in E'$ of the train move

$(e_{dep}, e_{arr}) \in TM$, in the dependency graph, there is an edge from the node representing e_{dep} to the node representing e_{arr} . On the other side, for each $e_{dep} \in E'$, there is an edge to the node representing e_{dep} from each node representing an event in $E_{arr}^{depend}(e)$. Each node knows its incoming and outgoing edges.

Hence, combined with the hybrid approach from Sect. 3.8.3, the dependency graph comprises only those probability density functions which cannot be computed on-demand efficiently.

Note: The graph that represents the timetable is not connected to the dependency graph; they are two individual data structures.

Access to the nodes To access the function $\phi(e)$ or the dependencies of an event $e \in E$, we need to search for the respective node in the dependency graph. If a node for e exists in the dependency graph, it can be delivered; otherwise, $\phi(e)$ and the dependencies are determined on-the-fly.

To find nodes in the dependency graph efficiently, we create a unique key for each event; it can be derived from the data about the event which is available in the timetable. For example, we use a combination of the train category, the train number, the event type (departure or arrival), the scheduled event time, and the station of the event. This key can be hashed or stored in a binary search tree. Using the key, we can find the node of an event efficiently.

3.13.3 Negligible Probabilities

In order to manage numerical issues and reduce the number of elements in the supports of the probability density functions $\phi(e)$, $e \in E$, we drop negligible probabilities as follows. For each event $e \in E$, we regard a time $t \in \text{supp}(\phi(e))$ as *negligible* if $\phi(e, t) < \epsilon$. We drop a negligible t if t is outside the range of non-negligible times; that is, we drop a negligible t unless there are non-negligible times $t' < t$ and $t'' > t$ for the same event. The parameter ϵ is part of our experimental setting in Sect. 9.2.1 (cf. *Negligible probabilities*).

3.13.4 The Queue

In this section, we discuss how the queue from Sect. 3.8.2.2 can be implemented. We propose two variants: an *implementation as a priority queue* and an *implementation as a dial queue*.

Implementation as a priority queue First, the queue can be realized as a *priority queue*; in such a queue each element has a priority, and the operation *pop* always

delivers the element with the highest priority. Translated to our problem, the priority of each element in the queue, that is a train move, equals the latest time in *TIMES* minus the scheduled event time of the departure event of the train move; this ensures that the operation *pop* delivers the train moves sorted by the scheduled event times of their departure events in ascending order. If a *heap* is used as the backbone of the priority queue, the complexity of each call of the operations *push* and *pop* is $\mathcal{O}(\log n)$ where n is the number of the elements in the queue.

Implementation as a dial queue Alternatively, the queue can be realized analogous to the Dial’s implementation of the Dijkstra’s algorithm as discussed in [AMZ09]. More precisely, the queue is an array of *buckets*; it contains a bucket for each time $t \in \textit{TIMES}$. The first and the last element in the array are the buckets for the earliest and the latest time in *TIMES* respectively. The operation *push* inserts a train move (e_{dep}, e_{arr}) into the bucket for the time $\textit{sched}(e_{dep})$. For the operation *pop*, the queue contains a pointer to the bucket with the earliest time that contains at least one element; the operation *pop* delivers an element from this bucket, removes it from the bucket, and updates the pointer if the bucket has become empty. Consequently, the operations *push* and *pop* run in constant time.

3.14 Conclusion

In this chapter, we presented our method of assessing the reliability of train connections. More precisely, we compute the probability of the success of each train connection; the probability that—when taking the train connection—passengers can reach their destination and the train connection does not break because of delays. Our method is based on stochastic predictions for the departure and arrival events of trains (functions $\phi(e)$, $e \in E$). The evaluations of our method in Sect. 9.3 will reveal that our reliability assessments are realistic and widely accurate. However, the quality of our assessments strongly depends on the quality of the input data such as the probability distributions for the random durations of the train moves (cf. Sect. 10.1).

In this chapter, we also demonstrated efficient procedures of computing the functions $\phi(e)$, $e \in E$, as well as updating them according to the current situation in the railway network. We described how to store the computed probability density functions and provide fast access to them.

In Sect. 9.3, we will see that our method can be applied to large train networks such as those in Germany. The method is fast and efficient even for large timetables that include short- and long-distance trains as well as local public transport such as buses and streetcars.

Chapter 4

On-Time Arrival Guarantee

4.1 Introduction

In Chapter 3, we introduced our approach to assessing the probability of the success of a train connection. We now address the search for journeys that are reliable regarding disruptions in the timetable. In Chapters 4–8, we address relevant problems of travelers and provide solutions for them.

In Sect. 1.2, we introduced a train connection as a sequence of train and walk trips. Transfers in train connections can break due to delays. When a transfer in a train connection breaks, the journey does not end for the passenger who has taken the train connection; he/she continues the journey via an *alternative continuation* to the destination.

Alternative continuations Given a train connection $c = trip_1, trip_2, \dots, trip_n$ (cf. Sect. 1.2), an alternative continuation is a train connection $c' = trip'_1, trip'_2, \dots, trip'_m$ that fulfills the following two conditions:

1. The train connections c and c' end at the same station, that is, the destination station of the passenger.
2. The alternative continuation c' starts at a station s that is visited by c as well. Let $e_{dep}^{s,tr2}$ be the very first departure event of c' at s . Moreover, let $e_{arr}^{s,tr}$ denote the arrival event of the train connection c at s . The activity $(e_{arr}^{s,tr}, e_{dep}^{s,tr2})$ is either a dwell activity (if $tr = tr2$) or a transfer activity (if $tr \neq tr2$). In the case that c' starts with a walk trip, its very first departure event is at a station $s2 \neq s$, and $(e_{arr}^{s,tr}, e_{dep}^{s2,tr2})$ is a transfer activity with a walk trip.

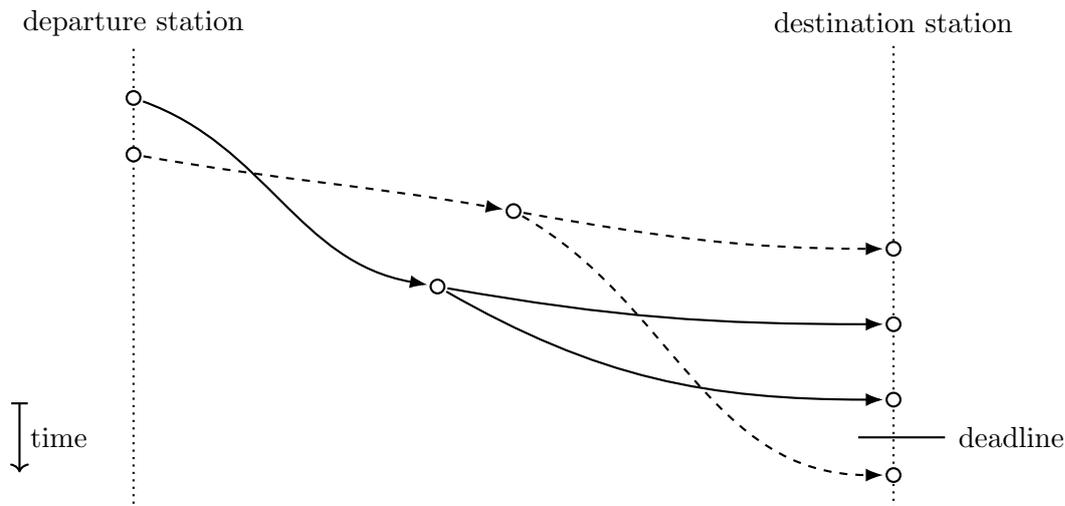


Figure 4.1: The figure illustrates the idea of computing complete connections by taking into account alternative continuations. The dashed paths illustrate a complete connection; it is a train connection with an alternative continuation where the latter arrives later than the deadline. In contrast, the solid paths illustrate another complete connection; it is a train connection with an alternative continuation where both arrive prior to the deadline. The solid train connection has a longer travel time compared to the dashed train connection; it seems to be less attractive at first appearance. However, for the solid complete connection, the probability of reaching the destination no later than the deadline is higher compared to the dashed complete connection. More details can be found in Fig. 4.2.

General problem In this chapter, we address a global problem faced by many thousands of passengers daily: arrive in time by train at the destination with a high probability even if a planned transfer breaks because of delays and an alternative continuation must be used. In addition, to save time, passengers usually desire to commence a journey as late as possible. A planned transfer from one train to another at some intermediate station breaks if it becomes impossible because the former train is delayed and the latter train does not wait for it. Breaks may even occur several times during a journey; consequently, an alternative continuation may break as well, and yet another be needed.

General idea: computing complete connections Fig. 4.1 and 4.2 illustrate the general idea. We compute attractive *complete connections* by optimizing each train connection along with its alternative continuations to the destination. A *complete connection* is a train connection (cf. Sect. 1.2) along with alternative continuations to the destination for the case of transfer breaks. The complete connection is constructed such that it has a high probability of success as required by the passenger. The destination must be reached no later than the deadline even if the passenger must change to alternative continuations. Complete connections are detailed in Sect. 4.3.2.

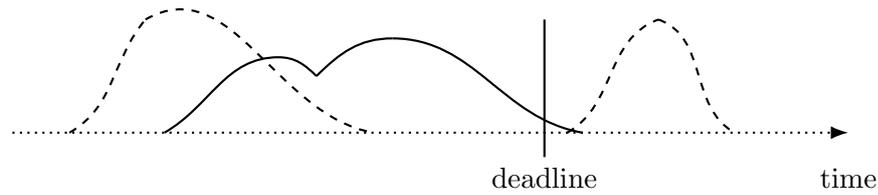


Figure 4.2: The figure, illustratively, demonstrates the probability density functions for the arrivals of the both complete connections from Fig. 4.1. More precisely, each probability density function illustrated at which time and with which probability the destination can be reached via the respective complete connection; either by the original train connection or by its alternative continuation. In contrast to the dashed complete connection, the solid one has a higher probability of arriving no later than the deadline at the destination. Note that in our model the probability density functions are discrete while the figure, illustratively, demonstrates continuous functions.

Overview First, in Sect. 4.2, we discuss the state-of-the-art. The exact problem definition can be found in Sect. 4.3. The approach employed to solve this problem is introduced in Sect. 4.4; it will be detailed in Chapters 5 and 6.

4.2 State of the Art

To the best of our knowledge, this problem has not been considered so far; there are no approaches to solving the problem of finding the latest departure time subject to a guarantee for the arrival no later than the deadline.

There are approaches focused on objectives such as least expected travel time or earliest expected arrival time as addressed in [Hal86, DPSW13, RBW16, DSW14]. The approaches presented in [Hal86, BMP⁺13, RBW16, DSW14] assume that there is no dependency between the departure and arrival times of different trains (or buses); this assumption is not realistic in a railway network where a train may be delayed because it has to wait for passengers from another delayed train. On the other hand, the works [DPSW13, DSW14] use a simplified model with the assumption that all departures are on time and all random arrival times are independent.

Approaches based on strict periodicity and designed for dense, high frequency networks are not transformable to a railway network including (but not limited to) long-distance trains which do run periodically to some extent but only roughly; an example is the approach from [Hal86].

Moreover, the approaches presented in [RBW16, DSW14] address the problem of minimizing the expected travel time subject to a guaranteed arrival at the destination prior to a deadline (called time threshold and latest arrival time in [RBW16] and [DSW14] respectively). The approaches are limited to problem instances with a required probability of success of 1; in contrast to our model, queries with arbitrary

values for the required probability of success are not supported (cf. Sect. 4.3.1). In addition, the approach presented in [RBW16] requires several minutes per query even for the very small dataset which was evaluated in that work. Thus, the approach does not scale to large train networks such as those from our computational study in Sect. 9.4.

The approach presented in [GKM⁺11, GSS⁺14] assumes a subset of all delay scenarios that can occur in the train network. A transfer is called robust if it never breaks in any delay scenario in the assumed subset. Based on this definition, two approaches are introduced: strict robustness and light robustness. The strict robustness approach delivers paths that solely contain robust transfers. In contrast, the light robustness approach delivers a path with a minimum number of non-robust transfers, subject to a maximal increase of travel time compared to the nominal solution (in those works, the nominal solution is the connection with the earliest arrival time at the destination).

In contrast to our objective, *strict* robustness results in travel times which are much longer than travel times obtained by our approach (cf. Fig. 6 in [GSS⁺14]). On the other hand, *light* robustness delivers paths with non-robust transfers for a significant number of queries. Hence, a probability of success cannot be guaranteed with the second approach—even for the seriously restricted delay scenarios that are evaluated in [GKM⁺11, GSS⁺14].

Another robustness approach has been presented in [GHM⁺13]. The response times of that approach are prohibitive in our real-world setting; that is, several minutes per query for the seriously restricted delay scenarios that are evaluated in [GHM⁺13].

Approaches for networks without a timetable (for example for road networks) such as presented in [FN06, NBK06, SBB11, SBB12, PSG13, FR98, Pre98, NPA04, NPA06, AJ14] are evidently not transferable to our problem; in contrast to a railway network, they can use each graph arc at any time, and they do not deal with the issue that transfers can break due to delays. The model presented in [Pre98, NPA04, NPA06] requires a hyper-arc for each possible value for the random variable of each node. In a timetable based network like ours, this would blow up the graph size prohibitively.

The authors of [MHS09] presented an approach to accompanying a passenger during his/her journey by providing him/her alternative continuations in case of train connection breaks. In Sect. 7.1.3 and 9.4.6, we will compare this approach to our approach from this work; we will see that this alternative approach often fails to deliver optimal solutions.

Publication of the author A manuscript has already been submitted to the journal *Transportation Science* (see [KSW17]); it is a short version of the content presented in Chapters 3–6, Sect. 7.1, and Sect. 9.4. In the mentioned chapters and sections, there

are phrases and wordings which also appear in [KSW17]. A former version of that submission had been published in [KSW14].

4.3 Problem Definition

In this section, we precisely define the problem introduced in Sect. 4.1.

4.3.1 Input and Queries

Input The input is a timetable, the function $\phi(e)$ for each event $e \in E$, the functions $\theta(tm, t_{dep})$ for each $tm \in TM$ and each $t_{dep} \in TIMES$, the functions $\xi(e_{dep})$ for each $e_{dep} \in E_{dep}$, and a *query*.

Queries A *query* q comprises

- a *departure station* where the journey starts, denoted by $dep_st(q) \in S$,
- a *destination station* where the journey ends, denoted by $dest_st(q) \in S$,
- a deadline when the passenger wants to arrive at the latest at $dest_st(q)$, denoted by $deadline(q) \in TIMES$, and
- a *required probability of success*, denoted by $\rho_{req}(q) \in [0, 1] \subset \mathbb{R}$, which might be close to 1 in real-world settings. This parameter can be chosen by the passenger or automatically set by a timetable information system.

4.3.2 Outcomes: Complete Connections

In Sect. 4.1, we introduced *complete connections*. An outcome for a query is a complete connection; that is,

- a train connection from $dep_st(q)$ to $dest_st(q)$,
- along with alternative continuations to $dest_st(q)$ for all points where the train connection may break,
- plus alternative continuations for these alternative continuations, and so on.

More precisely, for each situation that may occur during the journey, the outcome contains a precomputed *instruction* what to do next.

This definition of complete connections is sufficient to understand the rest of this chapter; the exact definition of complete connections is presented in Sect. 5.1. Complete connections have relevant properties which we introduce here and precisely define in Sect. 5.5.

Probability of the success of a complete connection For a complete connection c , computed for a query q , the value $\rho(c)$ denotes the *probability of the success* of c ; the probability that, when taking c , the passenger is able to travel from $dep_st(q)$ to $dest_st(q)$ up to $deadline(q)$. A complete connection c is called *feasible* if $\rho(c) \geq \rho_{req}(q)$; otherwise it is called *infeasible*. We discuss the computation of the probability of the success of a complete connection in Chapter 5.

The probability of the success of a complete connection can be guaranteed only at the time when it is computed. Whenever the network state has changed—due to delays or other real-time incidents (cf. Sect. 1.3)—the probability of the success of a formerly computed complete connection could be affected. We address this issue in Chapter 6.

Note: In Chapter 3, we used the terms “feasible” and “infeasible” for transfers and train connections in a different context and with a different meaning. There, we addressed the feasibility according to the timetable and the real times.

Note: In Sect. 3.7, we introduced the probability of the success of train connections. The probability of the success of a complete connection cannot be computed exactly as for a train connection since the former has a different structure and could contain multiple alternative continuations. However, the computation of the probability of the success of a complete connection is based on definitions and formulas from Chapter 3.

Scheduled departure time of a complete connection For a complete connection c , the value $dep_time(c) \in TIMES$ denotes the *scheduled departure time* of c at the departure station $dep_st(q)$ of the query q . In Sect. 5.1.1, we will specify this value more precisely.

Expected number of transfers For a complete connection c , this is the expected number of transfers which are necessary in order to travel from $dep_st(q)$ to $dest_st(q)$ when taking c . The computation is addressed in Sect. 5.3.4.

Expected arrival time For a complete connection c , this is the expected arrival time at $dest_st(q)$ when taking c . The computation is addressed in Sect. 5.4.2.

Expected travel time For a complete connection, this value is the time difference between the expected arrival time at $dest_st(q)$ and the scheduled departure time at $dep_st(q)$. The computation is addressed in Sect. 5.4.2.

4.3.3 Exact Problem Definition

In Sect. 4.3.3.1, we present the exact problem definition and specify the optimal outcome for a query q . Moreover, in Sect. 4.3.3.2–4.3.3.5, we define different variants of the problem with different objective functions. The problem from Sect. 4.3.3.1 is the main problem addressed in Chapters 4–6 and evaluated in Sect. 9.4. The other variants are introduced in order to demonstrate that our approach to computing complete connections can be adjusted in order to solve relevant related problems.

Set $C(q)$ For all problem variants, the optimal outcome is chosen from the set $C(q)$; it comprises all (not necessarily feasible) complete connections existing for a query q .

For the following definitions of the objective functions of the problem variants, we use the operators “arg max” and “arg min”; given a set of arguments (complete connections in our use case) and a function f , the operators “arg max” and “arg min” deliver the arguments that maximize and minimize the function output respectively; ties are broken arbitrarily.

4.3.3.1 Latest Departure Subject to Arrival Guarantee

The most interesting problem is to find a complete connection such that the scheduled departure time at $dep_st(q)$ is *as late as possible* subject to the guarantee that $dest_st(q)$ is reached no later than $deadline(q)$ with a probability of at least $\rho_{req}(q)$. Consequently, subject to the *required probability of success* $\rho_{req}(q)$, the departure time at $dep_st(q)$ has to be *maximized*. Formally, the *optimal* outcome o is a complete connection described by the following objective function:

$$\begin{aligned} C_{feasible}(q) &= \{ c \in C(q) \mid \rho(c) \geq \rho_{req}(q) \}, \\ o &= \arg \max \{ dep_time(c) \mid c \in C_{feasible}(q) \}. \end{aligned} \quad (4.3.1)$$

Degrees of freedom More than one element in the set $C_{feasible}(q)$ could maximize the objective function from Eq. 4.3.1. Hence, our approach for the search for optimal complete connections has degrees of freedom. In such a case, we could optimize a secondary criterion like the expected number of transfers, the expected travel time, or a combination of them. We will address this issue in Sect. 5.3.4 and 5.4.2. These tie-break rules are applicable to all problem variants from Sect. 4.3.3.

4.3.3.2 Maximum Probability of Success

Another variant of the problem from Sect. 4.3.3.1 is to find a complete connection that *maximizes* the probability of success. Among all complete connections which maximize

the probability of success, we are interested in the complete connection with the latest scheduled departure time at $dep_st(q)$. Formally, the *optimal* outcome o is a complete connection that is described by the following objective function:

$$\begin{aligned} C_{max}(q) &= \arg \max \text{set } \{ \rho(c) \mid c \in C(q) \}, \\ o &= \arg \max \{ dep_time(c) \mid c \in C_{max}(q) \}. \end{aligned} \quad (4.3.2)$$

4.3.3.3 On-Trip Scenario at a Station

We address a relevant problem that occurs in an *on-trip* scenario where the passenger has already started traveling, he/she already waits at a station at the time $ontrip_time \in TIMES$ (for example because his original train connection is broken at that station), and is looking for the complete connection with the highest probability of arriving at $dest_st(q)$ no later than $deadline(q)$.

We seek the complete connection c with the maximum probability of success $\rho(c)$ such that $dep_time(c)$ is not earlier than $ontrip_time$. The *optimal* outcome o is a complete connection that is described by the following objective function:

$$\begin{aligned} C_{ontrip}(q) &= \{ c \in C(q) \mid dep_time(c) \geq ontrip_time \}, \\ o &= \arg \max \{ \rho(c) \mid c \in C_{ontrip}(q) \}. \end{aligned} \quad (4.3.3)$$

4.3.3.4 On-Trip Scenario in a Train

We address another on-trip scenario; the passenger is in a train; more precisely, he/she is on the train move $tm \in TM$, and seeks for a complete connection in order to travel to $dest_st(q)$. Accurately, we seek the complete connection c that starts with the train move tm , and maximizes the probability of arriving at $dest_st(q)$ no later than $deadline(q)$. Let $C_{tm}(q) \subset C(q)$ denote the set of all (not necessarily feasible) complete connections that start with the train move tm . Formally, the *optimal* outcome o is a complete connection described by the following objective function:

$$o = \arg \max \{ \rho(c) \mid c \in C_{tm}(q) \}. \quad (4.3.4)$$

4.3.3.5 Minimum Expected Travel Time

Among all complete connections which guarantee a probability of success of 1, we seek the complete connection that has the minimum expected travel time and a scheduled departure time in the time interval $[earliest_dep_time, latest_dep_time]$. In contrast to the problem definitions in Sect. 4.3.3.1–4.3.3.3, the deadline plays no role; we set

$deadline(q) = +\infty$. Formally, the objective function is as follows:

$$\begin{aligned}
 C_{interval}(q) &= \{ c \in C(q) \mid dep_time(c) \geq earliest_dep_time \wedge \\
 &\quad dep_time(c) \leq latest_dep_time \}, \\
 C_{interval}^{reliable}(q) &= \{ c \in C_{interval}(q) \mid \rho(c) = 1 \}, \\
 o &= \arg \min \{ exp_travel_time(c) \mid c \in C_{interval}^{reliable}(q) \}. \tag{4.3.5}
 \end{aligned}$$

4.4 Two-Stage Approach

We propose a two-stage approach to solving the problem from Sect. 4.3, respectively, with two related software components:

1. *Booking stage*: To plan the journey in advance, the *search component* computes the *optimal* outcome (a complete connection as introduced in Sect. 4.3.2 and precisely defined in Sect. 5.1) so that the required probability of success $\rho_{req}(q)$ is met given all information available at the time of planning. For each situation that may occur while traveling, the search component computes the best *instruction* concerning what to do next.
2. *Journey*: While traveling, the passenger is guided by the *travel guidance component*. This component interprets the instructions computed by the search component. In short, whenever the passenger waits at a station, the travel guidance component suggests which train to take next. Whenever the passenger sits in a train, it suggests what to do at the very next halt: to stay in the train or to leave it.

In Chapter 5, we introduce our approach to the search component. The travel guidance component is detailed in Chapter 6.

This two-stage approach has become established in most timetable information systems. They usually consist of a back-end component that is responsible for processing the search query, and a front-end component that presents the results to the user. We correspondingly follow this two-stage approach. In addition, we generalize the second stage by introducing the travel guidance component. In addition to a comprehensive visualization of complete connections, by the travel guidance component, we address another relevant problem: as announced in Sect. 4.3.2, the probability of the success of a complete connection can be guaranteed only at the time of its computation. The travel guidance component is responsible for preserving the optimality of complete connections even after changes to the train network due to delays and other real-time incidents (cf. Sect. 1.3). This is obtained by updating the complete connections as discussed in Chapter 6.

Chapter 5

Optimal Complete Connections

As detailed in Sect. 4.4, we follow a two-stage approach in order to compute optimal outcomes to the problem from Sect. 4.3. The second stage, travel guidance, is presented in Chapter 6. The first stage is realized by the search component presented in this chapter. The input for the search component is as specified in Sect. 4.3.1 (cf. *Input*). Given a query, the search component computes the optimal outcome according to the problem definition from Sect. 4.3.2. We first introduce the *probability of success for an event* in the timetable; the search for optimal complete connections is based on the computation of this probability for events in the set E .

Probability of success $\rho(e)$ for an event Given a query q , the probability of success $\rho(e)$ for an event $e \in E$ is the probability that from e the destination station $dest_st(q)$ can be reached no later than $deadline(q)$. In Sect. 5.3.2, we explain how this probability is computed.

Overview First, in Sect. 5.1, we precisely define the outcome of the search component. In Sect. 5.2, we address the exponential size of the solution space, and abstractly explain our approach to computing and retrieving optimal complete connections. This abstract description is detailed in the subsequent sections.

In Sect. 5.3 and 5.4, we present two different recursive functions that compute the probability of success for events in the timetable as well as the best instruction to continue the journey for each situation that may occur when traveling (cf. Sect. 4.4). These recursive functions are later employed to compute optimal complete connections. The problem variants from Sect. 4.3.3.1–4.3.3.4 can be solved by the recursive function from Sect. 5.3 unless the travel time must be minimized as a secondary criterion (cf. Sect. 4.3.3.1). In the latter case, the function from Sect. 5.4 should be applied. Analogously, the problem variant from Sect. 4.3.3.5 can be solved only by the function from Sect. 5.4.

In Sect. 5.5, we precisely define relevant properties of complete connections, which have already been introduced in Sect. 4.3.2. In addition, we discuss how they are defined according to the values computed by employing the recursive functions from Sect. 5.3 and 5.4. Up to Sect. 5.6, we omit the details concerning the incorporation of walk trips into the computation of complete connections; in that section, these details are explained.

In Sect. 5.7, we translate the recursive functions from Sect. 5.3 and 5.4 into a dynamic programming approach. This approach delivers the optimal complete connection for each query. After that, in Sect. 5.8, we demonstrate a significant performance improvement of our dynamic programming approach (from Sect. 5.7); it is obtained by a restriction of the computations to events which are actually relevant for the query. We finally present in Sect. 5.9 an efficient algorithm for the computation of optimal complete connections; this algorithm is designed for the problem variant from Sect. 4.3.3.1 (cf. *Latest Departure Subject to Arrival Guarantee*).

In Sect. 5.10, we discuss the asymptotic complexity of the approaches to the search component. Then, in Sect. 5.11, we discuss heuristics which could improve the performance of the search component. In Sect. 5.12, we briefly address a parallelization of the search component. We provide details on our implementation of the search component in Sect. 5.13. Finally, this chapter is concluded in Sect. 5.14.

5.1 Exact Definition of Complete Connections

Fig. 5.1 (on page 79) illustrates a complete connection. In Sect. 4.1–4.4, we introduced a complete connection as a train connection, from $dep_st(q)$ to $dest_st(q)$ of a query q , along with alternative continuations to $dest_st(q)$. For every situation that can occur while traveling, a complete connection contains an instruction what to do next. More precisely, a complete connection starts with a train move departing from $dep_st(q)$ —as will be discussed in Sect. 5.1.1—and, from then on, it is based on arrival events. After the arrival at each station visited during the journey, by taking into account the arrival time, the complete connection contains an instruction concerning how to continue the journey towards the destination. Formally, this instruction is a departure event, and is introduced in Sect. 5.1.2.

5.1.1 Initial Departure Event

A relevant element of each complete connection is its *initial departure event*. Each complete connection, computed for a query q , starts with a specific departure event at $dep_st(q)$. The other way round, for each departure event $e_{dep}^{init} \in E_{dep}^{dep_st(q)}$, a set of complete connections can be computed which start with e_{dep}^{init} unless there is no

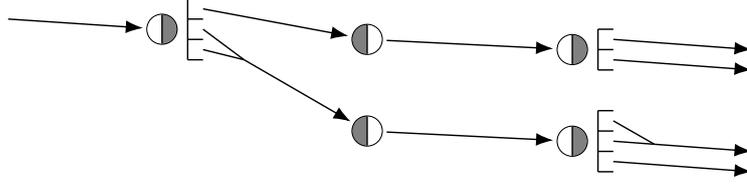


Figure 5.1: The figure illustrates a section of a complete connection schematically. The shapes \ominus and \oplus illustrate departure and arrival events respectively. For each arrival event and each real event time expected, a bucket is illustrated. For each bucket, a certain departure event is suggested for the continuation of the journey.

possibility in the timetable to start a complete connection with e_{dep}^{init} and, then, to travel to $dest_st(q)$; in the latter case, the probability of success $\rho(e_{dep}^{init})$ for the event e_{dep}^{init} equals zero. The scheduled departure time of a complete connection is the scheduled event time of its initial departure event.

5.1.2 The Instructions to Continue Traveling

Departure event $\hat{e}_{dep}^{cont}(e_{arr}, t)$ As discussed, each complete connection starts with a train move $(e_{dep}^{init}, e_{arr}^{s,tr}) \in TM$ where e_{dep}^{init} is the initial departure event (cf. Sect. 5.1.1). For each time $t \in \text{supp}(\phi(e_{arr}^{s,tr}))$ —in the support of the function describing the random time of $e_{arr}^{s,tr}$ (cf. Sect. 3.3.2), the complete connection contains a departure event denoted by $\hat{e}_{dep}^{cont}(e_{arr}^{s,tr}, t) \in E_{dep}$. The passenger is suggested to continue the journey via the departure event $\hat{e}_{dep}^{cont}(e_{arr}^{s,tr}, t)$; this is the instruction what to do next after the arrival at s with the train tr at the time t .

Hence, for $e_{arr}^{s,tr}$ and each time $t \in \text{supp}(\phi(e_{arr}^{s,tr}))$, there is a train move starting with $\hat{e}_{dep}^{cont}(e_{arr}^{s,tr}, t)$ and ending with an arrival event $e_{arr} \in E_{arr}$. Recursively, for e_{arr} and each time $t' \in \text{supp}(\phi(e_{arr}))$, the complete connection contains a departure event $\hat{e}_{dep}^{cont}(e_{arr}, t') \in E_{dep}$. This results in a *recursive* definition of complete connections as illustrated in Fig. 5.1. The recursion anchor is at $dest_st(q)$ since the destination is reached and no further instruction is required for arrival events at $dest_st(q)$.

In the above definition, for each arrival event $e_{arr}^{s,tr} \in E_{arr}$ and each time $t \in \text{supp}(\phi(e_{arr}^{s,tr}))$, the departure event $\hat{e}_{dep}^{cont}(e_{arr}^{s,tr}, t)$ may belong to tr ; in this case, the passenger simply stays in tr at s . Otherwise, the passenger leaves tr at s and enters the train of the event $\hat{e}_{dep}^{cont}(e_{arr}^{s,tr}, t)$. In the latter case, the following condition must be fulfilled:

Condition 5.1.1 There are two prerequisites for a departure event $e_{dep}^{s2,tr2} \in E_{dep}$ to be selected for $\hat{e}_{dep}^{cont}(e_{arr}^{s,tr}, t)$:

1. The activity $(e_{arr}^{s,tr}, e_{dep}^{s2,tr2})$ needs to be a transfer activity in $TRANS$.

2. The following constraint needs to be fulfilled:

$$t + \min_dur(e_{arr}^{s,tr}, e_{dep}^{s2,tr2}) \leq sched(e_{dep}^{s2,tr2}) + \max_wait(e_{arr}^{s,tr}, e_{dep}^{s2,tr2}).$$

In words, Condition 5.1.1 ensures that the passenger will be able to transfer from tr to $tr2$ if the former arrives at s at the time t . Depending on whether s and $s2$ are the same station or not, the activity $(e_{arr}^{s,tr}, e_{dep}^{s2,tr2})$ is a transfer without or with a walk trip respectively.

Case of failure In a complete connection, for some arrival event $e_{arr}^{s,tr}$ and some time $t \in \text{supp}(\phi(e_{arr}^{s,tr}))$, the value $\hat{e}_{dep}^{cont}(e_{arr}^{s,tr}, t)$ can be *void*. The complete connection consequently does not provide any instruction to continue traveling after the arrival at s with the train tr at the time t . The journey could get stuck at the station s . The probability of the success of such a complete connection is consequently lower than 1.

5.2 The Search Approach

In Sect. 5.1, we exactly defined the structure of the outcome of our search component; a complete connection. In this section, we address the problem of finding the optimal complete connection for each query. We first demonstrate that the solution space is exponentially large but, fortunately, can be reduced to a manageable size without loss of optimality. We then discuss how all complete connections in the reduced solution space can be retrieved. From those complete connections, the search component finally delivers the optimal one.

5.2.1 The Solution Space

Exponential size In Sect. 4.3.3, we introduced the set $C(q)$ of all (not necessarily feasible) complete connections existing for a query q ; this set builds the *search space*. For the problem variant from Sect. 4.3.3.1 (cf. *Latest Departure Subject to Arrival Guarantee*), the feasibility is required; that is, satisfying the required probability of success $\rho_{req}(q)$. The set of all *feasible* complete connections is called the *solution space*; it is a subset of the search space. Both of these sets can be exponentially large; an explanation follows.

Given an arrival event $e_{arr} \in E_{arr}$, let $n \in \mathbb{N}_0$ denote the number of the departure events which can be selected for $\hat{e}_{dep}^{cont}(e_{arr}, t)$ for some time $t \in \text{supp}(\phi(e_{arr}))$. In addition, we define $k = |\text{supp}(\phi(e_{arr}))|$. There are up to n^k different possibilities to set the values $\hat{e}_{dep}^{cont}(e_{arr}, t)$ for all $t \in \text{supp}(\phi(e_{arr}))$. Consequently, the search space—the set $C(q)$ —can be exponentially large in the number of the events in E and the

cardinalities of the supports of the functions $\phi(e)$, $e \in E$. Moreover, according to Eq. 4.3.1 (on page 73), for a query q with $deadline(q) = +\infty$, the solution space equals the search space, and could also be exponentially large.

Reducing the solution space The objective functions in our problem definition from Sect. 4.3.3 are based on the set $C(q)$. In order to find the optimal complete connection for a query efficiently, we need to reduce the solution space to a manageable size. More precisely, we find a subset of $C(q)$ which has a manageable size and still contains the optimal complete connection. This subset is denoted by $C_{best}(q) \subseteq C(q)$, and is defined as follows.

Set $C_{best}(q)$ First, for each departure event $e_{dep}^{init} \in E_{dep}^{dep-st(q)}$, there is at most one complete connection in the set $C_{best}(q) \subseteq C(q)$ with e_{dep}^{init} as the initial departure event (cf. Sect. 5.1.1); there is no complete connection for e_{dep}^{init} if $\rho(e_{dep}^{init})$ equals zero. Thus, the cardinality of the set $C_{best}(q)$ is lower than or equal to $|E_{dep}^{dep-st(q)}|$.

Second, each complete connection $c \in C_{best}(q)$ with the initial departure event $e_{dep}^{init} \in E_{dep}^{dep-st(q)}$ is constructed such that the condition

$$\rho(e_{dep}^{init}) = \rho(c) \geq \rho(c')$$

is satisfied for c and each complete connections $c' \notin C_{best}(q)$ with the initial departure event e_{dep}^{init} . In words, c is the complete connection by which the maximum probability of success can be obtained given that the passenger starts the journey with e_{dep}^{init} . According to the problem definitions from Sect. 4.3, this definition ensures that $C_{best}(q)$ contains the optimal complete connection for the query q —in addition to suboptimal ones.

Probability of the success of each complete connection in $C_{best}(q)$ Recall the definition of probability of success for complete connections and events, respectively, in Sect. 4.3.2 and at the beginning of Chapter 5. According to the definition of $C_{best}(q)$, for each complete connection $c \in C_{best}(q)$, the probability of success $\rho(c)$ equals the probability of success $\rho(e_{dep}^{init})$ of the initial departure event e_{dep}^{init} of c .

5.2.2 Computing the Optimal Complete Connection

Computing the complete connections in $C_{best}(q)$ The complete connections in the set $C_{best}(q)$ are computed as follows. In brief, for each situation that may occur while traveling, the search component computes the *best* instruction concerning how to continue the journey. More precisely, for each arrival event $e_{arr} \in E_{arr}$ and each time $t \in \text{supp}(\phi(e_{arr}))$, we select a departure event for $\hat{e}_{dep}^{cont}(e_{arr}, t)$ (cf. Sect. 5.1.2) such that

the following probability is *maximized*: the probability of reaching $dest_st(q)$ no later than $deadline(q)$ given that

- e_{arr} takes place at the time t and,
- from e_{arr} , the journey is continued via $\hat{e}_{dep}^{cont}(e_{arr}, t)$.

If there is no departure event that can be selected for $\hat{e}_{dep}^{cont}(e_{arr}, t)$, this value is set to *void*; we detail this case below (cf. *Case of failure*). Hence, the complete connections in the set $C_{best}(q)$ can be retrieved once the best values for $\hat{e}_{dep}^{cont}(e_{arr}, t)$ are computed for each $e_{arr} \in E_{arr}$ and each time $t \in supp(\phi(e))$.

This brief description of our approach to computing the optimal complete connection will be detailed in the rest of this chapter. In Sect. 5.3 and 5.4, we present two different recursive functions that compute the values $\hat{e}_{dep}^{cont}(e_{arr}, t)$ as described above. Moreover, for each departure event $e_{dep}^{init} \in E_{dep}^{dep_st(q)}$, the probability of success $\rho(e_{dep}^{init})$ can be computed by those recursive functions.

Note: In Sect. 5.8, we will see that the search component need not compute the value $\hat{e}_{dep}^{cont}(e_{arr}, t)$ actually for each arrival event $e_{arr} \in E_{arr}$. Without loss of optimality, the computations can be restricted to a subset of all events which we call “relevant events”.

The optimal complete connection Once the complete connections in $C_{best}(q)$ are retrieved, among them, the optimal complete connection is selected and delivered as the outcome of the search component. It is the complete connection from the set $C_{best}(q)$ that optimizes the respective objective function from Eq. 4.3.1–4.3.5 (cf. Sect. 4.3.3) depending on the problem variant. Hence, in each of the objective functions, we can safely replace $C(q)$ by $C_{best}(q)$ as already discussed in Sect. 5.2.1. The outcome of the search component will be *void* if the set $C_{feasible}(q)$, $C_{max}(q)$, $C_{ontrip}(q)$, $C_{tm}(q)$, and $C_{interval}^{reliable}(q)$ from Eq. 4.3.1–4.3.5, respectively, is empty.

The objective functions from Eq. 4.3.1–4.3.5 require two properties of each complete connection c : its scheduled departure time $dep_time(c)$ and its probability of success $\rho(c)$. Let $e_{dep}^{init} \in E_{dep}^{dep_st(q)}$ denote the initial departure event of c . We obtain $dep_time(c) = sched(e_{dep}^{init})$ and $\rho(c) = \rho(e_{dep}^{init})$.

Case of failure As addressed in Sect. 5.1.2 (cf. *Case of failure*), for an arrival event $e_{arr} \in E_{arr}^s$ and a time $t \in supp(\phi(e_{arr}))$, whenever no departure event can be found for $\hat{e}_{dep}^{cont}(e_{arr}, t)$, the journey may get stuck at s . The value $\hat{e}_{dep}^{cont}(e_{arr}, t)$ is *void*; it indicates that an arrival via e_{arr} at the time t is too late, and there is no chance to reach $dest_st(q)$ no later than $deadline(q)$. Consequently, the probability of success $\rho(c)$ of each complete connection c that contains e_{arr} is less than 1.

If the value $\hat{e}_{dep}^{cont}(e_{arr}, t)$ is *void*, and the passenger indeed arrives via e_{arr} at the time t , our search component does not provide any instruction to continue the journey. In such a case, the passenger is suggested to start a new query q with a value for $deadline(q)$ that is later than the original deadline since the latter cannot be met anymore. In this situation, the problem variant from Sect. 4.3.3.3 (cf. *On-Trip Scenario at a Station*) is in demand.

5.3 Computation of Instructions and Probabilities

In Sect. 5.2.2, we provided an overview of our approach to computing the optimal complete connection. Recall that, for each arrival event $e_{arr} \in E_{arr}$ and each time $t \in \text{supp}(\phi(e_{arr}))$, we compute the best instruction concerning how to continue the journey. More precisely, for $\hat{e}_{dep}^{cont}(e_{arr}, t)$, we find a departure event such that the following probability is *maximized*: the probability of reaching $dest_st(q)$ no later than $deadline(q)$ given that

- e_{arr} takes place at the time t and,
- from e_{arr} , the journey is continued via $\hat{e}_{dep}^{cont}(e_{arr}, t)$.

In this section, we present a recursive function that computes the best departure event for $\hat{e}_{dep}^{cont}(e_{arr}, t)$ as described above. Moreover, the probability of success $\rho(e)$ for each $e \in E$ can be computed by our recursive function. First, in Sect. 5.3.1, we introduce the *set of successor events* of each event; from that set we always select the best event to continue the journey. We then present our recursive function in Sect. 5.3.2. The selection of the best departure event to continue the journey follows in Sect. 5.3.3. From Sect. 4.3.3.1, recall that our search component has degrees of freedom so that secondary criteria can be optimized; we will introduce an optimization of the expected number of transfers in Sect. 5.3.4.

Note: The recursive function from this section optimizes the number of transfers as a secondary criterion. In Sect. 5.4, we will present another recursive function which additionally facilitates an optimization of the travel time as a secondary criterion. In fact, that recursive function is an extension of the recursive function from this section.

Note: In Sect. 5.7, we will translate the recursive function from this section into a dynamic programming approach by which we compute the complete connections in the set $C_{best}(q)$ (cf. Sect. 5.2).

5.3.1 Set of Successor Events

Set $E_{succ}(\mathbf{e}, \mathbf{t})$ For each event $e \in E$ and each time $t \in \text{supp}(\phi(e))$, the *set of successor events* comprises each event $e' \in E$ where, from e , the journey can be immediately continued via e' given that e takes place at the time t ; we denote this set by $E_{succ}(e, t)$. More precisely, for each departure event $e_{dep}^{s,tr} \in E_{dep}$, the set $E_{succ}(e_{dep}^{s,tr}, t)$ trivially contains the arrival event $e_{arr}^{s2,tr}$ of the train move $(e_{dep}^{s,tr}, e_{arr}^{s2,tr}) \in TM$. On the other hand, for each arrival event $e_{arr}^{s,tr} \in E_{arr}$, the set $E_{succ}(e_{arr}^{s,tr}, t) \subseteq E_{dep}$ comprises

- the departure event $e_{dep}^{s,tr} \in E_{dep}^s$ of the dwell activity $(e_{arr}^{s,tr}, e_{dep}^{s,tr}) \in DW$ —unless $e_{arr}^{s,tr}$ is the very last arrival event of tr —as well as
- each departure event in the set

$$\left\{ e_{dep}^{s2,tr2} \in E_{dep} \mid (e_{arr}^{s,tr}, e_{dep}^{s2,tr2}) \in TRANS \wedge \right. \\ \left. t + \text{min_dur}(e_{arr}^{s,tr}, e_{dep}^{s2,tr2}) \leq \text{sched}(e_{dep}^{s2,tr2}) + \text{max_wait}(e_{arr}^{s,tr}, e_{dep}^{s2,tr2}) \right\};$$

the latter set fulfills Condition 5.1.1 (on page 80).

For each event at the destination of the query, the set of successor events is empty.

Note: The departure event $\hat{e}_{dep}^{cont}(e_{arr}, t)$ is chosen from the set $E_{succ}(e, t)$ of the successor events of e_{arr} as will be detailed in Sect. 5.3.3.

Note: In Sect. 5.8, we will see that the set of successor events can be limited to a certain subset which solely comprises events that we call “relevant events”.

5.3.2 The Recursive Function

Computing the probability of success for an event At the beginning of Chapter 5, we introduced the probability of success $\rho(e)$ for an event $e \in E$; for a query q , it is the probability that from e the destination station $\text{dest_st}(q)$ can be reached no later than $\text{deadline}(q)$. This probability equals

$$\rho(e) = \sum_{t \in \text{supp}(\phi(e))} \phi(e, t) \cdot \varrho(e, t) \quad (5.3.1)$$

where $\phi(e, t) = P(X_e = t)$ as defined in Sect. 3.3.2; the probability $\varrho(e, t)$ is as follows.

Probability $\varrho(\mathbf{e}, \mathbf{t})$ For each event $e \in E$ and each time $t \in \text{supp}(\phi(e))$, the value $\varrho(e, t)$ is the probability that from e the destination station $\text{dest_st}(q)$ can be

reached no later than $deadline(q)$ given that the event e takes place at the time t . This value is described by a recursive formula as follows.

Recursion anchor For each arrival event $e_{arr} \in E_{arr}^{dest-st(q)}$ at the destination, the probability $\varrho(e_{arr}, t)$ equals 1 if $t \leq deadline(q)$, and 0 otherwise. Moreover, for each event $e \in E$ and each time $t > deadline(q)$, the probability $\varrho(e, t)$ equals 0. Finally, for each event $e \in E$, the probability $\varrho(e, t)$ equals 0 for all $t \notin supp(\phi(e))$ or if the set $E_{succ}(e, t)$ is empty.

Recursive step For each $e \in E \setminus E^{dest-st(q)}$ and each $t \in supp(\phi(e))$, the probability $\varrho(e, t)$ is determined as follows:

$$\varrho(e, t) = \max_{\tilde{e} \in E_{succ}(e, t)} \sum_{\tilde{t} \in supp(\phi(\tilde{e}))} P(X_{\tilde{e}} = \tilde{t} \mid X_e = t) \cdot \varrho(\tilde{e}, \tilde{t}). \quad (5.3.2)$$

If \tilde{e} is an arrival event, the probability $P(X_{\tilde{e}} = \tilde{t} \mid X_e = t)$ equals $\theta((e, \tilde{e}), t, \tilde{t})$ as defined in Sect. 3.3.2 (cf. *Random durations of train moves*). On the other hand, if \tilde{e} is a departure event, in order to compute the probability $P(X_{\tilde{e}} = \tilde{t} \mid X_e = t)$, we compute $P(X_{\tilde{e}} = \tilde{t})$ under the assumption that $P(X_e = t) = 1$ by employing the formula from Eq. 3.5.3 (on page 45).

5.3.3 Selection of the Best Event for Continuation

From Sect. 5.2.2, recall that if the arrival event e takes place at the time t , the search component suggests to continue the journey from e via the departure event $\hat{e}_{dep}^{cont}(e, t)$. The latter is set to the successor event $\tilde{e} \in E_{succ}(e, t)$ that maximizes the probability $\varrho(e, t)$ (cf. the maximization in Eq. 5.3.2). If the set $E_{succ}(e, t)$ is empty, the value $\hat{e}_{dep}^{cont}(e, t)$ is set to *void*.

5.3.4 Tie-Breaker: Optimizing the Number of Transfers

From Eq. 5.3.2 (on page 85), recall the maximization over the set $E_{succ}(e, t)$ of successor events. Observe that more than one element of that set may maximize the probability $\varrho(e, t)$. Thus, as announced in Sect. 4.3.3.1, our recursive function has degrees of freedom which allow us to optimize certain secondary criteria. In our computational study in Sect. 9.4.9, we minimize the expected number of transfers as the secondary criterion as follows.

Expected number of transfers For each event $e \in E$ and each time $t \in supp(\phi(e))$, we introduce a value $\mu(e) \in \mathbb{R}$. This value is the expected number of transfers which

are required to reach $dest_st(q)$ from the event e ; it is computed under the condition that $dest_st(q)$ is reached no later than $deadline(q)$ when following the instructions in the outcome of the search component.

The value $\mu(e)$ is computed recursively. For each departure event $e_{dep} \in E_{dep}$ of the train move (e_{dep}, e_{arr}) , we define $\mu(e_{dep}) = \mu(e_{arr})$. On the other hand, for each arrival event $e_{arr} \in E_{arr}^{dest_st(q)}$, we define $\mu(e_{arr}) = 0$. For all other arrival events $e_{arr} \in E_{arr}$, we define

$$\mu(e_{arr}) = \frac{\sum_t \phi(e_{arr}, t) \cdot [\lambda(e_{arr}, \hat{e}_{dep}^{cont}(e_{arr}, t)) + \mu(\hat{e}_{dep}^{cont}(e_{arr}, t))]}{\sum_{\tilde{t}} \phi(e_{arr}, \tilde{t})} \quad (5.3.3)$$

where the summations are over all times $t, \tilde{t} \in supp(\phi(e_{arr}))$ where $\rho(e_{arr}, t) > 0$ and $\rho(e_{arr}, \tilde{t}) > 0$ respectively; the value $\lambda(e_{arr}, \hat{e}_{dep}^{cont}(e_{arr}, t))$ equals

$$\lambda(e_{arr}, \hat{e}_{dep}^{cont}(e_{arr}, t)) = \begin{cases} 1 & \text{if } (e_{arr}, \hat{e}_{dep}^{cont}(e_{arr}, t)) \in TRANS, \\ 0 & \text{otherwise.} \end{cases}$$

Definition of the tie-breaker The tie-breaker that we use for the selection of $\hat{e}_{dep}^{cont}(e_{arr}, t)$ —in Sect. 5.3.3—is as follows. Among all departure events in $E_{succ}(e_{arr}, t)$ that maximize the probability $\rho(e_{arr}, t)$ (cf. Eq. 5.3.2), for $\hat{e}_{dep}^{cont}(e_{arr}, t)$, we select the departure event that minimizes

$$\lambda(e_{arr}, \hat{e}_{dep}^{cont}(e_{arr}, t)) + \mu(\hat{e}_{dep}^{cont}(e_{arr}, t)).$$

Tolerance Again, from Eq. 5.3.2 (on page 85), recall the maximization over the set $E_{succ}(e, t)$ of successor events. For that maximization, for each successor event in $E_{succ}(e, t)$, we compute the probability defined in Eq. 5.3.2; subsequently, these probabilities are compared to each other in order to find the maximum. For these comparisons, a tolerance can be applied. More precisely, two successors are assumed to be equivalent if the difference between the values for $\rho(e_{arr}, t)$, obtained by them, is smaller than a given threshold. Applying such a tolerance rule can increase the number of successors that maximize $\rho(e_{arr}, t)$. Consequently, there is a higher chance to find a successor with a smaller value for the expected number of transfers. However, such a tolerance rule could result in suboptimal outcomes. We will evaluate the effect of our tie-breaker and the tolerance rule in Sect. 9.4.9.

5.4 Computation Extended by Travel Time Optimization

In Sect. 5.3, we presented a recursive function that computes the values $\varrho(e, t)$ and $\hat{e}_{dep}^{cont}(e, t)$ for an event $e \in E$ and a time $t \in \text{supp}(\phi(e))$. There, we optimized the expected number of transfers as a secondary criterion. In this section, we extend the recursive formula from Sect. 5.3 in order to additionally facilitate a travel time optimization. For the problem variants from Sect. 4.3.3.1–4.3.3.4, this optimization can be incorporated as a tie-breaker into the computation of the optimal complete connection— analogously to Sect. 5.3.4. For the problem variant from Sect. 4.3.3.5 (cf. *Minimum Expected Travel Time*), this optimization is even the primary criterion for the computation of the optimal complete connection.

5.4.1 Extension of the Recursive Function

Given a query q , in addition to the probability $\varrho(e, t)$ from Sect. 5.3, we separately look at every arrival time at the destination. For this, we introduce the probability $\varrho(e, t, t_{arr})$.

Probability $\varrho(e, t, t_{arr})$ For an event $e \in E$, a time $t \in \text{supp}(\phi(e))$, and a time $t_{arr} \in \text{TIMES}$ where $t \leq t_{arr} \leq \text{deadline}(q)$, the value $\varrho(e, t, t_{arr})$ is the probability that from e the destination station $\text{dest_st}(q)$ can be reached exactly at the time t_{arr} given that e takes place at the time t . According to the definition of $\varrho(e, t)$ from Sect. 5.3, we obtain

$$\varrho(e, t) = \sum_{t_{arr} \leq \text{deadline}(q)} \varrho(e, t, t_{arr}).$$

Let the set $E_{succ}(e, t)$ of successor events be defined as in Sect. 5.3.1. The probability $\varrho(e, t, t_{arr})$ is described by the following recursive formula.

Recursion anchor For each arrival event $e_{arr} \in E_{arr}^{\text{dest_st}(q)}$ at the destination, the probability $\varrho(e_{arr}, t, t_{arr})$ equals 1 if $t \leq \text{deadline}(q)$ and $t_{arr} = t$, and 0 otherwise. Moreover, for each event $e \in E$ and times $t, t_{arr} \in \text{TIMES}$, the probability $\varrho(e, t, t_{arr})$ equals 0 once one of the following conditions is true: $t > \text{deadline}(q)$, $t_{arr} > \text{deadline}(q)$, $t > t_{arr}$, $t \notin \text{supp}(\phi(e))$, or $E_{succ}(e, t)$ is empty.

For each arrival event $e_{arr} \in E_{arr}^{\text{dest_st}(q)}$ at the destination and each $t \in \text{TIMES}$, the value $\hat{e}_{dep}^{cont}(e_{arr}, t)$ is set to *void*. In addition, for each $e \in E$ and $t \in \text{TIMES}$ where $\varrho(e, t)$ equals 0, the value $\hat{e}_{dep}^{cont}(e, t)$ is set to *void*.

Recursive step For each event $e \in E \setminus E^{\text{dest_st}(q)}$, each $t \in \text{supp}(\phi(e))$, and each $t_{arr} \in \text{TIMES}$ where $t \leq t_{arr} \leq \text{deadline}(q)$, the probability $\varrho(e, t, t_{arr})$ is defined as

follows. First, we introduce an additional value $\varrho(e, t, \tilde{e}, t_{arr})$; it is the probability that from e the destination station $dest_st(q)$ can be reached exactly at the time t_{arr} given that

- the event e takes place at the time t and,
- from e , the journey is continued via $\tilde{e} \in E_{succ}(e, t)$.

Formally, for $\tilde{e} \in E_{succ}(e, t)$, the probability $\varrho(e, t, \tilde{e}, t_{arr})$ equals

$$\varrho(e, t, \tilde{e}, t_{arr}) = \sum_{\tilde{t} \in \text{supp}(\phi(\tilde{e}))} P(X_{\tilde{e}} = \tilde{t} \mid X_e = t) \cdot \varrho(\tilde{e}, \tilde{t}, t_{arr}). \quad (5.4.1)$$

The term $P(X_{\tilde{e}} = \tilde{t} \mid X_e = t)$ is defined as in Eq. 5.3.2 (on page 85).

Subsequently, if e is an arrival event, we select the departure event $\hat{e}_{dep}^{cont}(e, t)$ as in Sect. 5.3.3:

$$\hat{e}_{dep}^{cont}(e, t) = \arg \max \left\{ \tilde{e} \in E_{succ}(e, t) \mid \sum_{t_{arr} \in \text{TIMES}} \varrho(e, t, \tilde{e}, t_{arr}) \right\}. \quad (5.4.2)$$

If the set $E_{succ}(e, t)$ is empty, the value $\hat{e}_{dep}^{cont}(e, t)$ is set to *void*.

Finally, if e is an arrival event, we define

$$\varrho(e, t, t_{arr}) = \varrho(e, t, \hat{e}_{dep}^{cont}(e, t), t_{arr}). \quad (5.4.3)$$

On the other hand, if e is a departure event, we define

$$\varrho(e, t, t_{arr}) = \varrho(e, t, \tilde{e}, t_{arr})$$

where \tilde{e} is the arrival event of the train move (e, \tilde{e}) starting with the departure event e .

This concludes the extended version of the recursive formula that computes the values $\varrho(e, t)$ and $\hat{e}_{dep}^{cont}(e, t)$. While both of these values are defined as in Sect. 5.3, the value $\varrho(e, t, t_{arr})$ allows us to optimize the travel time as a secondary criterion; this optimization is detailed in Sect. 5.4.2.

5.4.2 Travel Time Optimization

Set $E_{succ}^{max}(e, t)$ For each event $e \in E$ and each time $t \in \text{supp}(\phi(e))$, we define the set $E_{succ}^{max}(e, t) \subseteq E_{succ}(e, t)$ of all successor events that maximize the probability from Eq. 5.4.2 (on page 88). For the problems from Sect. 4.3.3.1–4.3.3.3, more than one element of the set $E_{succ}(e, t)$ may maximize the probability from Eq. 5.4.2. On the

other hand, for the problem from Sect. 4.3.3.5, we have $E_{succ}^{max}(e, t) = E_{succ}(e, t)$ since $deadline(q)$ equals $+\infty$.

Note: The latter statement can be wrong if t is a timestamp on the last day of the timetable such that the destination can not be reached since the timetable does not contain trains which can be used at a time equal to or later than t . For the sake of simplicity, we assume that this case never happens. Practically, a timetable information system could always load a timetable with a sufficient time period in the future, and allow queries with values for $deadline(q)$ such that the specified problem does not occur.

Since the set $E_{succ}^{max}(e, t)$ potentially contains more than one element, our recursive function has degrees of freedom which allow us to optimize certain secondary criteria. In Sect. 5.3.4, we presented a tie-breaker based on the expected number of transfers; in this section, we present another tie-breaker by which we optimize the travel time of complete connections based on the value $\varrho(e, t, \tilde{e}, t_{arr})$ from Sect. 5.4.1.

Note: We can use a tolerance as introduced in Sect. 5.3.4 in order to build the set $E_{succ}^{max}(e, t)$.

Minimization of the travel time Our tie-breaker selects from the set $E_{succ}^{max}(e_{arr}, t)$ a successor such that the travel time from e_{arr} to $dest_st(q)$ is minimized. There, the travel time is the time difference between the time t and the expected arrival time at $dest_st(q)$. While the expected arrival time at $dest_st(q)$ depends on the departure event that is selected from the set $E_{succ}^{max}(e, t)$, the value t has no influence on this optimization. Hence, for the minimization of the travel time, it is sufficient to select the departure event that minimizes the expected arrival time. The latter value is computed as follows.

Expected arrival time at the destination From Sect. 5.4.1 (cf. *Recursive step*), recall the definition of $\varrho(e, t, \tilde{e}, t_{arr})$. Under the assumption that the destination is reached no later than $deadline(q)$, the expected arrival time at the destination station equals

$$\left(\sum_{\substack{t_{arr} \in TIMES \\ t_{arr} \leq deadline(q)}} \varrho(e, t, \tilde{e}, t_{arr}) \cdot t_{arr} \right) \cdot \left(\sum_{\substack{t_{arr} \in TIMES \\ t_{arr} \leq deadline(q)}} \varrho(e, t, \tilde{e}, t_{arr}) \right)^{-1}. \quad (5.4.4)$$

Exact definition of the tie-breaker In order to minimize the travel time as the secondary criterion, from the set $E_{succ}^{max}(e, t)$, we select for $\hat{e}_{dep}^{cont}(e, t)$ the successor event that minimizes the expected arrival time (cf. Eq. 5.4.4).

Alternatively, we can optimize a weighted sum of the travel time and the expected number of transfers (as introduced in Eq. 5.3.3 on page 86) as a secondary criterion.

More precisely, for each event in $E_{succ}^{max}(e, t)$, we compute a weighted sum; we then compare these weighted sums to each other. Given two successors $e_{dep}, e'_{dep} \in E_{succ}^{max}(e, t)$, let $arr_time(e_{dep})$ and $arr_time(e'_{dep})$ denote their expected arrival times respectively; moreover, $trans(e_{dep})$ and $trans(e'_{dep})$ denote their expected number of transfers respectively. To determine whether e_{dep} dominates e'_{dep} , we evaluate the constraint

$$\begin{aligned} w_a \cdot arr_time(e_{dep}) + w_t \cdot trans(e_{dep}) &< \\ w_a \cdot arr_time(e'_{dep}) + w_t \cdot trans(e'_{dep}) & \end{aligned} \quad (5.4.5)$$

where $w_a, w_t \in \mathbb{R}$ are the weights for the expected arrival time and the expected number of transfers respectively. In our computational study, in Sect. 9.4.9, we evaluate our extended recursive function with the tie-breaker from this section.

5.5 Properties of Complete Connections

We now discuss properties of complete connections which are relevant for our computational study in Sect. 9.4. They have already been introduced in Sect. 4.3.2 but the precise definition can be found in this section. In the following, for a complete connection c , let $e_{dep}^{init} \in E_{dep}^{dep_st(q)}$ denote the initial departure event of c (cf. Sect. 5.1.1).

Probability of success In Sect. 4.3.2, we introduced the probability of success $\rho(c)$ of a complete connection c . For each complete connection $c \in C_{best}(q)$, computed by our search component, we define $\rho(c) = \rho(e_{dep}^{init})$ (cf. Sect. 5.2.1). The probability $\rho(e_{dep}^{init})$ is computed as explained in Sect. 5.3.2.

Note: A complete connection $c \in C(q) \setminus C_{best}(q)$ could have a probability of success $\rho(c) < \rho(e_{dep}^{init})$ if it does not contain the best instructions for traveling. We will address the computation of the probability of the success of such a complete connection in Sect. 6.2.2.1.

Scheduled departure time In Sect. 4.3.2, we introduced the scheduled departure time $dep_time(c)$ of a complete connection; we define $dep_time(c) = sched(e_{dep}^{init})$.

Expected number of transfers For a complete connection, the expected number of transfers equals $\mu(e_{dep}^{init})$; the latter value is defined by the recursive formula from Eq. 5.3.3 (on page 86). This value is computed under the assumption that $dest_st(q)$ is reached no later than $deadline(q)$.

Expected arrival time The expected arrival time of a complete connection can be computed using the extended recursive function from Sect. 5.4. For a complete

connection c , it equals

$$\rho(e_{dep}^{init})^{-1} \cdot \left[\sum_{\substack{t_{arr} \in TIMES \\ t_{arr} \leq deadline(q)}} t_{arr} \cdot \underbrace{\left(\sum_{t \in supp(\phi(e_{dep}))} \phi(e_{dep}^{init}, t) \cdot \varrho(e_{dep}^{init}, t, t_{arr}) \right)}_{\text{the probability of arriving at } dest_st(q) \text{ at } t_{arr}} \right]. \quad (5.5.1)$$

This value is computed under the assumption that $dest_st(q)$ is reached no later than $deadline(q)$.

Expected travel time For a complete connection, this value is the time different between the expected arrival time at $dest_st(q)$ and the scheduled departure time at $dep_st(q)$. This value is computed under the assumption that $dest_st(q)$ is reached no later than $deadline(q)$.

5.6 Walk Trips in Complete Connections

So far, we supported transfers with walk trips (cf. Sect. 1.2) in complete connections—cf. the definitions of $\hat{e}_{dep}^{cont}(e, t)$ and $E_{succ}(e, t)$ in Sect. 5.1.2 and 5.3.1 respectively. We now discuss how to support walk trips at the beginning and at the end of complete connections.

Note: We do not support complete connections consisting of direct walk trips from $dep_st(q)$ to $dest_st(q)$.

Walk trips from the departure station In Sect. 5.1.1, we introduced the initial departure event of a complete connection; so far, we have assumed that the passenger leaves the departure station of the query via a train move. Actually, the passenger could walk from $dep_st(q)$ to another station that is connected to $dep_st(q)$ via a walk trip, and continue the journey from there by public transport. We support this case as follows. Given a query q , first we define the sets

$$S_{walk}^{dep_st(q)} = \left\{ s \in S \mid (dep_st(q), s) \in WALKS \right\}$$

and

$$E_{dep}^{walk} = \bigcup_{s \in S_{walk}^{dep_st(q)}} \left\{ e_{dep}^s \in E_{dep}^s \mid sched(e_{dep}^s) \leq deadline(q) \right\}.$$

Then, for each $e_{dep}^s \in E_{dep}^{walk}$, we create an artificial train move (e'_{dep}, e'_{arr}) , and add it to the timetable; this artificial train move is defined as follows:

- e'_{dep} is a departure event at $dep_st(q)$,

- $sched(e'_{dep}) = sched(e^s_{dep}) - min_dur(dep_st(q), s)$,
- $sched(e'_{arr}) = sched(e_{arr})$ where e_{arr} is the arrival event of the train move $(e^s_{dep}, e_{arr}) \in TM$,
- $min_dur(e'_{dep}, e'_{arr}) = min_dur(e^s_{dep}, e_{arr}) + min_dur(dep_st(q), s)$,
- $\forall t \in TIMES : \phi(e'_{dep}, t) = \phi(e^s_{dep}, t + min_dur(dep_st(q), s))$, and
- (e^s_{dep}, e_{arr}) , (e'_{dep}, e'_{arr}) , and their respective events are equivalent in the rest of the attributes.

From then on, these artificial train moves and their events are part of the timetable, and the corresponding walk trips can be part of the computed complete connections. They are only used for the computations for the query q , and are removed from the timetable as soon as the search component is finished with processing q .

Walk trips to the destination station In Sect. 5.3.2 and 5.4.1, the recursion anchor is at the arrival events at $dest_st(q)$; so far, we have assumed that the passenger arrives at the destination station of the query via a train move. Actually, the passenger could arrive at a station that is connected to $dest_st(q)$ via a walk trip and walk to the destination station. We support this case as follows. Given a query q , first we define the sets

$$S_{walk}^{dest_st(q)} = \left\{ s \in S \mid (s, dest_st(q)) \in WALKS \right\}$$

and

$$E_{arr}^{walk} = \bigcup_{s \in S_{walk}^{dest_st(q)}} \left\{ e_{arr}^s \in E_{arr}^s \mid sched(e_{arr}^s) + min_dur(s, dest_st(q)) \leq \right. \\ \left. deadline(q) \right\}. \quad (5.6.1)$$

Then, for each $e_{arr}^s \in E_{arr}^{walk}$, we create an artificial train move (e'_{dep}, e'_{arr}) , and add it to the timetable; it is defined as follows:

- e'_{arr} is an arrival event at $dest_st(q)$,
- $sched(e'_{dep}) = sched(e_{dep})$ where e_{dep} is the departure event of the train move $(e_{dep}, e_{arr}^s) \in TM$,
- $sched(e'_{arr}) = sched(e_{arr}^s) + min_dur(s, dest_st(q))$,
- $min_dur(e'_{dep}, e'_{arr}) = min_dur(e_{dep}, e_{arr}^s) + min_dur(s, dest_st(q))$,
- $\forall t \in TIMES : \phi(e'_{arr}, t) = \phi(e_{arr}^s, t - min_dur(s, dest_st(q)))$, and

- (e_{dep}, e_{arr}^s) , (e'_{dep}, e'_{arr}) , and their respective events are equivalent in the rest of the attributes.

From then on, these artificial train moves and their events are part of the timetable, and the corresponding walk trips can be part of the computed complete connections. They are only used for the computations for the query q , and are removed from the timetable as soon as the search component is finished with processing q .

5.7 Translation into Dynamic Programming

In Sect. 5.3 and 5.4, we presented recursive functions that compute the values $\hat{e}_{dep}^{cont}(e, t)$ and $\varrho(e, t)$ for $e \in E$ and $t \in \text{supp}(\phi(e))$. Each of these recursive functions can be translated into a *dynamic programming* approach. The idea of dynamic programming is that each subproblem is solved once and its solution is available when it is required (see [CLRS01], Chapter 15, Page 323). According to our problem, by dynamic programming, we compute each of the values $\hat{e}_{dep}^{cont}(e, t)$ and $\varrho(e, t)$ —for each event $e \in E$ and each time $t \in \text{supp}(\phi(e))$ —only once, and store the computed values such that they are available whenever required.

In Sect. 5.7.1, we present our dynamic programming approach to computing the values $\hat{e}_{dep}^{cont}(e, t)$ and $\varrho(e, t)$. It can be applied to compute optimal outcomes for all variants of the problem definition from Sect. 4.3.3. Once the dynamic programming approach is performed, the optimal outcome of the search component can be delivered as discussed in Sect. 5.2.2.

Exclusively for the problem variant from Sect. 4.3.3.1 (cf. *Latest Departure Subject to Arrival Guarantee*), we then extend our dynamic programming approach by early termination in order to achieve a better performance. This extension is presented in Sect. 5.7.2.

5.7.1 The Approach

Let $\varrho(e, t)$ and $\varrho(\tilde{e}, \tilde{t})$ be defined as in Eq. 5.3.2 (on page 85). We perform the computations from Sect. 5.3 and 5.4 in a specific order such that the following condition is satisfied.

Condition 5.7.1 For events $e \in E$ and $\tilde{e} \in E_{succ}(e, t)$ as well as for $t \in \text{supp}(\phi(e))$ and $\tilde{t} \in \text{supp}(\phi(\tilde{e}))$, the value $\varrho(\tilde{e}, \tilde{t})$ must be computed and be available prior to the computation of the value $\varrho(e, t)$. This condition also applies to Eq. 5.4.1 (on page 88).

Our dynamic programming approach is presented in Algorithm 5.1. Given a set $E_{input} \subseteq E$, the function $\text{process_events}(E_{input})$ from Algorithm 5.1 computes the values

```

1 Function process_events( $E_{input}$ )
   Data: Set  $E_{input} \subseteq E$ ; timetable  $TT$ ; all functions  $\phi(e)$ ,  $e \in E$ ; probability
   distributions describing random prepared times and random train
   move durations
   Result:  $\varrho(e, t)$  and  $\hat{e}_{dep}^{cont}(e, t)$  for each  $e \in E_{input}$  and  $t \in supp(\phi(e))$ 
2 Create empty queue;
3 foreach  $e_{arr} \in \{E_{input} \cap E_{arr}^{dest\_st(q)}\}$  do
4   | queue.push( $e_{arr}$ );
5 end
6 while queue is not empty do
7   |  $e \leftarrow$  queue.pop();
8   | /* Compute the values  $\varrho(e, t)$  and  $\hat{e}_{dep}^{cont}(e, t)$  */
9   | foreach  $t \in supp(\phi(e))$  do
10  |   | if  $e \in E_{arr}$  then
11  |     | ( $\varrho(e, t)$ ,  $\hat{e}_{dep}^{cont}(e, t)$ )  $\leftarrow$  process_arrival( $e, t$ );
12  |   | else /*  $e \in E_{dep}$  */
13  |     |  $\varrho(e, t) \leftarrow$  process_departure( $e, t$ );
14  |   | end
15  |   | /* Add the predecessor events to the queue */
16  |   | foreach  $(e', e) \in \{TM \cup TRANS \cup DW\}$  do
17  |     | if  $e' \in E_{input}$  then
18  |       | queue.push( $e'$ );
19  |     | end
20  |   | end
21 end

```

Algorithm 5.1: The dynamic programming approach to processing all events in a given set $E_{input} \subseteq E$ in order to compute the values $\varrho(e, t)$ and $\hat{e}_{dep}^{cont}(e, t)$ for each event $e \in E_{input}$ and each $t \in supp(\phi(e))$. The functions process_departure and process_arrival compute the values $\varrho(e, t)$ and $\hat{e}_{dep}^{cont}(e, t)$ as described in Sect. 5.3 and 5.4; the value $\hat{e}_{dep}^{cont}(e, t)$ is computed only for arrival events.

$\varrho(e, t)$ and $\hat{e}_{dep}^{cont}(e, t)$ for each event $e \in E_{input}$ and each time $t \in \text{supp}(\phi(e))$. In this section, the reader may assume that E_{input} equals E ; in Sect. 5.8, we will explain that this set can be limited to *relevant* events. The functions $\text{process_departure}(e, t)$ and $\text{process_arrival}(e, t)$ compute the values $\varrho(e, t)$ and $\hat{e}_{dep}^{cont}(e, t)$ by the recursive formulas from Sect. 5.3 and 5.4 for departure and arrival events respectively. Each value computed for some event is subsequently stored so that it is available whenever required.

The queue from Algorithm 5.1 maintains events. It is a dial queue as introduced in Sect. 3.13.4 with the following difference: the *pop* operation delivers the events sorted according to their scheduled event times in descending order; among all events with the same scheduled event time, departure events are delivered first. The *push* operation adds a new element to the queue only if it is not already contained in the queue.

Algorithm 5.1 fulfills Condition 5.7.1 (on page 93) as follows. First, for each train move $(e, \tilde{e}) \in TM$, the event \tilde{e} is processed prior to e since the event e is added to the queue only after e' has been processed—cf. Line 15 in Algorithm 5.1. Second, the *pop* operation of the queue (described above) ensures that all departure events from the set $E_{succ}(e_{arr}, t)$ of successor events of an arrival event $e_{arr} \in E_{arr}$ are visited and processed prior to e_{arr} . The latter statement is based on the following assumption.

Assumption 5.7.1 We assume that the following constellation does not exist in the timetable; the constellation is illustrated in Fig. 5.2 (on page 96). For an arrival event $e_{arr}^{s1, tr1} \in E_{arr}$ and a time $t \in \text{supp}(\phi(e_{arr}^{s1, tr1}))$, let the set $E_{succ}(e_{arr}^{s1, tr1}, t)$ of successor events (cf. Sect. 5.3.1) contain the departure events $e_{dep}^{s2, tr2}$ and $e_{dep}^{s3, tr3}$; it is irrelevant whether or not $s1 = s2$ and $s1 = s3$. Moreover, let $e_{arr}^{s4, tr2}$ and $e_{arr}^{s5, tr3}$ denote the arrival events of the train moves starting with $e_{dep}^{s2, tr2}$ and $e_{dep}^{s3, tr3}$ respectively. Algorithm 5.1 could process the event $e_{arr}^{s1, tr1}$ prior to one of its both successors if all of the five events—specified above—have the same scheduled event time. As mentioned, we assume that this constellation does not exist in the timetable.

This assumption is realistic for real timetables in Germany that we evaluate in our computational study. For a timetable which violates this assumption, instead of the approach presented in Algorithm 3.1, we propose to process the nodes in a topological order (see [CLRS01], Chapter 22, Page 549). A topological ordering exists for the events in our timetable since the timetable can be represented as a directed acyclic graph as described in Sect. 2.1.

5.7.2 Early Termination

We now present an improvement of the dynamic programming approach; it can be applied when solving the problem variant from Sect. 4.3.3.1 (cf. *Latest Departure Subject to Arrival Guarantee*). Algorithm 5.1 processes the events sorted by their scheduled event

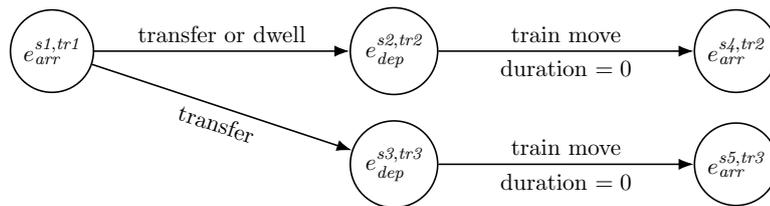


Figure 5.2: The figure illustrates a constellation of the events in the timetable; all five events are assumed to have the same scheduled event time. This constellation could result in a processing of the events in a wrong order. For more details, see Assumption 5.7.1 (on page 95).

times in descending order. Consequently, the departure event e_{dep} at $dep_st(q)$ with a probability of success $\rho(e_{dep}) \geq \rho_{req}(q)$ that is visited and processed by Algorithm 5.1 first is the initial departure event (cf. Sect. 5.1.1) of the optimal complete connection for the query q . Thus, as soon as this departure event is found, we can terminate the algorithm and deliver the optimal complete connection. This early termination could take place immediately after Line 14 in Algorithm 5.1. In our computational study, in Sect. 9.4.12, we demonstrate the speedup obtained by this early termination.

5.8 Relevant Events

In Sect. 5.7, we presented our dynamic programming approach to computing the values $\varrho(e, t)$ and $\hat{e}_{dep}^{cont}(e, t)$ for all events in the set E_{input} . We could define $E_{input} = E$ and compute the outcome for every query by our dynamic programming approach; in practice, it would be very inefficient. It is even unnecessary to compute the values $\varrho(e, t)$ and $\hat{e}_{dep}^{cont}(e, t)$ for each event $e \in E$.

Set $E_{relevant}$ of relevant events For each query, the computations can be restricted to a subset of the events that we call the set $E_{relevant} \subseteq E$ of all *relevant* events. It contains each event that is relevant for the computation of the outcome for the query. In other words, the result of the dynamic programming approach is the same for $E_{input} = E$ and $E_{input} = E_{relevant}$. As announced in Sect. 5.3.1, we also restrict the set of successor events to relevant events. More precisely, for $e \in E_{relevant}$ and $t \in \text{supp}(\phi(e))$, from the set $E_{succ}(e, t)$, we remove all events that are not contained in $E_{relevant}$.

Our definition of the set $E_{relevant}$ implies that $\varrho(e, t) = 0$ and $\rho(e) = 0$ for each $e \in E \setminus E_{relevant}$ and each $t \in \text{TIMES}$. Moreover, the value $\hat{e}_{dep}^{cont}(e_{arr}, t)$ is set to *void* for each $e_{arr} \in E_{arr} \setminus E_{relevant}$ and each $t \in \text{TIMES}$ —recall that this value is defined for arrival events solely.

Overview In Sect. 5.8.1, we precisely define when an event is called *relevant*. In Sect. 5.8.2, we demonstrate how to discover the set of all relevant events in a pre-processing step prior to the dynamic programming approach from Sect. 5.7. Afterwards, in Sect. 5.9, we present a very efficient incremental approach that integrates the discovery of relevant events into the dynamic programming approach from Sect. 5.7, and incrementally builds the optimal outcome.

From Sect. 4.3.3, recall that we introduced different variants of our problem definition. While the approach from Sect. 5.8.2 can be employed to solve each of the problem variants from Sect. 4.3.3.1–4.3.3.5, the incremental approach from Sect. 5.9 is exclusively designed for the problem variant from Sect. 4.3.3.1 (cf. *Latest Departure Subject to Arrival Guarantee*). By the incremental approach a significantly better performance can be obtained as evaluated in Sect. 9.4.12.

5.8.1 Definition of Relevant Events

Relevant events The relevant events for a query q are defined as follows. We first introduce two sets of events: $\tilde{E}_{dep}^{dep-st(q)} \subseteq E_{dep}^{dep-st(q)}$ and $\tilde{E}_{arr}^{dest-st(q)} \subseteq E_{arr}^{dest-st(q)}$. An event $e \in E$ is called *relevant* for q if there is a train connection (cf. Sect. 1.2) in the timetable that connects some departure event in $\tilde{E}_{dep}^{dep-st(q)}$ to some arrival event in $\tilde{E}_{arr}^{dest-st(q)}$ via the event e . In the following, we define the sets $\tilde{E}_{dep}^{dep-st(q)}$ and $\tilde{E}_{arr}^{dest-st(q)}$.

The sets $\tilde{E}_{dep}^{dep-st(q)}$ and $\tilde{E}_{arr}^{dest-st(q)}$ We distinguish between the different variants of our problem definition from Sect. 4.3.3; the sets $\tilde{E}_{dep}^{dep-st(q)}$ and $\tilde{E}_{arr}^{dest-st(q)}$ are defined as follows.

- For “latest departure subject to arrival guarantee” (cf. Sect. 4.3.3.1) and “maximum probability of success” (cf. Sect. 4.3.3.2), we define

$$\begin{aligned} \tilde{E}_{dep}^{dep-st(q)} &= \{ e_{dep} \in E_{dep}^{dep-st(q)} \mid sched(e_{dep}) \leq deadline(q) \} \\ \tilde{E}_{arr}^{dest-st(q)} &= \{ e_{arr} \in E_{arr}^{dest-st(q)} \mid sched(e_{arr}) \leq deadline(q) \}. \end{aligned} \quad (5.8.1)$$

- For “on-trip scenario at a station” (cf. Sect. 4.3.3.3), we define

$$\begin{aligned} \tilde{E}_{dep}^{dep-st(q)} &= \{ e_{dep} \in E_{dep}^{dep-st(q)} \mid ontrip_time \leq sched(e_{dep}) < deadline(q) \} \\ \tilde{E}_{arr}^{dest-st(q)} &= \{ e_{arr} \in E_{arr}^{dest-st(q)} \mid ontrip_time \leq sched(e_{arr}) < deadline(q) \} \end{aligned} \quad (5.8.2)$$

where *ontrip_time* is as defined in Sect. 4.3.3.3.

- For “on-trip scenario in a train” (cf. Sect. 4.3.3.4), we define

$$\begin{aligned}\tilde{E}_{dep}^{dep-st(q)} &= \{ e_{dep} \} \\ \tilde{E}_{arr}^{dest-st(q)} &= \{ e_{arr} \in E_{arr}^{dest-st(q)} \mid sched(e_{dep}) \leq sched(e_{arr}) < deadline(q) \}\end{aligned}\quad (5.8.3)$$

where e_{dep} is the departure event of the train move where the passenger is located.

- For “minimum expected travel time” (cf. Sect. 4.3.3.5), we define

$$\begin{aligned}\tilde{E}_{dep}^{dep-st(q)} &= \{ e_{dep} \in E_{dep}^{dep-st(q)} \mid sched(e_{dep}) \geq earliest_dep_time \wedge \\ &\quad sched(e_{dep}) \leq latest_dep_time \} \\ \tilde{E}_{arr}^{dest-st(q)} &= \{ e_{arr} \in E_{arr}^{dest-st(q)} \mid sched(e_{arr}) \geq earliest_dep_time \}\end{aligned}\quad (5.8.4)$$

where $earliest_dep_time$ and $latest_dep_time$ are as defined in Sect. 4.3.3.5.

Note: In Eq. 5.8.1, we ignored arrival events at $dest_st(q)$ with $sched(e_{arr}) > deadline(q)$ where there is a time $t \leq deadline(q)$ such that $\phi(e_{arr}, t)$ is greater than zero. We renounced to add such events to $\tilde{E}_{arr}^{dest-st(q)}$ since their scheduled arrival times are later than the deadline, and using them could have an adverse effect on the sense of reliability from passengers’ point of view. Alternatively, the definition of the set $\tilde{E}_{arr}^{dest-st(q)}$ could be changed to

$$\{ e_{arr} \in E_{arr}^{dest-st(q)} \mid \exists t \in \text{supp}(\phi(e_{arr})), t \leq deadline(q) \}.$$

This consideration also applies to Eq. 5.8.2 and 5.8.3.

5.8.2 Discover in Pre-Processing

The set $E_{relevant}$ of relevant events can be discovered in a pre-processing step prior to the dynamic programming approach from Sect. 5.7. More precisely, to compute the optimal outcome for a query, we first discover the set $E_{relevant}$ and then call Algorithm 5.1 with $E_{input} = E_{relevant}$.

The approach to discovering the set $E_{relevant} \subseteq E$ is divided in two steps. First, in a step called *backward mark*, presented in Algorithm 5.2, we mark all events that are contained in train connections which end with arrival events in the set $\tilde{E}_{arr}^{dest-st(q)}$. The set $E_{relevant}$ is a subset of the set of all events marked.

In a subsequent step called *forward mark*, presented in Algorithm 5.3, we build the set $E_{relevant}$ of all relevant events. For this, we detect all events that are contained in train connections which start with departure events in $\tilde{E}_{dep}^{dep-st(q)}$ and exclusively consist

```

Data: Timetable  $TT$ ; set  $\tilde{E}_{arr}^{dest-st(q)}$ 
Result: Set  $E_{marked} \subseteq E$  of marked events
1  $E_{marked} \leftarrow \emptyset$ ;
2 foreach  $e_{arr} \in \tilde{E}_{arr}^{dest-st(q)}$  do
3 |   backward_mark( $e_{arr}$ );
4 end
5 Function backward_mark( $e \in E$ )
6 |   if  $e \notin E_{marked}$  then
7 | |    $E_{marked} \leftarrow E_{marked} \cup \{e\}$ ;
8 | |   foreach  $(e', e) \in \{TM \cup TRANS \cup DW\}$  do
9 | | |   backward_mark( $e'$ );
10 | |   end
11 |   end
12 end

```

Algorithm 5.2: The procedure of marking each event which is contained in some train connection that ends with an arrival event in the set $\tilde{E}_{arr}^{dest-st(q)}$.

```

Data: Timetable  $TT$ ; set  $\tilde{E}_{dep}^{dep-st(q)}$ ; set  $E_{marked} \subseteq E$  of events marked in the
backward mark step.
Result: Set  $E_{relevant} \subseteq E_{marked}$  of all relevant events.
1  $E_{relevant} \leftarrow \emptyset$ ;
2 foreach  $e_{dep} \in \tilde{E}_{dep}^{dep-st(q)}$  do
3 |   forward_mark( $e_{dep}$ );
4 end
5 Function forward_mark( $e \in E$ )
6 |   if  $e \notin E_{relevant} \wedge e \in E_{marked}$  then
7 | |    $E_{relevant} \leftarrow E_{relevant} \cup \{e\}$ ;
8 | |   foreach  $(e, e') \in \{TM \cup TRANS \cup DW\}$  do
9 | | |   forward_mark( $e'$ );
10 | |   end
11 |   end
12 end

```

Algorithm 5.3: The procedure of the forward mark step. It builds the set $E_{relevant}$ of all relevant events by detecting all events that are contained in train connections which start with departure events in $\tilde{E}_{dep}^{dep-st(q)}$ and exclusively consist of events that have been marked in the backward mark step.

of events that have been marked in the first step.

After the set of relevant events is discovered, we restrict the expensive computations—which are required in order to compute the optimal complete connection—to relevant events solely. This results in a significant improvement of the performance of the search component. However, the approach to discovering relevant events that is presented in this section visits a large number of events in the backward mark step; this is very time-consuming. In Sect. 5.11, we will present heuristics which can be applied in order to improve the performance of the approach from this section. In Sect. 5.9, we present a very efficient approach to discovering relevant events that is designed for the problem from Sect. 4.3.3.1 (cf. *Latest Departure Subject to Arrival Guarantee*). The approaches to discovering relevant events are evaluated in Sect. 9.4.12.

Note: For the problem variants from Sect. 4.3.3.4 and 4.3.3.5, it could be more efficient to switch the order of the forward and backward mark steps.

5.9 Discovering and Processing Incrementally

We developed an efficient approach that *incrementally* discovers relevant events and builds the optimal outcome. It terminates as soon as the optimal outcome is built, and delivers it. The relevant events in the set E_{relevant} are not discovered in a pre-processing step as in Sect. 5.8.2. Instead, the procedures of discovering relevant events and processing events are linked (the latter is performed by the dynamic programming approach from Sect. 5.7). In Sect. 5.9.1, we present an abstract view on our incremental approach. Afterwards, in Sect. 5.9.2, we detail the algorithm.

Note: This approach is exclusively designed for the problem from Sect. 4.3.3.1 (cf. Latest Departure Subject to Arrival Guarantee).

Note: In this section, we explain the approach combined with the method from Sect. 5.3; however, it could be combined with the extended method from Sect. 5.4 as well.

5.9.1 Abstract View

Our incremental approach to discovering and processing is iterative. For each query q , we start with an empty set $E_{\text{relevant}}^{\text{inc}}$; at each iteration, a new subset of E_{relevant} is discovered and added to $E_{\text{relevant}}^{\text{inc}}$. These newly discovered events are processed by the dynamic programming approach from Algorithm 5.1 (on page 94). This iterative procedure is terminated and the optimal outcome is delivered as soon as all events in the optimal complete connection have been added to $E_{\text{relevant}}^{\text{inc}}$ and the complete connection can be built. The advantage is that it is not necessary to discover all relevant events but

only a subset of $E_{relevant}$ that needs be discovered till an early termination is possible. In the following, we discuss the variant, invariant, and termination criterion of each iteration. Before that, we introduce the *ordered sequence of departure events*.

Ordered sequence of departure events Let e_1, e_2, \dots, e_n be an *ordered sequence* of all departure events from the set $\tilde{E}_{dep}^{dep-st(q)} \cap E_{relevant}$ (cf. Sect. 5.8) such that the events in this sequence are sorted by their scheduled departure times in *descending* order. Hence, as introduced in Sect. 5.8 (*Relevant events*), for $i = 1, \dots, n$, there are train connections from e_i to events in $\tilde{E}_{arr}^{dest-st(q)}$ (cf. Sect. 5.8) which exclusively consist of relevant events in the set $E_{relevant}$.

The variant Initially, the set $E_{relevant}^{inc}$ is empty. For each iteration $i = 1, \dots, n$, the variant is as follows:

1. The set $E_{relevant}^{inc}$ is extended by all events that are not still contained in $E_{relevant}^{inc}$ but are contained in train connections—existing in the timetable—which start with the departure event e_i (cf. *Ordered sequence of departure events*) and end with events in $\tilde{E}_{arr}^{dest-st(q)}$.
2. The values $\rho(e, t)$ and $\hat{e}_{dep}^{cont}(e, t)$ are computed for each event e that is inserted to $E_{relevant}^{inc}$ in the i -th iteration—and each time $t \in \text{supp}(\phi(e))$.

The invariant For each iteration $i = 1, \dots, n$, the invariant is as follows:

1. After i iterations, the set $E_{relevant}^{inc}$ comprises all events contained in all train connections—existing in the timetable—which start with the events e_1, e_2, \dots, e_i (cf. *Ordered sequence of departure events*) and end with events in $\tilde{E}_{arr}^{dest-st(q)}$.
2. For each event $e \in E_{relevant}^{inc}$ and each time $t \in \text{supp}(\phi(e))$, the values $\rho(e, t)$ and $\hat{e}_{dep}^{cont}(e, t)$ are correctly computed.

Termination criterion For the query q , the incremental approach *terminates* after $i \leq n$ iterations if $\rho(e_i) \geq \rho_{req}(q)$; in words, it terminates if a complete connection c with e_i as the initial departure event (cf. Sect. 5.1.1) is found where c satisfies the probability of success required by the query q . The complete connection c is *feasible* and *optimal* according to the definitions in Sect. 4.3.2 (also see below). After termination, c is delivered as the search component's outcome. In the worst case, the incremental approach terminates after n iterations if in the timetable there is no complete connection satisfying $\rho_{req}(q)$. All computations are limited to $E_{relevant}^{inc}$ as built till termination.

The termination as described above does not compromise optimality since the relevant departure events at $dep-st(q)$ are visited and processed in descending order of

their scheduled event times (cf. *Ordered sequence of departure events*). If the condition $\rho(e_i) \geq \rho_{req}(q)$ is true, the complete connection with the initial departure event e_i optimizes the objective function from Eq. 4.3.1 (on page 73, cf. *Latest Departure Subject to Arrival Guarantee*).

5.9.2 Detailed View

Up to now, we described the general idea, the variant, the invariant, and the termination criterion of the iterative approach. In the following, we detail how the approach discovers new relevant events that are added to $E_{relevant}^{inc}$ incrementally, and how these are processed.

The algorithm is presented in Algorithm 5.4 (on page 103). Initially, the set $E_{relevant}^{inc}$ is empty. In order to discover relevant events, the algorithm uses a queue (cf. *inc_queue*) that is equivalent to the queue from Algorithm 5.1; thus, the events are visited in descending order of their scheduled times. The queue is initialized with all events in the set $\tilde{E}_{arr}^{dest-st(q)}$. The iterative approach, introduced in Sect. 5.9.1, is then executed. Each iteration has three phases; in Algorithm 5.4 these phases can be found in Lines 8–11, Line 12, and Lines 13–15 respectively. In the following, we explain each phase in detail.

Phase 1 – discover relevant events The goal of the first phase of the i -th iteration is to discover all events contained in train connections—existing in the timetable—which end with arrival events in the set $\tilde{E}_{arr}^{dest-st(q)}$ and start with the departure event $e_i \in \tilde{E}_{dep}^{dep-st(q)} \cap E_{relevant}$; the latter is the i -th departure event at $dep-st(q)$ from the *ordered sequence of departure events* (cf. Sect. 5.9.1). Some of these events could have already been discovered and marked in prior iterations; thus, at the i -th iteration, we only seek for the events that are still not marked as relevant but are contained in the train connections mentioned above. The first phase has two steps; these are partly similar to the pre-processing approach from Sect. 5.8.2.

The first step is as follows. Recall that the queue is initialized with all events in $\tilde{E}_{arr}^{dest-st(q)}$. In the i -th iteration, using the queue, we visit and mark the events which are contained in train connections ending with events in $\tilde{E}_{arr}^{dest-st(q)}$; these events are visited in descending order of their scheduled times. This procedure is finished as soon as a departure event at $dep-st(q)$ is found; it is the i -th departure event at $dep-st(q)$ from the *ordered sequence of departure events*. In Algorithm 5.4, this step is called *inc.backward.mark*; the respective function extends the set E_{marked} of marked events (which grows in each iteration) and returns the mentioned departure event $e_i \in \tilde{E}_{dep}^{dep-st(q)} \cap E_{relevant}$.

In the second step, we subsequently detect all events that are contained in train

```

Data: Query  $q$ ; timetable  $TT$ ; all functions  $\phi(e)$ ,  $e \in E$ ; probability
distributions describing random prepared times and random train move
durations
Result: The optimal complete connection, if a feasible one exists.
1  $E_{relevant}^{inc} \leftarrow \emptyset$ ;
2  $E_{marked} \leftarrow \emptyset$ ;
3 Create empty inc_queue;
4 foreach  $e_{arr} \in \tilde{E}_{arr}^{dest-st(q)}$  do
5 | inc_queue.push( $e_{arr}$ );
6 end
7 while inc_queue is not empty do
8 |  $E_{new} \leftarrow \emptyset$ ;
9 |  $e_i \leftarrow \text{inc\_backward\_mark}()$ ; /* see below */
10 | inc\_forward\_mark( $e_i$ ,  $E_{new}$ ); /* see below */
11 |  $E_{relevant}^{inc} \leftarrow E_{relevant}^{inc} \cup E_{new}$ ;
12 | process\_events( $E_{new}$ ); /* cf. Algorithm 5.1 on page 94 */
13 | if  $\rho(e_i) \geq \rho_{req}(q)$  then
14 | | Return the optimal complete connection with  $e_i$  as the initial departure
15 | | event;
16 | end
17 end
18 Function inc\_backward\_mark()
19 | while inc_queue is not empty do
20 | |  $e \leftarrow \text{inc\_queue.pop}()$ ;
21 | |  $E_{marked} \leftarrow E_{marked} \cup \{e\}$ ;
22 | | foreach  $(e', e) \in \{TM \cup TRANS \cup DW\}$  do
23 | | | inc\_queue.push( $e'$ );
24 | | end
25 | | if  $e \in \tilde{E}_{dep}^{dep-st(q)}$  then break;
26 | end
27 | return  $e$ ;
28 end
29 Function inc\_forward\_mark( $e \in \tilde{E}_{dep}^{dep-st(q)}$ ,  $E_{new}$ )
30 | if  $e \notin E_{relevant}^{inc} \wedge e \notin E_{new} \wedge e \in E_{marked}$  then
31 | |  $E_{new} \leftarrow E_{new} \cup \{e\}$ ;
32 | | foreach  $(e, e') \in \{TM \cup TRANS \cup DW\}$  do
33 | | | inc\_forward\_mark( $e'$ );
34 | | end
35 | end
36 end

```

Algorithm 5.4: The approach to discovering relevant events and computing optimal complete connections incrementally.

connections which start with the departure event $e_i \in \tilde{E}_{dep}^{dep-st(q)} \cap E_{relevant}$ and exclusively consist of events that have been marked in the first step. All of them are relevant events; each of them is inserted into the set E_{new} if it is not already contained in $E_{relevant}^{inc}$. Hence, at the end of the second step of the first phase, the set E_{new} comprises all events which have been marked as relevant in the i -th iteration. The set $E_{relevant}^{inc}$ is then extended by the events in E_{new} . In Algorithm 5.4, the second step of the first phase is called *inc_forward_mark*.

Phase 2 – process events As described above, at the end of the first phase of the i -th iteration, the set E_{new} comprises all relevant events which have been discovered in the i -th iteration. For each $e \in E_{new}$ and each time $t \in \text{supp}(\phi(e))$, the second phase computes the values $\varrho(e, t)$ and $\hat{e}_{dep}^{cont}(e, t)$ —the latter value for arrival events solely. For this, the function *process_events* from Algorithm 5.1 is called. However, in contrast to the approach from Sect. 5.7, all events in the set $E_{relevant}^{inc} \setminus E_{new}$ have already been processed in prior iterations; for them, the above values need not be computed again but are persistently available. Thus, the function *process_events* only processes the events in E_{new} .

Phase 3 – check the termination criterion The incremental approach is terminated if for the departure event $e_i \in \tilde{E}_{dep}^{dep-st(q)} \cap E_{relevant}$ —from the ordered sequence of departure events—the condition $\rho(e_i) \geq \rho_{req}(q)$ is true. Otherwise, the approach is continued with the iteration $i + 1$.

This concludes the description of our approach to discovering relevant events and computing the optimal complete connection incrementally. The performance of this approach is evaluated in Sect. 9.4.12.

5.10 Asymptotic Complexity

In this section, we analyze the asymptotic complexity of the search component. Analogously to Sect. 3.10, the values c_m and c_a denote the costs of a multiplication and an addition (or subtraction) of two probabilities respectively. Moreover, we define

$$n_T = \max_{e \in E} |\text{supp}(\phi(e))| \quad \text{and} \quad n_D = \max_{e \in E_{dep}} |E_{arr}^{depend}(e_{dep})|$$

where the set $E_{arr}^{depend}(e_{dep})$ has been defined in Sect. 3.5.1.1. Last, we define

$$n_S = \max_{\substack{e \in E \\ t \in \text{supp}(\phi(e))}} |E_{succ}(e_{dep}, t)|.$$

Processing a single event An upper bound for the number of multiplications and additions required for Eq. 5.3.2 (on page 85) is

$$n_T^2 \cdot c_m$$

for each arrival event and

$$n_T^2 \cdot n_S \cdot (c_m + c_a + 2 \cdot n_D \cdot (c_a + c_m))$$

for each departure event. For Eq. 5.4.2 (on page 88) of the approach from Sect. 5.4, we obtain

$$n_T^2 \cdot |TIMES| \cdot c_m$$

for each arrival event and

$$n_T^2 \cdot n_S \cdot |TIMES| \cdot (c_m + c_a + 2 \cdot n_D \cdot (c_a + c_m))$$

for each departure event. The summation in Eq. 5.4.2 (on page 88) is over all times in *TIMES*; however, in practice, we maintain the information for which values for t_{arr} the probability $\varrho(e, t, \tilde{e}, t_{arr})$ is greater than zero. Thus, the summation is actually over a limited number of values.

In the worst case, for a complete graph, the asymptotic complexity to process an event is $\mathcal{O}(|E|^2)$. However, since the values n_S , n_T , and n_D are limited to small values in practice, an event can be processed in constant time.

Processing a set of events We now discuss the asymptotic complexity of the dynamic programming approach from Sect. 5.7 employed in order to process the set E_{input} of events. Let c_e denote the cost of processing a single event (see above). Let n_A be equal to $|TM \cup TRANS \cup DW|$ where each of these sets of activities is limited to the respective activities between each event pair e, e' contained in E_{input} . The cost of processing the set E_{input} is

$$|E_{input}| \cdot c_e + n_A \cdot |E_{input}|$$

where the second summand is required since the queue performs a duplicate check for each push operation; we assume a simple duplicate check that iterates over all events in the queue. In our computational study in Sect. 9.4, we use an implementation that requires these duplicate checks. Alternatively, we could mark each event that is added to the queue so that duplicate checks are not necessary anymore. This would improve the asymptotic complexity of the dynamic programming approach to $\mathcal{O}(|E_{input}| + n_A)$.

Discovering relevant events in pre-processing Each of the backward and forward mark procedures from Sect. 5.8.2 has an asymptotic complexity of

$$\mathcal{O}(|E| + |TM \cup TRANS \cup DW|).$$

Discovering and processing incrementally We now discuss the asymptotic complexity of Algorithm 5.4 (on page 103). The while-loop (Lines 7–16) will perform a number of $|\tilde{E}_{dep}^{dep-st(q)}|$ iterations in the worst case. Over all iterations, each of the procedures for forward and backward mark costs

$$|E| + |TM \cup TRANS \cup DW|.$$

We assume that the events added to the queue are marked so that duplicate checks are not necessary anymore.

Over all iterations, the procedure of processing relevant events costs

$$|E_{relevant}| + |TM \cup TRANS \cup DW|;$$

again, each of the latter sets of activities is limited to the respective activities between each event pair e, e' contained in $E_{relevant}$. For the termination criterion, the probability $\rho(e_i)$ is computed which costs $n_T \cdot (c_a + c_m)$. Altogether, the asymptotic complexity of Algorithm 5.4 is

$$\mathcal{O}(|E_{relevant}| + |TM \cup TRANS \cup DW|).$$

As we see, compared to the dynamic programming approach from Sect. 5.7 combined with the approach to discovering relevant events in pre-processing, the incremental approach from Sect. 5.9 does not reduce the asymptotic complexity. However, in practice, the incremental approach significantly reduces the number of events which are visited and processed; the computational study can be found in Sect. 9.4.12.

Note: In our computational study in Sect. 9.4, we use an implementation that requires duplicate checks.

5.11 Heuristics

In this section, we introduce four heuristics which accelerate the search component but potentially result in suboptimal outcomes for some queries. All heuristics accelerate the response times of the search component by decreasing the size of the set of relevant events (cf. Sect. 5.8). More precisely, for a given query, we remove from the set $E_{relevant}$ events that are relevant according to Sect. 5.8.1 but are unlikely to be part of the

optimal complete connection for the query.

The heuristic presented in Sect. 5.11.1 can be used for the pre-processing approach (cf. Sect. 5.8.2) as well as for the incremental approach (cf. Sect. 5.9) to discovering relevant events. The other heuristics presented in this section are only beneficial for the pre-processing approach; the incremental approach efficiently delivers optimal outcomes without the need for them. In Sect. 9.4.11, we will evaluate all heuristics.

5.11.1 Maximum Waiting Time of Passengers

As discussed by [Sch09] (cf. Sect. 10.3.3.2), passengers do not desire long waiting times at stations. [Sch09] proposes a realistic assumption to limit the maximum waiting time of passengers at a station for a transfer. Theoretically, this assumption is a heuristic that potentially results in suboptimality. However, from a practical point of view, this loss of optimality is negligible as will be discussed in Sect. 9.4.11. We use this realistic assumption as a heuristic in order to significantly reduce the number of relevant events as follows.

This heuristic modifies the definition of the set $E_{succ}(e_{arr}, t)$ of successor events from Sect. 5.3.1. Let $t_{waiting}^{max}$ denote the maximum waiting time of passengers. For $e_{arr} \in E_{arr}$ and $t \in \text{supp}(\phi(e_{arr}))$, we remove from the set $E_{succ}(e_{arr}, t)$ each departure event e_{dep} for which the condition

$$\text{sched}(e_{arr}) + \text{min_dur}(e_{arr}, e_{dep}) + t_{waiting}^{max} < \text{sched}(e_{dep}) \quad (5.11.1)$$

is true. The parameter $t_{waiting}^{max}$ is part of the experimental setting and will be defined in Sect. 9.4.11.

Based on the modification of $E_{succ}(e_{arr}, t)$, we modify the procedures of discovering relevant events from Sect. 5.8.2 and 5.9. Thus, the for-loops in Algorithm 5.2 on page 99 (Line 8) and Algorithm 5.3 on page 99 (Line 8) are adjusted according to the constraint from Eq. 5.11.1 (see above). In Algorithm 5.4 this adjustment applies to Lines 22 and 32 respectively.

Note: This heuristic is applicable to all problem variants from Sect. 4.3.3.

5.11.2 Earliest Arrival Time at the Destination

We define an earliest time for the arrival at the destination station. This heuristic reduces the size of the set $\tilde{E}_{arr}^{dest_st(q)}$ from Sect. 5.8.1 and, consequently, the number of events which are marked as relevant. The earliest arrival time is selected depending on $\text{deadline}(q)$ and is part of our experimental setting in Sect. 9.4.11.

5.11.3 Earliest Departure Time

We define an earliest time for the departure from $dep_st(q)$. This heuristic reduces the size of the set $\tilde{E}_{dep}^{dep_st(q)}$ from Sect. 5.8.1 and, consequently, the number of events which are marked as relevant. The earliest departure time is selected depending on $deadline(q)$ and an estimation of the duration of the complete connection; the selection is part of our experimental setting in Sect. 9.4.11.

If this heuristic is applied, the procedure of discovering relevant events is modified as follows. In Algorithm 5.2 on page 99 (Line 6), we do not mark the event e if $sched(e)$ is earlier than the earliest departure time; analogously, its predecessors are not visited anymore. In Algorithm 5.4 (on page 103), the same behavior could be applied to the function *inc_backward_mark*.

5.11.4 Cycles Through the Departure Station

When a transfer breaks during the complete connection, in rare, exotic cases, it may be preferable to travel back to the departure station and take another train from there. Forbidding such cycles decreases the number of events which are marked as relevant. To apply this heuristic, in Algorithm 5.2 on page 99 (Line 8), we do not visit the predecessors of an event anymore if the event is at $dep_st(q)$. Again, in Algorithm 5.4 (on page 103), the same behavior could be applied to the function *inc_backward_mark*.

5.12 Parallelization

In his master's thesis [Glö15], Peter Glöckner developed and evaluated approaches to computing optimal complete connections in parallel. There are two general ideas for a parallelization. First, for each event $e \in E$ and each pair of times $t, t' \in TIMES$ where $t \neq t'$, the values $\varrho(e, t)$ and $\hat{e}_{dep}^{cont}(e, t')$ can be computed separately and in parallel. Second, events in the timetable can be processed in parallel if the computations do not depend on each other. In [Glö15], such a parallelization approach is discussed.

Briefly, by a parallelization, the computations can be accelerated. However, the parallelization is not in the focus of this work, and we refer to [Glö15] for more details.

5.13 Implementation

In this section, we provide details on our implementation of the search component from Chapter 5. We used the programming language C++.

5.13.1 Accessing the Timetable

In our implementation, we used the extended variant of the time-expanded graph model as introduced in Sect. 3.13.1. When discovering relevant events (cf. Sect. 5.8) and processing them (cf. Sect. 5.7), we access the predecessors and successors of events in the timetable as follows—such an access is required for example in Lines 22 and 32 in Algorithm 5.4 on page 103.

Note: In our implementation, the number of successors and predecessors of arrival nodes is significantly reduced by applying the heuristic from Sect. 5.11.1 (cf. Maximum Waiting Time of Passengers).

Successors Each departure node (representing a departure event) has exactly one successor; namely, the arrival node of its outgoing train edge. From each arrival node $v_{arr}^{s,tr} \in V_{arr}^s$, the succeeding departure node $v_{dep}^{s,tr}$ can be reached over the dwell edge except for the very last arrival node of each train. Moreover, the other successors of $v_{arr}^{s,tr}$ which are nodes of other trains—so that a transfer is required—are reachable over the change layer; that is, they are reachable either via special-transfer edges or via leaving, waiting, and entering edges.

Last, there are successors of $v_{arr}^{s,tr}$ which are reachable via walk trips (cf. Sect. 1.2, *Transfers with walk trips*). In order to find them, we access each station $s2$ which is connected to s via a walk trip. We then find each departure node $v_{dep} \in V_{dep}^{s2}$ where a transfer from $v_{arr}^{s,tr}$ to v_{dep} is feasible according to the timetable; these are all departure events with a scheduled event time no earlier than the scheduled event time of $v_{arr}^{s,tr}$ plus the duration of the walk trip from s to $s2$ (cf. Sect. 1.2, *Transfers with walk trips*).

Predecessors The predecessors of each event can be accessed analogously to the successors; the graph need be traversed backwards. In our time-expanded graph, model this is possible since each node knows its incoming edges.

5.13.2 Relevant Events

In Sect. 5.8 and 5.9, we introduced the sets $E_{relevant}$ and $E_{relevant}^{inc}$ of relevant events and presented approaches to discovering them. In our implementation, we extend the nodes in the graph model by an additional attribute *state* that assumes the values *unknown*, *backward*, and *relevant*; the values have the following meanings:

- *unknown*: the node has not yet been visited;
- *backward*: the node has been visited and marked during the backward mark procedure; i.e. after performing Algorithm 5.2 (on page 99) or *inc_backward_mark*—in

the current or one of the previous iterations of Algorithm 5.4 (on page 103);

- *relevant*: the node has been visited and marked as *relevant* during the forward mark procedure; i.e. after performing Algorithm 5.3 (on page 99) or *inc_forward_mark*—in the current or one of the previous iterations of Algorithm 5.4 (on page 103).

5.13.3 The Search Labels

We now explain how we store the values computed during the search for complete connections. For each event e that is processed by Algorithm 5.1 (on page 94), we create exactly one *label*; it is a data structure that comprises

- the values $\varrho(e, t)$ or $\varrho(e, t, t_{arr})$ (cf. Sect. 5.3 and 5.4) for all $t \in \text{supp}(\phi(e))$,
- the values $\hat{e}_{dep}^{cont}(e, t)$ for each $t \in \text{supp}(\phi(e))$ if e is an arrival event,
- the expected number of transfers $\mu(e)$ as introduced in Sect. 5.3.4, and
- a list of pointers to all labels existing for (relevant) events that are successors (cf. Sect. 5.13.1) of the event e .

Hence, in our implementation, the queue in Algorithm 5.4 (on page 103) stores no events but the created labels. According to the last point in the enumeration above, the labels are connected to each other. More precisely, from the label that is created for a departure event e_{dep} at $dep\text{-}st(q)$, we can reconstruct the complete connection which has e_{dep} as the initial departure event; this is obtained by following the interconnected labels and the values $\hat{e}_{dep}^{cont}(e, t)$ (cf. Sect. 5.1).

Below, we detail how labels are created. We distinguish between two variants of our search approach: first, the dynamic programming approach from Algorithm 5.1 (on page 94) with the discovery of relevant events in pre-processing (cf. Sect. 5.8.2); second, our incremental approach from Sect. 5.9.

Dynamic programming with pre-processing All relevant events are discovered in a pre-processing step (cf. Sect. 5.8.2); the states—introduced in Sect. 5.13.2—of the relevant nodes are set to *relevant*. First, we create labels for all relevant events at the destination station and add them to the queue (cf. Line 3 in Algorithm 5.1).

Whenever a label created for an arrival node v_{arr} is pulled from the queue and processed, we create a new label for the departure node of the train edge ending with v_{arr} , and add it to the queue. On the other side, whenever a label created for a departure node is processed, for each of its predecessors, we create a new label and add it into the queue unless a label already exists for the predecessor; recall that each arrival event can

be the predecessor of multiple departure events. Furthermore, the processed departure label is added to the list of successor labels of its predecessors' labels.

Note: If a predecessor is not marked as relevant, we create no label for it.

Incremental approach The set E_{new} from Algorithm 5.4 (on page 103) is realized as a C++-vector that contains each node which is marked as relevant in the first phase of the current iteration. In the second phase, we execute Algorithm 5.1 (on page 94) with E_{new} as input. Whenever a departure node is processed, we manage all of its predecessors which are marked as relevant as explained above for the approach *Dynamic programming with pre-processing*. For the other predecessors, no labels are created but they are stored in a separate data structure (a binary search tree). For each predecessor, we maintain the list of all its successor labels. At the end of the first phase of each iteration—where new relevant nodes are discovered, we check each of the nodes which are stored separately for being discovered as relevant. Finally, for each of them which has been marked as relevant, we create a label and add it to the queue; it is then removed from the separate data structure which maintains the predecessors. An explanation follows.

Above (cf. *Dynamic programming with pre-processing*), after processing a departure node, we could completely ignore predecessors that were not marked as relevant—and create no labels for them—since all relevant nodes had been discovered in the pre-processing step. In contrast, in each iteration of the incremental approach, a new subset of the relevant nodes is discovered. Consequently, each departure node v_{dep} that is processed could have predecessors which are relevant but not discovered yet. On the one side, at the moment when v_{dep} is processed, we cannot create labels for those predecessors of v_{dep} since we do not know whether or not they are actually relevant. On the other side, each label should store pointers to all of its successor labels (see at the beginning of Sect. 5.13.3). By the separate data structure that maintains such predecessors, we overcome the problem described.

5.14 Conclusion

In this chapter, we presented our approach to computing optimal complete connections. For each relevant arrival event and each time in the support of the function describing its random event time, our search component computes the best departure event to continue the journey. It delivers a complete connection that guarantees an on-time arrival; more precisely, the destination is reached no later than the deadline with the probability of success required in the query. Subject to the on-time arrival guarantee, the scheduled departure time at the departure station is maximized.

Travel time optimization Our approach optimizes the travel time as a secondary criterion. On the one side, the scheduled departure time is maximized. On the other side, when selecting the best departure event to continue the journey, we proceed as follows. Subject to the probability of success, the continuation is selected that has the earliest expected arrival time; alternatively, a weighted sum of travel time and number of transfers is optimized.

Local public transport In Sect. 9.4, we evaluate our optimal approach. We will see that the approach is efficient for timetables comprising all long- and short-distance trains in Germany; such a timetable has more than a million departure and arrival events per day. However, our approach, as introduced in this chapter, is still not suitable for timetables which include local public transport such as buses and streetcars; such timetables are by orders of magnitude larger than timetables which comprise short- and long-distance trains solely. The support for local public transport is left to future research as will be discussed in Sect. 10.2.

Tickets Transportation companies usually allow passengers to purchase tickets for specific train rides; the tickets are often limited to specific train categories. For example, a passenger who has a ticket for a regional train is not allowed to use an intercity express. In order to facilitate passengers using complete connections as computed by our approach, transportation companies should accommodate passengers who are faced by delays. The passengers should be permitted to use other trains in the case of delays. At least, special tickets should be offered for complete connections. Independently, our approach could be easily adjusted to find complete connections that comprise trains of specific categories solely. To realize this, when discovering relevant events, we could omit all train moves of disallowed train categories.

This concludes our search component. In the next chapter, we will discuss our travel guidance component.

Chapter 6

Travel Guidance

As detailed in Sect. 4.4, we follow a two-stage approach in order to find optimal complete connections for the problem from Sect. 4.3. The first stage, realized by the search component (cf. Chapter 5), computes optimal complete connections for queries given. The second stage is realized by the *travel guidance component*, which is introduced in this chapter.

Two main problems Two principal problems overcome by the travel guidance component are the following. First, complete connections have a complex structure, and could contain large numbers of events; therefore, a clear and understandable visualization is required. Second, as announced in Sect. 4.3.2, the probability of the success of a complete connection can be guaranteed only at the time when it is computed. Whenever the network state has changed due to delays, the probability of the success of a formerly computed complete connection could be affected. Hence, the main value and necessity of the travel guidance component is to facilitate access to the instructions computed by the search component, and to preserve the optimality of the complete connections delivered.

General idea While traveling, the passenger is guided by the travel guidance component. This component interprets the instructions of the outcome of the search component. In brief, whenever the passenger waits at a station, the travel guidance component suggests which train to take next; and whenever the passenger sits in a train, it suggests what to do at the very next halt: to stay in the train or to leave it. The travel guidance component provides the travelers access to the complete connection (delivered by the search component). More precisely, it provides access to the values $\hat{e}_{dep}^{cont}(e, t)$ (cf. Sect. 5.1.2), computed for $t \in \text{supp}(\phi(e))$ and each event e contained in the complete connection.

The second problem described above (cf. *Two main problems*) is solved as follows.

Whenever a formerly computed complete connection is affected by delays, the travel guidance component updates it by a re-invocation of the search component. This update step delivers a complete connection that is optimal according to the current situation in the train network. Thus, the instructions provided to the traveler are kept optimal and up to date.

Related work [FMHS08] and [Sch09] presented an approach to the travel guidance component that checks train connections (as defined in Sect. 1.2) for their feasibility according to real times (cf. Sect. 1.3.2). It computes alternative continuations for infeasible train connections. Our approach to preserving the optimality of complete connections is based on their approach.

Overview We here discuss different possible realizations of the travel guidance component. A travel guidance component could solve each of the two above problems to a greater or lesser extent. In Sect. 6.1, we attend to the first problem; two different visualizations of complete connections are presented. In Sect. 6.2, the second problem is addressed. Finally, we conclude this chapter in Sect. 6.3.

Note: In this chapter, we present exemplary realizations of the travel guidance component. However, no evaluations of these approaches are presented since the travel guidance component is a straightforward implementation task rather than a scientific contribution. Hence, in this work, we discuss different variants of the travel guidance component only conceptually.

6.1 Interactive Visualization

In this section, we discuss how complete connections could be presented to travelers. We first demonstrate a web interface as a graphical user interface that provides a detailed view on complete connections. We then present a mobile application that provides a compact but clear presentation of complete connections; it is eminently suitable as a travel guidance component in order to accompany travelers during their journeys.

We want to thank Sebastian Fahnenschreiber, Felix Gündling, Steffen Maus, and Hannes Wernery who implemented the graphical user interface and the mobile application presented in this section.

6.1.1 Web Interface: a Detailed Visualization

We developed a web interface to visualize complete connections; it is illustrated in Fig. 6.1. It provides a visualization that is based on the illustration of complete con-

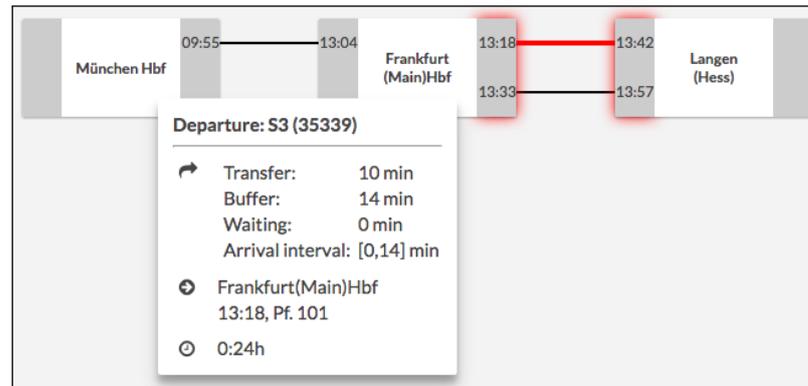


Figure 6.1: Web interface presenting a complete connection from *München Hbf* to *Langen(Hess)* with a transfer at *Frankfurt(Main)Hbf*. At *Frankfurt(Main)Hbf*, there are two different alternative continuations with scheduled departure times at 13:18 and 13:33 o'clock respectively. The user is suggested to take the first one if he/she arrives at *Frankfurt(Main)Hbf* with up to 14 minutes delay, and the second one if the delay is larger.

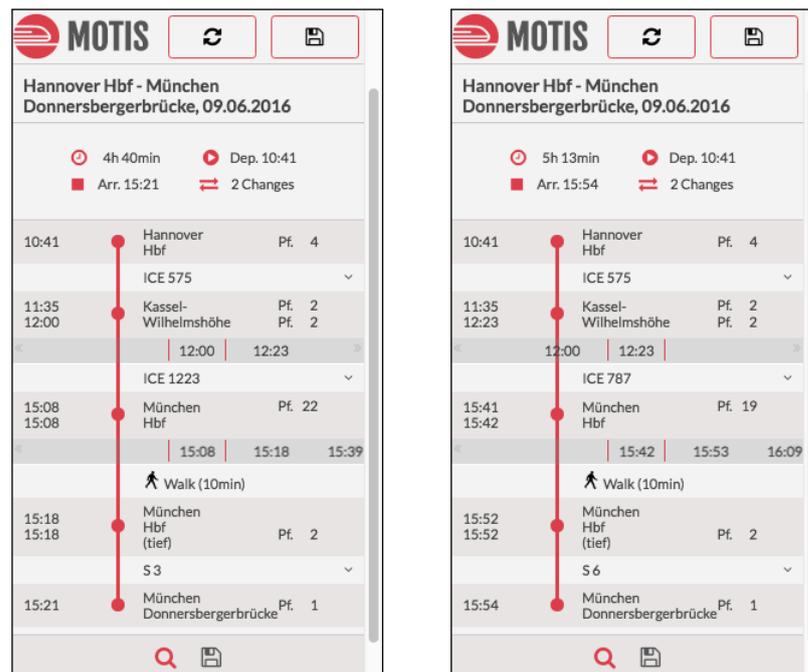


Figure 6.2: A mobile application presenting a complete connection from *Hannover Hbf* to *München Donnersbergerbrücke*. There are transfers at *Kassel-Wilhelmshöhe* and *München Hbf*; the latter is a transfer with a walk trip to *München Hbf(tief)*. At *Kassel-Wilhelmshöhe*, there are two alternative continuations that depart at 12:00 and 12:23 o'clock. By touching the departure times or swiping to the left or to the right, the user can switch between the available alternative continuations in order to gather information about them.

nections from Fig. 5.1 (on page 79). By rectangles, we represent departure and arrival events in the complete connection; from each rectangle (titled with the station name), a set of alternative continuations depart. The user can gather information about each alternative continuation by moving the mouse over each of them. In Appx. A.2, we present further screenshots of this web interface.

The web interface provides a global view on a complete connection. For simple complete connections with a relatively small number of alternative continuations, this view is clear and understandable. In contrast, for larger complete connections, this visualization is not suitable; it would be unclear and even confusing. To overcome this problem, we developed another visualization that is presented below in Sect. 6.1.2.

6.1.2 Mobile Application: a Compact Visualization

We developed a visualization that is eminently suitable for complete connections; it is illustrated in Fig. 6.2 (on page 115). We integrated it into a mobile application such that, while traveling, users can access the instructions which are computed by the search component and available in the complete connection.

The course of the complete connection is visualized top-down along the y-axis. If there are multiple alternative continuations which can be taken after an arrival event, their departure times are listed horizontally below the arrival event. The user has the possibility to click on each departure time or to swipe between them in order to access each alternative continuation.

Passenger localization In Fig. 6.2, the mobile application does not illustrate the information which alternative continuation should be taken next. It is possible to extend the visualization in order to display this information analogously to the visualization from Sect. 6.1.1: clarifying which alternative continuation is the best choice for each arrival delay expected. Another solution could be the following: the user enters his/her position as well as the current delay—in the case he/she is in a train, and the travel guidance component delivers the best instruction to continue the journey.

Alternatively, a travel guidance component that is informed about the real delay data—as known so far—of all trains in the complete connection could realize the following behavior. Using the delay data and the events in the complete connection, the passenger could be *localized*. Whenever the passenger has reached a station s with a train tr at a time t , the travel guidance component proposes the departure event $\hat{e}_{dep}^{cont}(e_{arr}^{s,tr}, t)$ to the passenger. In this way, the passenger is guided by the mobile application in every situation that occurs during the journey (see also *complete connection updates* in Sect. 6.2.2).

If at some station s a transfer is necessary due to the actual situation in the train

network, the passenger should be forewarned so that he/she is prepared for the transfer. If we assume that the travel guidance component is informed about the current delay of the train tr in which the passenger is sitting, few minutes prior to the arrival at the station s , the travel guidance component could predict the arrival time at s quite reliably. Based on this prediction, the passenger could be forewarned of the upcoming transfer. However, the best option to continue the journey can be delivered only after the real time of $e_{arr}^{s,tr}$ is definitively known.

Real delay data could be provided to the travel guidance component as to the search component. The passenger can be localized based on the complete connection and the current delay data of the trains. The localization could be improved by receiving GPS data from the smart phone of the passenger.

6.2 Preserving the Optimality of Complete Connections

We now focus on the second problem addressed at the beginning of Chapter 6; preservation of the optimality of complete connections after changes to the train network due to delays and other real-time incidents (cf. Sect. 1.3). We discuss two different variants of the travel guidance component; technically, the main difference between the variants is in the interaction between the search component and the travel guidance component. In 6.2.1, we present an interaction variant where the search component is executed once for each query, and the computed complete connection is delivered to the travel guidance component once. The result is an *offline database* for the travel guidance component; the complete connection is computed once and is not updated anymore. In Sect. 6.2.2, we present a second variant where both components are interconnected and exchange data as often as necessary; the result is an *online database* for the travel guidance component. Complete connections formerly computed are updated whenever the network state has changed because of delays and other disruptions. Only the second one solves the optimality problem introduced at the beginning of Chapter 6.

6.2.1 One-Time Interaction: Offline Database

The database of the travel guidance component could be a complete connection that is computed once by the search component. In this variant, the travel guidance component performs lookups on the computed data, and presents them to the user. The advantage of this variant is that after the booking stage, the computed complete connection can be stored in the mobile application. Thus, travelers could have detailed information about their complete connections on their smart phones without any need for an online Internet connection. If a transfer becomes infeasible because of train delays, the traveler has the possibility to look for other alternative continuations which

are already computed and available offline in the mobile application.

On the other side, the optimality of the computed complete connections can only be guaranteed at the booking time. As soon as the network state is changed according to delays and other disruptions (cf. Sect. 1.3), a complete connection which was optimal at the booking time could become suboptimal or even infeasible. The probability of the success of the complete connection could significantly sink, and trains contained in the complete connection could even be canceled. Hence, the optimality of complete connections can not be preserved using an offline database. This problem motivated us for another interaction variant resulting in an online database; it is presented in Sect. 6.2.2.

6.2.2 Ongoing Interaction: Online Database

We now discuss an ongoing interaction between the search component and the travel guidance component which results in an *online database* for the travel guidance component. Similar to the variant with an offline database from Sect. 6.2.1, for each query, the search component computes the optimal complete connection, and delivers it to the travel guidance component. In contrast to that variant, the interaction between the both components does not end at this point but the computed complete connection is *updated* whenever it is necessary because of changes to the train network (cf. Sect. 1.3). These updates require ongoing data exchanges between the search and travel guidance component.

Check and update Trivially, we could recalculate each complete connection formerly computed whenever the network state has changed because of delays and other disruptions. However, for a railway company with millions of travelers per day, this approach would require very large compute capacities. Thus, we minimize the number of the expensive recalculations as follows. Whenever the network state changes after the computation of a complete connection or the last time it was updated, we *check* whether the probability of the success of the complete connection is affected by the delays newly occurred. Only if this is the case, the complete connection needs to be *updated*; more precisely, the complete connection is recalculated. In Sect. 6.2.2.1 and 6.2.2.2, we detail the procedures of checking and updating complete connections.

6.2.2.1 Checking Complete Connections

The *check*, whether the probability of the success of a complete connection has changed since its computation or last update, can be triggered *manually* by the traveler or *automatically* by the travel guidance component.

Manual check If a traveler—who has decided for a complete connection—asks the travel guidance component for the validity of the complete connection, the travel guidance component could ask the search component to recalculate the probability of the success of the given complete connection as will be explained below (cf. Sect. 6.2.2.1, *Recalculation of the probability of success*). The traveler is then informed about how significantly the probability of success has changed since the computation of the complete connection or its last update. If the probability of success has fallen significantly, the traveler could manually trigger an update of the complete connection as will be explained in Sect. 6.2.2.2.

Automatic check A more sophisticated travel guidance component could monitor all complete connections selected by travelers, check them automatically, and inform travelers about significant changes. The checks could be performed either in certain time intervals or whenever a check is actually necessary. While the former can be simply realized, the latter requires a more complex logic and implementation. More precisely, for each complete connection c that is monitored and each event e contained in c , the travel guidance component need be informed about changes to the function $\phi(e)$ (cf. Sect. 3.3). Whenever a function $\phi(e)$ has changed for an event e in a complete connection c , the probability of the success of c has to be recalculated as follows.

Recalculation of the probability of success In Sect. 4.3.2, we introduced the probability of the success of a complete connection, and we presented a recursive function to compute this probability in Sect. 5.3. The selection of the best option (cf. *departure event* $\hat{e}_{dep}^{cont}(e, t)$) to continue the journey was integrated into the recursive function; cf. the maximization in Eq. 5.3.2 (on page 85).

To recalculate the probability of the success of a complete connection c , we can use the recursive function from Sect. 5.3 with one modification as follows. For every arrival event e_{arr} in the complete connection c , we keep all values for $\hat{e}_{dep}^{cont}(e_{arr}, t)$, stored in c , unmodified; these are the values which have originally been computed by the search component. More precisely, we assume that the set $E_{succ}(e_{arr}, t)$ of successors (cf. Sect. 5.3.1) solely comprises the departure event that has originally been selected for $\hat{e}_{dep}^{cont}(e_{arr}, t)$ in the complete connection c . Under this assumption, the value $\varrho(e, t)$ is then recalculated for each e in the complete connection c and each time $t \in \text{supp}(\phi(e))$.

After the modification described, the recursive function would not compute the optimal complete connection but recalculate the probability of the success of the originally computed complete connection c ; this is the probability to reach $dest_st(q)$ no later than $deadline(q)$ when following c .

6.2.2.2 Updating Complete Connections

If the recalculation of the probability of the success of a complete connection reveals that this probability has fallen significantly, an update of the complete connection is required to preserve optimality. This update could be triggered either manually by the passenger or automatically. For the latter, one could define a threshold for the minimum change to the probability of success that is required to trigger an update.

Technically, an update is a recalculation of the complete connection. If the traveler still has not commenced the journey, the search component is reinvoked with the original query of the traveler. On the other side, if the traveler is already en route, first, a localization is required, which can be obtained as discussed in Sect. 6.1.2. If he/she sits in a train, we recompute the complete connection by solving the problem “on-trip scenario in a train” from Sect. 4.3.3.4. If he/she is waiting at a station, we solve the problem “on-trip scenario at a station” from Sect. 4.3.3.3.

In such an update procedure, the search component also takes into account new alternative continuations that were unplanned but are possible due to delays which change the network state. In a real application, this would make perfect sense. In Fig. 6.3, by way of example, we demonstrate how a manual update could be triggered by the traveler. There, the traveler enters his/her position.

6.3 Conclusion

Our search component computes the optimal complete connection for each query q so that the required probability of success $\rho_{req}(q)$ is met given all information that is available at the time the search component is executed. The probability of the success of a complete connection can consequently be guaranteed only at that time. Whenever the network state has changed, for example due to delays, the probability of the success of a formerly computed complete connection could be affected. To overcome this problem, the travel guidance component checks the complete connection and, if necessary, updates it by a re-invocation of the search component. Moreover, the train guidance component is responsible for a comprehensive presentation of complete connections to travelers.

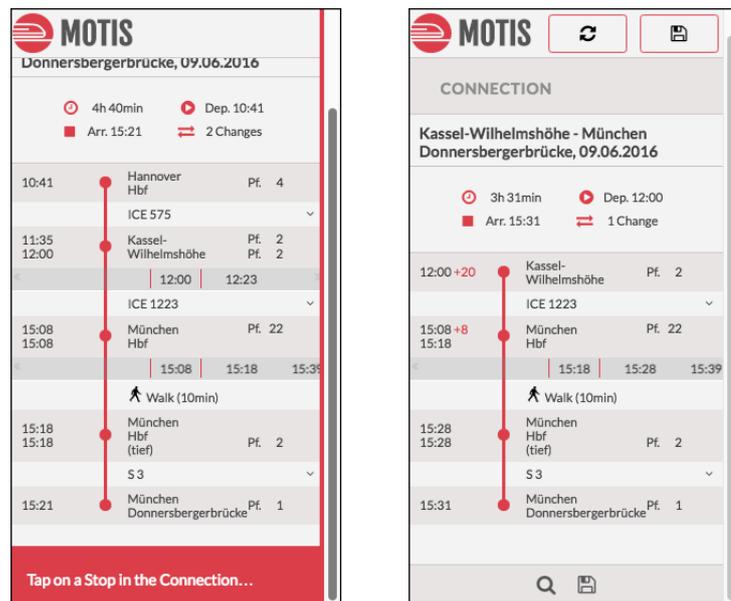


Figure 6.3: A mobile application that allows the traveler to update a complete connection manually. First, the traveler clicks on the *refresh* button; he/she then enters his/her position (left figure) as well as the current time. Subsequently, the search component is executed and an updated complete connection is presented (right figure). We can see that the train *ICE 1223* is delayed, and the alternative continuation departing from *München Hbf* at 15:08 o'clock is not contained in the updated complete connection anymore.

Chapter 7

Alternative Search Approaches

In Chapter 4, we introduced the problem of finding complete connections which have a high probability of success. We presented in Chapter 5 an approach that solves this problem. In this chapter, in Sect. 7.1, we discuss alternative approaches to computing complete connections. In contrast to the approach from Chapter 5, the approaches from this chapter are not optimal. However, these are approaches employed by travelers and state-of-the-art timetable information systems in order to find reliable train connections or even complete connections. In Sect. 9.4, we evaluate our optimal approach from Chapter 5 by its comparison to these alternative approaches.

After that, in Sect. 7.2, we present a further approach to the search for reliable train connection. This approach is based on Pareto Dijkstra (cf. Sect. 2.2) and optimizes the reliability of train connections as a Pareto criterion.

7.1 On-Time Arrival Guarantee: Alternative Approaches

As already announced, we now discuss alternative approaches to the solution of the problem specified in Sect. 4.3.3.1 (cf. *Latest Departure Subject to Arrival Guarantee*). First, in Sect. 7.1.1, we discuss two approaches used in state-of-the-art timetable information systems which compute attractive train connections but without regard for reliability. Then, in Sect. 7.1.2, we discuss an approach that finds train connections with increased buffer times for transfers and for the arrival prior to the deadline. Last, in Sect. 7.1.3, we present an alternative approach to computing complete connections.

Note: In Sect. 9.4, we compare our optimal approach from Chapter 5 to each of the approaches from this section in order to evaluate different properties of the optimal approach.

7.1.1 Disregarding the Probability of Success (*DP-L* and *DP-LP*)

In this section, we present two approaches that disregard the probability of success, and assume that all events take place at their scheduled event times. For each query q , each of these two approaches delivers a train connection (cf. Sect. 1.2) without alternative continuations. It is a train connection from $dep_st(q)$ to $dest_st(q)$ that has a scheduled arrival time at $dest_st(q)$ no later than $deadline(q)$ and satisfies the following condition respectively:

- For the first approach, denoted by *DP-L*, the scheduled departure time at $dep_st(q)$ is as *late* as possible.
- For the second approach, denoted by *DP-LP*, the train connection is the one with the *latest* scheduled departure time at $dep_st(q)$ among all Pareto optimal train connections (cf. Sect. 2.2) that arrive at $dest_st(q)$ within the time interval $[deadline(q) - \delta, deadline(q)]$. The parameter δ will be defined in our computational study in Sect. 9.4.2. The criteria optimized by the Pareto search are travel time and number of transfers.

Technically, both approaches are realized by the Pareto Dijkstra algorithm as introduced in Sect. 2.2.2. For both searches, the graph has to be traversed backwards (cf. *Backward search*). The first approach is a backward variant of the on-trip search; for the second approach, an interval for the arrival at $dest_st(q)$ is specified.

Probability of success In Sect. 3.7.2, we introduced the probability of the success of a train connection and presented formulas to compute this value. The probability of success for train connections computed by the approaches *DP-L* and *DP-LP* can be calculated by the method presented in Sect. 3.7.2.

In Sect. 9.4.6, we will evaluate how often such train connections satisfy the probability of success $\rho_{req}(q)$ required by the query q .

7.1.2 Minimum Buffer Times for Transfers and Arrival (*BT*)

From Sect. 1.2, recall that the minimum time required for a transfer activity $(e_{arr}^{s1,tr1}, e_{dep}^{s2,tr2}) \in TRANS$ is denoted as $min_dur(e_{arr}^{s1,tr1}, e_{dep}^{s2,tr2})$. According to the timetable, the condition

$$sched(e_{arr}^{s1,tr1}) + min_dur(e_{arr}^{s1,tr1}, e_{dep}^{s2,tr2}) \leq sched(e_{dep}^{s2,tr2})$$

is satisfied for each transfer activity. Let us denote the time difference

$$(sched(e_{dep}^{s2,tr2}) - sched(e_{arr}^{s1,tr1})) - min_dur(e_{arr}^{s1,tr1}, e_{dep}^{s2,tr2})$$

as the *buffer time* of the transfer activity $(e_{arr}^{s1,tr1}, e_{dep}^{s2,tr2})$. An intuitive approach to finding reliable train connections is to compute train connections with transfer activities that satisfy a *minimum buffer time*.

Note: A similar approach is provided on the official website of the German railway company [Bah16c]; users can specify a minimum buffer time for the transfers used in the train connections. That approach is the basis for the approach presented in this section.

The approach We denote this approach by *BT*. For each query q , the outcome of *BT* is a train connection (without alternative continuations) with a scheduled departure time at $dep_st(q)$ as late as possible such that a *minimum buffer time* τ is obeyed as follows:

1. The scheduled arrival time of the train connection at $dest_st(q)$ is no later than $deadline(q) - \tau$, and,
2. for each transfer activity in the train connection, the buffer time (introduced above) is greater than or equal to τ .

The value τ is a parameter in our experimental setting. In our computational study in Sect. 9.4.2 and 9.4.7, we will evaluate different values for τ .

Technically, the approach *BT* is realized by the Pareto Dijkstra algorithm (cf. Sect. 2.2.2, *Backward search*) with a modification: during the search for train connections, transfer activities are used only if they obey the minimum buffer time.

Probability of success Analogously to *DP-L* and *DP-LP*, the probability of success of train connections computed by the approach *BT* can be computed by the method presented in Sect. 3.7.2.

7.1.3 Connections Extended by Alternatives (*CEA*)

State-of-the-art traffic information systems do not provide a computation of complete connections with high probabilities of success. To overcome this problem, some passengers choose an attractive train connection—computed with no regard for the probability of success—and change to the next available alternative continuation in case that a transfer becomes infeasible due to delays.

We developed an approach that, in a sense, simulates the behavior of passengers who care about alternative continuations as described above. We denote it by *CEA*. Analogously to our optimal approach from Chapter 5, the approach *CEA* computes complete connections. It aims to compute optimal outcomes; however, it is not optimal

for every query. The approach *CEA* does not compute its outcomes using our recursive functions from Sect. 5.3–5.4 but it follows another strategy as follows.

The approach *CEA* consists of three consecutive steps.

1. In the first step, for a query q , the approach *CEA* computes the set C of all Pareto optimal train connections (cf. Sect. 2.2) from $dep_st(q)$ to $dest_st(q)$ that arrive at $dest_st(q)$ within the time interval $[deadline(q) - \delta, deadline(q)]$. The value δ is a parameter of our experimental setting and will be defined in Sect. 9.4.2. The criteria optimized by the Pareto search are travel time and number of transfers.
2. In the second step, from each train connection $c \in C$, the approach *CEA* builds a complete connection c' and computes the probability of the success of c' . This step will be detailed below (cf. *Building complete connections*).
3. In the third step, among all *feasible* complete connections built, *CEA* finally delivers the complete connection with the latest departure time at $dep_st(q)$. From Sect. 4.3.2, recall that a complete connection c' is called *feasible* if $\rho(c') \geq \rho_{req}(q)$. If all complete connections built are infeasible, the outcome of *CEA* is *void*.

Building complete connections Given a train connection c , the approach *CEA* builds a complete connection c' as follows. In short, we extend the train connection c by alternative continuations to the destination station $dest_st(q)$ of the query q . The alternative continuations start at the stations where planned transfers could break. In the same way, alternative continuations are recursively extended by alternative continuations. A more precise definition of *CEA* follows.

As introduced in Sect. 1.2, let

$$(e_{dep}^1, e_{arr}^1), (e_{dep}^2, e_{arr}^2), \dots, (e_{dep}^i, e_{arr}^i), \dots, (e_{dep}^n, e_{arr}^n)$$

denote the ordered sequence of all train moves in the train connection c . We initially start with an empty complete connection c' . The initial departure event of c' (cf. Sect. 5.1.1) is set to the departure event e_{dep}^1 of c . We then set the instructions to continue the journey (cf. Sect. 5.1.2) as follows.

- For each arrival event e_{arr}^i where $(e_{arr}^i, e_{dep}^{i+1})$ is a dwell activity, we set the value $\hat{e}_{dep}^{cont}(e_{arr}^i, t)$ to the departure event e_{dep}^{i+1} for each time $t \in \text{supp}(\phi(e_{arr}^i))$.
- For each arrival event e_{arr}^i where $(e_{arr}^i, e_{dep}^{i+1})$ is a transfer activity, we set the value $\hat{e}_{dep}^{cont}(e_{arr}^i, t)$ to the departure event e_{dep}^{i+1} for each time $t \in \text{supp}(\phi(e_{arr}^i))$ that

satisfies the condition

$$t + \text{min_dur}(e_{arr}^i, e_{dep}^{i+1}) \leq \text{sched}(e_{dep}^{i+1}) + \text{max_wait}(e_{arr}^i, e_{dep}^{i+1}). \quad (7.1.1)$$

This definition conforms to the definition of the set of successor events which is used in the recursive function of our optimal approach (cf. Sect. 5.3.1).

After the above step, if there is no transfer activity in c , the procedure of building the complete connection c' is finished. Otherwise, there could exist arrival events e_{arr} in the complete connection c' where the value $\hat{e}_{dep}^{cont}(e_{arr}, t)$ is yet not defined for some $t \in \text{supp}(\phi(e_{arr}))$. Thus, we search for alternative continuations that can be used in such situations as follows.

As long as in c' there is at least one arrival event e_{arr} at some station $s \in S$ where the value $\hat{e}_{dep}^{cont}(e_{arr}, t)$ is yet not defined for some $t \in \text{supp}(\phi(e_{arr}))$, we add to c' an alternative continuation (cf. Sect. 4.1) from s to $\text{dest_st}(q)$ that can be taken if e_{arr} takes place at the time t . Let e_{dep} denote the very first departure event of this alternative continuation. We require that the condition from Eq. 7.1.1 is satisfied for e_{arr} , e_{dep} , and t . We then set the value $\hat{e}_{dep}^{cont}(e_{arr}, t')$ to e_{dep} for each time $t' \in \text{supp}(\phi(e_{arr}))$ where

$$t \leq t' \leq (\text{sched}(e_{dep}) + \text{max_wait}(e_{arr}, e_{dep})) - \text{min_dur}(e_{arr}, e_{dep}).$$

The alternative continuations are found using on-trip searches as discussed in Sect. 2.2. Among all possible alternative continuations to $\text{dest_st}(q)$ that can be taken if e_{arr} takes place at the time t , we choose the one with the earliest arrival time at $\text{dest_st}(q)$; by this choice, we simulate the behavior of passengers who after a connection break try to reach the destination as early as possible.

Probability of success Once a train connection is extended to a complete connection as described above, the probability of the success of the complete connection can be computed. Recall that the value $\hat{e}_{dep}^{cont}(e_{arr}, t)$ is set for each arrival event e_{arr} in the complete connection and each time $t \in \text{supp}(\phi(e_{arr}))$. We compute the probability of the success of such a given complete connection as discussed in Sect. 6.2.2.1. The result is the probability to reach $\text{dest_st}(q)$ no later than $\text{deadline}(q)$ when following the given complete connection.

7.2 Reliability as Pareto Criterion

Within the scope of the bachelor's thesis by Marco Plaue [Pla16], we developed a further approach to the search for reliable train connections. In this section, we conceptually describe this approach; details can be found in [Pla16]. The approach may be attractive

if reliability should be optimized as a criterion beside other criteria such as travel time and number of transfers. It is based on the Pareto Dijkstra algorithm.

The approach is based on a definition of *reliability* which is explained in Sect. 7.2.1. The search algorithm is detailed in Sect. 7.2.2.

7.2.1 Reliability Assessment

The reliability of a train connection is assessed based on the functions $\phi(e_{arr})$, $e_{arr} \in E_{arr}$, describing the random times of the arrival events in the timetable (cf. Sect. 3.3). For each transfer activity $(e_{arr}, e_{dep}) \in TRANS$, a value $unreliability(e_{arr}, e_{dep})$ is introduced that describes the *unreliability* of the transfer activity; it is defined as follows:

$$\begin{aligned} unreliability(e_{arr}, e_{dep}) &= P(X_{e_{arr}} + min_dur(e_{arr}, e_{dep}) > sched(e_{dep})) \\ &= \sum_{\substack{t \in supp(\phi(e_{arr})) \\ t + min_dur(e_{arr}, e_{dep}) > sched(e_{dep})}} \phi(e_{arr}, t). \end{aligned}$$

The definition of unreliability is then extended to train connections. For a train connection c with $k \in \mathbb{N}_0$ transfers, let $(e_{arr}^j, e_{dep}^j) \in TRANS$, $j = 1, \dots, k$, denote the transfer activities in c . The *unreliability* of the train connection c equals

$$unreliability(c) = \sum_{j=1, \dots, k} unreliability(e_{arr}^j, e_{dep}^j).$$

Hence, given two train connections c_1 and c_2 where $unreliability(c_1) > unreliability(c_2)$, we assume that c_2 is more reliable than c_1 .

Compared to our reliability assessment method from Sect. 3.7, the method from this section disregards some factors influencing the reliability (such as the waiting times of the trains for each other). However, the evaluation of the method demonstrated that it is still realistic [Pla16]; the greater the unreliability value is, the higher the probability of a connection break is. The advantage of this method is in its simplicity; compared to the method from Sect. 3.7, it is faster, and less computations are required.

7.2.2 The Search Algorithm

A query is as defined in Sect. 2.2; the outcome is the set of all Pareto optimal train connections for the query. An arbitrary set of Pareto criteria can be selected to be optimized. The search approach is an extension of the Pareto Dijkstra algorithm (cf. Sect. 2.2.2) applied on a time-dependent graph (cf. Sect. 2.1.2). Compared to the basic version of Pareto Dijkstra from 2.2, there are three main modifications: *unreliability criterion*, *adjustment of the domination rules*, and *expansion of the search space*; these

are detailed in the following.

Unreliability criterion The *reliability* of the train connections is optimized as an additional Pareto criterion. More precisely, we minimize the unreliability value introduced in Sect. 7.2.1. Recall that each search label (cf. Sect. 2.2.2, *Labels*) represents a train connection from the departure station of the query to the station where the label is created. Each search label is extended by the unreliability value of the train connection that is represented by the label. For start labels, this value equals zero. For labels created during the search, this value is set as follows.

Recall the creation of the labels during the search (cf. Sect. 2.2.2, *Creating labels during the search*); given an existing label \hat{l} at a node $v \in V$ and an outgoing edge $(v, w) \in A$ of v , a new label l is created at the node w . Whenever taking the edge (v, w) requires a new transfer in the train connection, first, the unreliability of this transfer is assessed by the approach from Sect. 7.2.1. The unreliability value of l is then set to the unreliability value of \hat{l} plus the unreliability value of the transfer activity. On the other hand, if taking (v, w) does not result in a new transfer activity, the unreliability value of l is set to the unreliability value of \hat{l} .

Adjustment of the domination rules According to the Pareto dominance rules from Sect. 2.2.1, a label l cannot dominate another label l' if the unreliability value of l is greater than the unreliability value of l' . Additionally, we must adjust the domination rules as follows in order to find optimal train connections.

Let l and l' denote two labels created at the same node $v \in V$ at a station $s \in S$. Let the unreliability value of l be lower than or equal to the unreliability value of l' . In words, according to the transfer activities up to s , the train connection represented by l is at least as reliable as the train connection represented by l' . Now, let e_{arr} and e'_{arr} denote the arrival events at the station s of each of these train connections respectively. We forbid l to dominate l' if

$$\exists t \in TIMES : P(X_{e_{arr}} \leq t) < P(X_{e'_{arr}} \leq t).$$

The reason is as follows: according to the functions $\phi(e_{arr})$ and $\phi(e'_{arr})$, a traveler taking the train connection represented by l' could reach the station s earlier. Hence, a transfer at s after the label l' could be more reliable compared to the label l .

Expansion of the search space From Sect. 2.1.2 (cf. *Routes*), recall that the time-dependent graph model is based on the following assumption for the computation of attractive train connections: after the arrival at each station, it is sufficient to consider solely the very next possible departure on each route at that station. This assumption

does not hold when the reliability is optimized as a Pareto criterion. While the transfer into the very next departing train on a route could be highly unreliable, taking the subsequent departure could result in the most reliable train connection. We consequently must expand the search space as follows.

From Sect. 2.1.2, recall that each route edge represents a set of train moves. Given a label l at a node v , the Pareto Dijkstra algorithm creates new labels for each outgoing route edge (v, w) ; these are successor labels of l . Let e_{arr} denote the arrival event at the station of v of the train connection represented by l . Moreover, for a route edge (v, w) , let $e_{dep}^1, e_{dep}^2, \dots, e_{dep}^n$ denote the successive departures of the train moves represented by (v, w) .

In the basic variant of Pareto Dijkstra, we create a new label for the very next departure of the train move represented by (v, w) . From the set of departure events specified above, let e_{dep}^i denote this very next departure event where $i \in \{1, 2, \dots, n\}$. If reliability is an additional Pareto criterion, we must create a new label for each of the departure events $e_{dep}^i, e_{dep}^{i+1}, \dots, e_{dep}^j$ where e_{dep}^j is the earliest departure event such that

$$\max\{t \in \text{supp}(\phi(e_{arr}))\} + \text{min_dur}(e_{arr}, e_{dep}^j) \leq \text{sched}(e_{dep}^j).$$

This concludes the description of the approach; more details can be found in [Pla16].

7.3 Conclusion

In this chapter, we discussed alternative approaches to the search for reliable train connections and complete connections. The approaches from Sect. 7.1 aim to solve the problem from Sect. 4.3.3.1 (cf. *Latest Departure Subject to Arrival Guarantee*). In our computational study in Sect. 9.4, these approaches are compared to our optimal approach from Chapter 5.

The approach from Sect. 7.2 is an extension of the Pareto Dijkstra algorithm. It may be attractive if reliability should be optimized as a criterion beside other criteria such as travel time and number of transfers. It could be combined with the approach presented in Sect. 8.3.3 in order to compute reliable intermodal connections.

Chapter 8

Reliable Intermodal Connections

In the previous chapters, we addressed the reliability assessment for train connections and the computation of reliable complete connections. In this chapter, we first extend our reliability assessment to intermodal connections. As discussed in Sect. 1.4, intermodal connections consist of public transport but also of parts where individual modes of transport are used. More precisely, we focus on individual modes of transport for which the availability is not guaranteed. Examples include car sharing and bike sharing services since there is no guarantee they will be available when required. We predict the availability of cars and bikes by evaluating historical availability data concerning car and bike sharing stations. This method helps us to assess the reliability of intermodal connections. We then discuss approaches to searching for reliable intermodal connections. Our main contribution is the integration of the reliability assessments into the search for intermodal connections.

After that, we address the problem of connection breaks late at night, particularly for the case in which a destination cannot be reached prior to the end of operations. We propose a search approach that finds intermodal connections with taxi rides to the destination or overnight stays in hotels. Our approach aims to satisfy the passengers and to minimize the costs for the railway company.

Overview In Sect. 8.1, we discuss the state-of-the-art. Our reliability assessment for intermodal connections is presented in Sect. 8.2. Then, in Sect. 8.3, we discuss the search for reliable intermodal connections. Last, in Sect. 8.4, we address the problem of connection breaks late at night.

Joint contribution The approaches presented in Sect. 8.2 and 8.3 are joint contributions of Felix Gündling and the author of this work. While Felix Gündling is mainly focused on intermodal train connections, the focus of this work is on reliability and its integration into the search for intermodal connections. Moreover, we sincerely

thank Sebastian Fahnenschreiber who supported our research on reliable intermodal connections with helpful hints and suggestions as well as code implementations.

8.1 Related Work

As mentioned in Sect. 1.4, intermodal connections have been addressed by [Gün15, Gün14, Arn14] in the research group Algorithm Engineering at the Technische Universität Darmstadt, Germany. Our research is based on the approaches introduced by them, and our implementations were extensions of their code base.

To the best of our knowledge, the problems of connection breaks late at night and finding reliable intermodal connections have not been addressed so far in the literature. There are various works which focus on the reliability of individual modes of transport, for example, predictions for the availability of parking spaces [CBM12]. However, reliable intermodal connections—which are a combination of reliable rides in public transport and reliable rides with individual modes of transport—have not been considered so far.

8.2 Reliability Assessment

In this section, we present a method of assessing the reliability of intermodal connections. As discussed in Sect. 1.4, each intermodal connection consists of an initial and an end part where individual modes of transport are used. In the middle, there is a train connection as defined in Sect. 1.2. Hence, for this public transport part, we can assess the reliability exactly as introduced in Sect. 3.7. The reliability assessment for the initial part and the end part is discussed in the following.

In this work, we assume that individual modes of transport such as walking, individual cars, individual bikes, and taxis are reliable. In contrast, for station-based car and bike sharing (cf. Sect. 2.3.1), there is no guarantee that at the respective station a car or a bike actually will be available. While passengers desire to plan their journeys beforehand, the availabilities of car sharing and bike sharing services are usually known for the current situation solely. There is no guarantee that there will be a car or a bike that can be used at the time when the passenger arrives at the car sharing or bike sharing station respectively.

Note: A similar problem occurs when using an individual car and parking spaces close to train stations have limited capacities. There is no guarantee that the passenger finds a parking close to the train station. The solution which we propose for car sharing and bike sharing is also applicable in order to predict the availability of parking areas.

In the following, for the sake of convenience, we will focus on bike sharing but the approaches presented here are also applicable to car sharing. First, in Sect. 8.2.1, we explain how to retrieve predictions for the availabilities of the bikes. Then, in Sect. 8.2.2, we explain how the reliability of an intermodal connection can be assessed based on the predictions.

8.2.1 Predictions Based on Historical Data

We predict the availability of an individual mode of transport by evaluating historical availability data. In this work, we assume that the day of week and the time of day are two factors influencing the bike availabilities. Given a bike sharing station bst , a day of week d , and an hour of day h , the *prediction* for the availability is a number $n \in \mathbb{N}$. In words, we predict that n bikes are available at the station bst on the day of week d and at the hour of day h .

The predictions are computed as follows. Usually, bike sharing providers have interfaces which can be used in order to retrieve the current availability state [Arn14]. We gather availability data about the bike sharing stations over a sufficient time period. By evaluating this data, we compute predictions for the availability of the bikes. In the following, we elucidate these procedure.

First step: gathering availability data We first need to gather data about the availabilities of the bikes. For this, we gather availability data by sending requests to the interfaces of the bike sharing providers periodically; this procedure is repeated over a sufficient time period (three months by way of example). The response to such a request usually contains the number of available bikes at each station. Formally, for each bike sharing station bst of a bike sharing provider, we retrieve a data tuple (bst, t, n) where bst is the respective bike sharing station, t is time at which the availability is queried, and $n \in \mathbb{N}$ is the number of bikes available at bst at the time t .

Second step: classifying gathered data We then classify the gathered data according to factors influencing the bike availabilities; we assume that the bike availabilities depend on the day of week and the hour of day. Therefore, we associate the data tuples (bst, t, n) to new data tuples (bst, d, h) where $d \in \{ Mon, Tue, Wed, Thu, Fri, Sat, Sun \}$ and $h \in \{0, 1, 2, \dots, 23\}$ are the day of week and the hour of the day respectively. More precisely, each data tuple (bst, t, n) is associated to the data tuple (bst, d, h) where the point in time t is at the day of week d and at the h -th hour of day.

Third step: predictions After the classification in the second step, for each bike sharing station bst , each day of week d , and each hour of day h , we have a set of

availability values: the values n from the data tuples (bst, t, n) . Let the set $N(bst, d, h)$ comprise all availability values for bst , day d and hour h . In order to obtain a prediction for the bike availability at a station bst , on a day d , and at the h -th hour of day, different aggregators can be applied as follows.

- *Average*: We can compute the average over all values in $N(bst, d, h)$. By way of example, we could make a point that between eight o'clock and nine o'clock on Mondays, there are five bikes on average at bst .
- *Minimum*: Computing the minimum over all values in $N(bst, d, h)$ would result in a lower bound for the availability. We believe that this aggregator is too conservative. As soon as there is one data tuple (bst, t, n) where n equals zero, the minimum value would be zero.
- *Quantile*: We could compute various quantiles over the set $N(bst, d, h)$. This would result in statements such as between eight o'clock and nine o'clock on Mondays, at bst , four bikes can be found with a 90% guarantee.

Depending on the individual mode of transport and the gathered data, a suitable aggregator should be selected. One could evaluate different aggregators in order to find out by which of them realistic predictions can be obtained. This concludes our method of computing predictions for bike availabilities. In Sect. 8.2.2, we discuss how the reliability of an intermodal connection can be assessed based on the predictions from this section.

8.2.2 Reliability Assessment Based on the Predictions

Given an intermodal connection c , we assess the reliability of the initial and the end part of c based on the predictions from Sect. 8.2.1. From Sect. 1.4 and Fig. 1.4 (on page 12), recall that, in the station-based bike sharing model, the passenger collects a bike from a bike sharing station and returns it at another bike sharing station. The station at which the bike is returned has no influence on the reliability. In contrast, the availability of a bike at the station where a bike is collected is crucial; thus, the assessment is derived from the prediction for that station.

First, for the initial part as well as the end part of the intermodal connection c where bike sharing is used, we retrieve the prediction for the bike sharing station where a bike is collected. For the predictions from Sect. 8.2.1, we require the day of week and the hour of day; both can be retrieved from the intermodal connection c (cf. Sect. 1.4, *Intermodal connections*).

Based on the predictions, we then assess the reliability of the individual and the end part of c . We discuss two assessment methods: *binary classification* and *multiclass*

classification; these are as follows.

Binary classification: reliable or unreliable Let bst_1 and bst_2 denote the bike sharing stations where the bikes are collected for the initial and the end part of the intermodal connection c respectively. Moreover, let $n_1, n_2 \in \mathbb{N}$ denote the predictions for the availabilities of bikes at bst_1 and bst_2 respectively.

We define a threshold $\tau \in \mathbb{N}$. The initial part of c is classified as *reliable* if $n_1 \geq \tau$ and as *unreliable* otherwise. Analogously, the end part of c is classified as *reliable* if $n_2 \geq \tau$ and as *unreliable* otherwise.

Multiclass classification We define $m \in \mathbb{N}$ thresholds $\tau_1, \tau_2, \dots, \tau_m$ such that $\tau_j < \tau_{j+1}$ for $j = 1, \dots, m - 1$. By these m thresholds, we introduce a classification into $m + 1$ classes; the bike sharing stations associated to the classes 1 and $m + 1$ have the lowest and the highest reliabilities respectively. Let bst_1, bst_2, n_1 , and n_2 be defined as above. The bike sharing stations bst_1 is associated to a class $class \in \{1, \dots, m + 1\}$ where $class$ is determined as follows:

$$class = \begin{cases} 1 & \text{if } n_1 < \tau_1 \\ k \in \{2, \dots, m - 1\} & \text{if } n_1 \in [\tau_k, \tau_{k+1}) \\ m + 1 & \text{if } n_1 \geq \tau_m. \end{cases}$$

The station bst_2 is classified analogously.

Discussion While the binary classification is simpler, the multiclass classification makes possible a differentiated view on the bike availabilities. The binary classification could be used if unreliable stations should be identified by a hard boundary (the threshold τ). The multiclass classification could be used to obtain a multilevel assessment; for example it is suitable if reliability is optimized as a Pareto criterion. In the next section, we will discuss different approaches to the search for reliable intermodal connections.

For different modes of individual transport, different thresholds need to be defined. The thresholds should be adjusted to the respective individual mode of transport. For example, when assuming that bikes are booked more frequently than cars, a specific number of available cars could indicate a reliable station while the same number of available bikes indicates an unreliable station. Generally, we could evaluate the supply-demand balance at each station in order to obtain more accurate reliability assessments.

8.3 The Search for Reliable Intermodal Connections

In this section, we discuss different approaches to the search for *reliable* intermodal connections and *reliable intermodal complete connections*. Intermodal connections have been discussed in Sect. 1.4; intermodal complete connections will be introduced below.

Queries A query q comprises a starting location and a destination location as well as a time interval $[begin(q), end(q)]$ for the start of the journey from the starting location (for sake of convenience, we do not discuss the on-trip scenario here). The passenger seeks for a reliable journey from the starting location to the destination location that starts in the time interval $[begin(q), end(q)]$.

The search approaches Given a query, we first retrieve the routes for individual modes of transport; these are denoted as *individual routes* (cf. Sect. 2.3.1). We then assess the reliability of each individual route. These two steps (retrieve and assess) are explained in Sect. 8.3.1; they are the prerequisite for all search approaches that we discuss here. The search approaches are the following.

1. *Finding reliable intermodal connections:* From Sect. 7.1.2 and 7.2, recall the approaches that compute reliable train connections based on Pareto Dijkstra. We discuss two methods of extending a search approach, such as those from Sect. 7.1.2 and 7.2, to the computation of reliable intermodal connections:
 - (a) We solely use rides by individual modes of transport that are classified as *reliable* by the binary classification from Sect. 8.2.2; this approach is discussed in Sect. 8.3.2.
 - (b) We use the multiclass classification from Sect. 8.2.2, and optimize the reliability of the initial and end part of the intermodal connections as an additional Pareto criterion; this approach is discussed in Sect. 8.3.3.
2. *Finding reliable intermodal complete connections:* An intermodal complete connection is a complete connection (cf. Sect. 4.3.2) that is extended by an initial part as well as end parts where individual transport is used. The initial part connects the starting location to a public transport station at which the journey by public transport starts. Each of the alternative continuations of the intermodal complete connection has an end part where individual modes of transport are used in order to reach the destination location. In Sect. 8.3.4, we propose an approach to computing reliable intermodal complete connections.

For all approaches, we use the time-dependent graph model from Sect. 2.1.2. Moreover, for all approaches, we require the step to retrieve the *individual routes* and their reliability assessments; this common step is explained in Sect. 8.3.1.

Note: We implemented the approaches from Sect. 8.3.2 and 8.3.4; sample results can be found in Appx. A.2. The approach from Sect. 8.3.3 is only discussed conceptually. In this work, we do not evaluate the approaches to computing reliable intermodal connections and intermodal complete connections.

8.3.1 Retrieving Individual Routes and Their Assessments

As announced, the first step of the search for reliable intermodal connections or complete connections is retrieving *individual routes* and assessing their reliabilities. The individual routes are computed and retrieved as described in Sect. 2.3.1. Given a query q , let IR_{init} and IR_{end} respectively denote the sets of all individual routes computed for the initial parts and the end parts of the individual connections for q .

Assessment tuples We assess the reliability of each individual route in IR_{init} and IR_{end} by the method from Sect. 8.2.2. For the approaches from Sect. 8.3.2 and 8.3.4, we apply the binary classification from Sect. 8.2.2. On the other side, for the approach from Sect. 8.3.3, we apply the multiclass classification from Sect. 8.2.2. The result is a set of tuples for each individual route where each tuple (t_1, t_2, a) comprises the reliability assessment a that is valid from $t_1 \in TIMES$ to $t_2 \in TIMES$. In the following, we detail the procedure of retrieving these tuples.

Note: As explained in Sect. 8.2, we assume that individual modes of transport such as walking, individual cars, individual bikes, and taxis are reliable. Hence, for the individual routes of such modes, an assessment is not necessary.

The assessment procedure From Sect. 8.2.1, recall that the reliability assessments depend on the day of week and the hour of day. Thus, for each individual route, the reliability needs to be assessed for each time at which the individual route could be used in the intermodal connection. Recall that the user provides a time interval $[begin(q), end(q)]$ for the start of the journey at the starting location. The individual routes in IR_{init} are consequently used in the time interval $[begin(q), end(q)]$. Thus, the reliabilities of the individual routes in IR_{init} need to be assessed for the time interval $[begin(q), end(q)]$.

On the other hand, for the individual routes in IR_{end} , we assess the reliability of each individual route in IR_{end} for the time interval $[begin(q), begin(q) + t_{max}]$ where t_{max} is chosen sufficiently large; for example, twenty-four hours if we limit the maximum

duration of intermodal connections to twenty-four hours. A smaller time interval cannot be defined since, prior to the search, the time is unknown at which the individual routes for the end parts of intermodal connections are used.

Subsequently, the parameters day of week and hour of day are derived from the intervals $[begin(q), end(q)]$ and $[begin(q), begin(q) + t_{max}]$ for the individual routes in IR_{init} and IR_{end} respectively. These time intervals potentially overlap several hours. Thus, as described below (cf. *Splitting a time interval*), we first split each of them into subintervals of one hour length (the very first and the very last subinterval can be shorter). For each individual route in IR_{init} , we assess the reliability of the individual route for each subinterval derived from $[begin(q), end(q)]$. Analogously, for each individual route in IR_{end} , we assess the reliability of the individual route for each subinterval derived from $[begin(q), begin(q) + t_{max}]$. Finally, as announced before, for each individual route, we obtain a set of assessment tuples where each assessment tuple (t_1, t_2, a) comprises a reliability assessment a for the time interval $[t_1, t_2]$.

Splitting a time interval Basically, we split a time interval $[t_{begin}, t_{end}]$ into $n + 2$ subintervals

$$\begin{aligned}
 & [t_{begin} , t'_{begin}), \\
 & [t'_{begin} , t'_{begin} + 60\text{min}), \\
 & [t'_{begin} + 60\text{min} , t'_{begin} + 120\text{min}), \\
 & \quad \dots , \\
 & [t'_{begin} + i \cdot 60\text{min} , t'_{begin} + (i + 1) \cdot 60\text{min}), \\
 & \quad \dots , \\
 & [t'_{begin} + (n - 1) \cdot 60\text{min} , t'_{begin} + n \cdot 60\text{min}), \\
 & [t'_{begin} + n \cdot 60\text{min} , t_{end}]
 \end{aligned}$$

where $n \in \mathbb{N}_0$, $i = 0, 1, \dots, n - 1$, and the time t'_{begin} is the next full hour after t_{begin} . We split the intervals $[begin(q), end(q)]$ and $[begin(q), begin(q) + t_{max}]$ into subintervals as described for $[t_{begin}, t_{end}]$. The parameters day of week and hour of day—required for the reliability assessment—are then derived from these subintervals; note that the length of each subinterval is an hour.

Example: Let t_0 denote the midnight of a specific day. We split the time interval $[t_0 + 30, t_0 + 210]$ into the subintervals $[t_0 + 30, t_0 + 60)$, $[t_0 + 60, t_0 + 120)$, $[t_0 + 120, t_0 + 180)$, $[t_0 + 180, t_0 + 210]$.

8.3.2 Excluding Unreliable Rides

In this section, we discuss an approach that delivers intermodal connections with reliable initial and end parts solely. It can be combined with the approaches from Sect. 7.1.2 and 7.2. The approach consists of three steps.

1. First, as described in Sect. 8.3.1, we retrieve the individual routes and their assessments tuples. For the search approach presented in this section, the assessment a in each assessment tuple (t_1, t_2, a) is a binary classification such that $a \in \{reliable, unreliable\}$.
2. Second, the individual routes are incorporated into the search for reliable intermodal connections as will be explained below (cf. *Modeling the initial part* and *Modeling the end part*). An individual route is used in an intermodal connection solely in the time intervals in which it is classified as *reliable*.
3. Finally, we perform the Pareto Dijkstra algorithm (cf. Sect. 2.2) in order to find all Pareto optimal intermodal connections from the starting location to the destination location. The intermodal connections computed consist of reliable individual parts solely.

The first and the third step are self-explaining; we now detail the second step. Individual routes of individual modes of transport which are assumed to be reliable (such as taxi, individual car, and individual bike) are incorporated into the search as described in Sect. 2.3. On the other side, for individual modes as bike sharing and car sharing, we take into account the reliability assessment as follows. From Fig. 1.4 (on page 12), recall the structure of a station-based individual route; it consists of a walk, a ride, and a walk again. In the following, let δ_{w1} , δ_r , and δ_{w2} denote the durations of each of these sections respectively.

Modeling the initial part As described in Sect. 2.3.3, individual routes for the initial parts of intermodal connections are modeled by creating respective start labels. Recall that each individual route for the initial part connects the starting location to a public transport station. For each individual route ir for the initial part, we create a new start label for each departure event $e_{dep} \in E_{dep}$ at the destination station of ir where

- $sched(e_{dep}) - (\delta_{w1} + \delta_r + \delta_{w2}) \in [begin(q), end(q)]$ and
- there is an assessment tuple (t_1, t_2, a) of ir such that $a = reliable$ as well as $sched(e_{dep}) - (\delta_r + \delta_{w2}) \in [t_1, t_2)$. The latter condition ensures that the individual mode of transport (as bike sharing or car sharing) is used at the time when it is classified as reliable.

The new label is associated to the route node that represents e_{dep} (cf. Sect. 2.1.2), and the timestamp of the new label is set to $sched(e_{dep})$. The time cost of the start label is set to $\delta_{w1} + \delta_r + \delta_{w2}$.

Modeling the end part As described in Sect. 2.3.2, in the basic approach to the search for intermodal connections, individual routes for the end parts of intermodal connections are modeled by *individual transport edges*. For each query, these edges are added to the time-dependent graph that represents the timetable; after the query is processed, they are removed. The individual transport edges connect station nodes to the destination node; the latter represents the destination location.

In order to search for *reliable* intermodal connections, we introduce *time-dependent* individual transport edges. Analogously to the individual transport edges from Sect. 2.3.2, a *time-dependent* individual transport edge connects the public transport station—at which the individual route starts—to the destination node. We create a time-dependent individual transport edge for each individual route and each of its assessment tuples (t_1, t_2, a) where $a = \text{reliable}$. The individual transport edge additionally stores the time interval $[t_1, t_2)$. Analogously to route edges (cf. Sect. 2.1.2, *Edge costs*), the duration of a time-dependent individual transport edge depends on the time it is used (cf. Sect. 2.2.1, *Creating labels during the search*). More precisely, given a time $t \in \text{TIMES}$, the length of an individual route edge equals

$$length = \begin{cases} +\infty & \text{if } t + \delta_{w1} \geq t_2 \\ \delta_{w1} + \delta_r + \delta_{w2} & \text{if } t + \delta_{w1} \in [t_1, t_2) \\ t_1 - t + \delta_r + \delta_{w2} & \text{if } t + \delta_{w1} < t_1 \end{cases}$$

where t is the time at which the individual route edge is used. The time difference $t_1 - t$ includes δ_{w1} as well as the waiting time for an available bike—a bike is assumed to be available no earlier than t_1 . The values δ_{w1} , δ_r , and δ_{w2} have been defined at the beginning of Sect. 8.3.2.

Discussion After Pareto Dijkstra is performed (with the desired criteria), we retrieve Pareto optimal intermodal connections which use reliable individual routes solely. As announced before, to obtain reliable train connections for the middle parts of the intermodal connections, we could apply approaches as discussed in Sect. 7.1.2 and 7.2. In the next sections, we discuss further search approaches.

8.3.3 Reliability as Pareto Criterion

In this section, we discuss an approach that optimizes the reliability of the initial and end parts of intermodal connections as an additional Pareto criterion. It can be combined with the approaches from Sect. 7.1.2 and 7.2. The approach consists of three steps.

1. First, as described in Sect. 8.3.1, we retrieve the individual routes and their assessments tuples. For the search approach presented in this section, the assessment a in each assessment tuple (t_1, t_2, a) is a multiclass classification as described in Sect. 8.2.2.
2. Second, the individual routes are incorporated into the search for reliable intermodal connections as will be explained below (cf. *Modeling the initial part* and *Modeling the end part*).
3. Finally, we perform the Pareto Dijkstra algorithm (cf. Sect. 2.2) to find all Pareto optimal intermodal connections from the starting location to the destination location; the reliability of the initial and end parts is optimized as an additional Pareto criterion as will be discussed below (cf. *Pareto criterion for reliability*).

Individual routes of individual modes of transport which are assumed to be reliable (as taxi, individual car, and individual bike) are incorporated into the search as described in Sect. 2.3. On the other side, for individual modes as bike sharing and car sharing, we take into account the reliability assessments as follows. From Fig. 1.4 (on page 12), recall the structure of a station-based individual route; it consists of a walk, a ride, and a walk again. In the following, let δ_{w1} , δ_r , and δ_{w2} denote the durations of each of these sections respectively.

Modeling the initial part As described in Sect. 2.3.3, individual routes for the initial parts of intermodal connections are modeled by creating respective start labels. For each reliable individual route ir and each of its assessment tuples (t_1, t_2, a) , we create a new start label for each departure event $e_{dep} \in E_{dep}$ at the destination station of ir where

- $sched(e_{dep}) - (\delta_{w1} + \delta_r + \delta_{w2}) \in [begin(q), end(q)]$, and
- $sched(e_{dep}) - (\delta_r + \delta_{w2}) \in [t_1, t_2]$.

The new label stores the reliability assessment a from the assessment tuple. It is associated to the route node that represents e_{dep} (cf. Sect. 2.1.2), and the timestamp of the new label is set to $sched(e_{dep})$. The time cost of the start label is set to $\delta_{w1} + \delta_r + \delta_{w2}$.

Modeling the end part Analogously to the search approach from Sect. 8.3.2 (cf. *Excluding Unreliable Rides*), we create *time-dependent* individual transport edges. We create such an edge for each individual route and each of its assessment tuples (t_1, t_2, a) . The edge additionally stores the time interval $[t_1, t_2)$ as well as the assessment a for the individual route. The durations of the time-dependent individual transport edges are defined as in Sect. 8.3.2.

Pareto criterion for reliability We introduce a new Pareto criterion for the Pareto Dijkstra algorithm (cf. Sect. 2.2); it models the reliability of the rides by individual modes of transport. From Sect. 8.2.2, recall that, by the multiclass classification, we assign to each individual route a specific class where the lowest and the highest class refer to the lowest and highest reliability assessment respectively. We optimize this reliability class as a Pareto criterion. Consequently, during the search, a label with the reliability class rc cannot dominate another label with the reliability class rc' if $rc > rc'$ (cf. Sect. 2.2.1, *Pareto Dominance and Pareto Optimality*).

Since each intermodal connection has two parts where individual modes of transport are used, it has up to two reliability assessments for those parts. Each label could store either the minimum or the average of these both assessments. Another solution is to use two Pareto criteria; one for the initial part and one for the end part of the intermodal connections. However, this would result in a higher number of Pareto optimal intermodal connections. We explain these different methods by taking the following example illustrated in Table 8.1 (on page 143); the explanations follow here.

Let A , B , Y , and Z be different rides by individual modes of transport where A as well as B can be used as initial parts, and Y as well as Z can be used as end parts of intermodal connections. Let 0 and 10 be the lowest and the highest reliability classes respectively. The reliability assessments for A , B , Y , and Z are 0, 4, 10, and 4 respectively. We assume that there is only one Pareto optimal train connection—the middle part of the intermodal connections—that can be combined with A or B as the initial part and with Y or Z as the end part. This results in four possible intermodal connections, denoted by AY , AZ , BY , and BZ ; these use the initial and end parts as follows: A and Y , A and Z , B and Y , and B and Z respectively.

If we use the minimum value of the reliability classes in order to find the Pareto optimal intermodal connections, the assessments for the intermodal connections would be as follows: AY : 0, AZ : 0, BY : 4, and BZ : 4. The intermodal connections which use A would be dominated. However, the superiority of Y over Z ($10 > 4$) has no effect on the result; BY and BZ are assumed to be equally reliable.

Building the average value would result in the following assessments: AY : 5, AZ : 2, BY : 7, and BZ : 4. We would correctly identify that BY is the most reliable variant.

conn	assessment		min	avg	two criteria
	initial p.	end part			
<i>AY</i>	0	10	0	5	0, 10
<i>AZ</i>	0	4	0	2	0, 4
<i>BY</i>	4	10	4	7	4, 10
<i>BZ</i>	4	4	4	4	4, 4

Table 8.1: The table illustrates an example with four intermodal connections. We compare different variants modeling the reliability assessments of the individual routes. The values which are illustrated bold refer to the intermodal connections which are Pareto optimal according to the applied variant.

“min”: the minimum of the assessments for the initial and the end part is chosen for the Pareto criterion for reliability.

“avg”: the average over the assessments is built.

“two criteria”: two separate Pareto criteria are used for the two reliability assessments.

However, *AY* is assumed to be more reliable than *BZ* although *A* has a reliability assessment of 0. One could construct examples where building the average results in suboptimal results.

Introducing separate Pareto criteria for the initial and the end part would result in one Pareto optimal intermodal connection: the connection *BY*; it dominates *AY*, *AZ*, and *BZ*. We believe this variant is the most suitable; however, for example, if the reliability assessments were all the same, all intermodal connections specified above were Pareto optimal. Thus, this approach can result in higher numbers of Pareto optimal intermodal connections that appear in the result set.

Discussion After Pareto Dijkstra is performed (with the desired criteria), we retrieve Pareto optimal intermodal connections where the reliability of the individual routes is optimized as an additional criterion. As announced before, to obtain reliable train connections for the middle parts of the intermodal connections, we could apply approaches as discussed in Sect. 7.1.2 and 7.2. In the next section, we discuss an approach to the search for reliable intermodal complete connections.

8.3.4 Intermodal Connections Extended by Alternatives

At the beginning of Sect. 8.3, we defined intermodal complete connections. In this section, we extend the approach *connections extended by alternative continuations (CEA)* from Sect. 7.1.3 to intermodal complete connections. In order to retrieve intermodal complete connections with reliable initial and end parts, we first retrieve all individual routes and then exclude the unreliable ones as described in Sect. 8.3.2.

More precisely, for each query q , we first compute the set of all Pareto optimal intermodal connections from $dep_st(q)$ to $dest_st(q)$ that depart in $[begin(q), end(q)]$ as described in Sect. 8.3.2. For each of these intermodal connections, we build an

intermodal complete connection as explained in Sect. 7.1.3 (cf. *Building complete connections*). There, we incorporate into the search for alternative continuations the reliable individual routes from public stations to the destination location as described in Sect. 8.3.2. The result is a set of intermodal complete connections which consist of reliable rides by individual transport.

Since there is no deadline in the search scenario from this chapter, the probability of the success of such an intermodal complete connection equals 1. According to the *CEA* approach from Sect. 7.1.3, such an intermodal complete connection has an alternative continuation for every situation that could occur with respect to the probability density functions describing the random times of the departure and arrival events.

Conclusion This concludes the search for reliable intermodal complete connections. The search for intermodal complete connections which are optimal according to the problem from Sect. 4.3 is left to future research; we address this issue in Sect. 10.2.

8.4 Late-Night Connections

In this section, we address an important problem in public transportation. Assume a passenger takes a train connection but it breaks due to delays or other real-time incidents. The break is late at night such that the destination cannot be reached by public transport prior to the end of operations. In such a situation, the passenger has the following options:

- to wait at some station to bridge the nighttime,
- to travel by special night trains or night buses if available,
- to get a taxi transfer to the destination,
- to stay in a hotel overnight and to travel to the destination at the next day.

For the first two options, the passenger must travel during nighttime; usually, this is undesired by passengers. For the latter two options, we assume that the transportation company pays the costs; usually, the transportation companies decide on a case-by-case basis when taxi transfers or hotel stays are offered to passengers.

Delay compensation Since the original connection is broken, the passenger potentially reaches the destination later than planned. Transportation companies usually must pay a compensation for the delayed arrival. The amount depends on the delay and the ticket price of the original, broken train connection. According to the policies of Deutsche Bahn AG [Bah16b], we assume that a delay compensation is paid even in cases of taxi transfers and overnight stays at hotels.

Taxi transfers and hotel stays We assume that the transportation provider can organize taxi transfers to the destination; the passenger is collected at some public transport station. This station is not necessarily the station where the original train connection is broken. Instead, the passenger could be suggested to travel to a station that is closer to the destination by public transport. From there on, the passenger is brought to the destination by a taxi. This option would result in lower taxi costs.

Moreover, we assume that there are public transport stations close to hotels; the set of all these stations is provided by the transportation company; we denote this set by $S_H \subset S$. Some transportation companies—which offer hotel stays in case of late-night connection breaks—have contractual agreements with selected hotels.

Two parameters ensure reasonable hotel stays; *earliest check-out time* and *minimum stay duration*. These parameters can be specified by passengers. The passenger does not desire to leave the hotel too early in the morning; the former parameter models this constraint. The minimum stay duration is a lower bound for the time the passenger desires to spend in a hotel if a hotel stay is necessary. This time comprises the sleep time as well as the time required for check-in and check-out.

General solution: late-night connections We compute a set of *late-night connections* for the described scenario. A *late-night connection* models one of the travel options from the enumeration at the beginning of Sect. 8.4. A late-night connection always starts at the station where the original train connection of the passenger is broken; it always ends at the destination station. Since a late-night connection potentially has a taxi transfer to the destination, it can be intermodal. We believe that a taxi transfer to the destination is the only reasonable individual mode of transport for the scenario from this section. Thus, in contrast to intermodal connections as defined in Sect. 1.4, late-night connections have no initial part where individual transport is used.

Our approach is based on the Pareto Dijkstra algorithm (cf. Sect. 2.2). The Pareto Dijkstra algorithm is extended by a minimization of the travel time during nighttime and the costs to be paid by the transportation company. Transportation companies could use our approach in order to get a proper overview about the available options to accompany passengers after connection breaks late at night. They would still have the possibility to decide—on a case-by-case basis—which options are actually offered to the passengers. In Sect. 9.5, we will evaluate the approach from this section. We discuss our approach in the rest of this section.

8.4.1 The Search Approach

Queries A query comprises a departure station $dep_st(q)$ (the station at which the original train connection is broken), a destination station $dest_st(q)$, and a time

$ontrip.time(q)$ for the departure. The latter represents the on-trip scenario since, after the original train connection is broken, the passenger already waits at a station and asks for an option to reach the destination. Moreover, the query comprises the ticket price and the planned arrival time of the original, broken train connection; these are required to compute the compensation amount according to a delayed arrival at the destination. Finally, the query contains the minimum stay duration and the earliest check-out time for hotel stays as described in Sect. 8.4 (cf. *Taxi transfers and Hotel stays*).

The approach Given a query q , our approach computes a set of attractive late-night connections; the search approach consists of four steps.

1. We first retrieve a set of reasonable *taxi transfers* and their costs as will be discussed in Sect. 8.4.2 and 8.4.3.
2. We then retrieve the costs for hotel stays as will be discussed in Sect. 8.4.3; as already announced, we assume that there is a known set of stations at which there are hotels.
3. We extend the time-dependent graph—which represents the timetable—by additional edges that represent hotel stays and taxi transfers; this step will be discussed in Sect. 8.4.4.
4. We finally apply the Pareto Dijkstra algorithm on the extended graph in order to retrieve the set of Pareto optimal late-night connections. For the Pareto search, we introduce additional Pareto criteria to minimize the travel time during nighttime as well as the costs to be paid by the transportation company.

8.4.2 Retrieving Taxi Transfers

In order to incorporate taxi transfers into the search for late-night connections, we require a pre-processing step prior to employing the Pareto Dijkstra algorithm. We retrieve a set of reasonable taxi transfers from public transport stations to the destination station $dest_st(q)$ of the query q . First, we define the set of stations $S_T \subset S$ where the passenger could be collected by a taxi. These are stations within a specific radius around $dest_st(q)$. We introduce this limitation in order to limit the taxi costs; the radius is a parameter, and can be specified by the transportation company. The definition of the radius also accelerates the search since a large number of stations are excluded which are not relevant for the current query. Since the number of the stations in the selected radius could still be large, for faster run times, we restrict the selection

to stations where short-distance and long-distance trains halt; thus, bus and streetcar stations are excluded from the set S_T .

Then, from each station $s \in S_T$, we compute a taxi transfer to $dest_st(q)$. This can be accomplished using interfaces of taxi service providers if available. Alternatively, we could use solutions as the Open Source Routing Machine in order to compute taxi routes [LV11]. The costs for the taxi transfers are retrieved as will be described in Sect. 8.4.3. The taxi transfers are finally incorporated into the search as will be explained in Sect. 8.4.4.

Note: If the departure station $dep_st(q)$ of the query is within the radius around $dest_st(q)$, a direct taxi transfer from $dep_st(q)$ to $dest_st(q)$ could be a reasonable option. Our approach even finds such late-night connections.

8.4.3 The Costs Calculation

Delay compensation As described at the beginning of Sect. 8.4, the transportation company pays compensations for delayed arrivals of passengers at their destinations. The amount depends on the delay and the original ticket price. We assume that the function of computing the delay compensation is provided by the transportation company.

Hotel stay We assume that the cost for a hotel stay is fixed and known. Alternatively, it could be retrieved from interfaces of the hotel providers in a pre-processing step.

Taxi transfer For a taxi transfer, we assume that the cost depends on the distance; in addition there is a base price which is added to the distance dependent price. We assume that there is either a function to compute the price of a taxi transfer or the services used to retrieve the taxi transfers also price it.

8.4.4 Modeling Taxi Transfers and Hotel Stays

We extend the time-dependent graph (cf. Sect. 2.1.2)—which represents the timetable—by *hotel and taxi edges* as illustrated in Fig. 8.1 (on page 148); these edges model hotel stays and taxi transfers. Once the hotel stays and the taxi transfers are modeled, we find the set of attractive late-night connections by applying the Pareto Dijkstra algorithm as will be explained in Sect. 8.4.5. In the following, we introduce hotel and taxi edges more precisely.

Hotel edges From Sect. 8.4 (cf. *Taxi transfers and Hotel stays*), recall that at each station in the set $S_H \subset S$ there is hotel. For each station $s \in S_H$, we introduce a

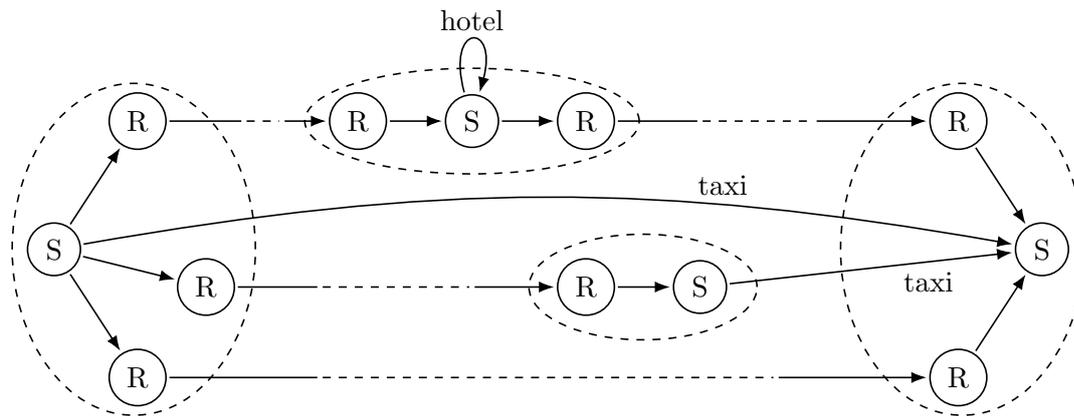


Figure 8.1: The figure illustrates, schematically, sequences of edges and nodes which build late-night connections. The dashed ellipses illustrate stations; all nodes in such an ellipse belong to the same station. The stations at the left and at the right of the figure are $dep_st(q)$ and $dest_st(q)$ respectively. Station nodes and route nodes are labeled with S and R respectively. Each dashed arrow represents a sequence of edges in the graph—for example a sequence of train moves and transfer activities. Four different late-night connections are depicted. The first one contains a hotel stay, the second one is a direct taxi transfer, the third one contains a taxi transfer after moving by public transport, and the fourth one solely consists of public transport—for example a special night train.

hotel edge. This edge is a loop, and connects the station node of s to itself.

Analogously to a route edge (cf. Sect. 2.1.2), a hotel edge is time-dependent; its length depends on the time $t \in TIMES$ when the edge is asked for its length by the Pareto Dijkstra algorithm (cf. Sect. 2.2.1, *Create labels during the search*). Moreover, the length depends on the parameters minimum stay duration and earliest check-out time (cf. Sect. 8.4, *Taxi transfers and Hotel stays*). Given a time $t \in TIMES$, a minimum stay duration $min_stay \in \mathbb{N}$ (in minutes), and an earliest checkout time $checkout \in TIMES$, the length of a hotel edge equals

$$\max \{ checkout - t, min_stay \}.$$

The price of a hotel edge is a fixed value as described in Sect. 8.4.3. The transfer cost of a hotel edge equals 0.

Taxi edges In Sect. 8.4.2, we described the computation of the taxi transfers for a query. Each taxi transfer starts at some station $s \in S_T$ and ends at the destination station $dest_st(q)$. Each taxi transfer is modeled by a *taxi edge* that connects the station node of s to the station node of $dest_st(q)$. The length and the price of a taxi edge are fixed values; they are computed as described in Sect. 8.4.2 and 8.4.3. The transfer cost for taxi edges equals 0 except for direct taxi edges from the station node of $dep_st(q)$ to the station node of $dest_st(q)$; an explanation follows.

From Sect. 2.1.2 (cf. *Edge costs*), recall that, in the time-dependent graph model, the edge from a route node to a station node has a transfer cost of 1. For each train connection, we even count a transfer when taking the edge from a route node at the destination station to the station node at the destination station. Consequently, each label at the destination node has a transfer count that is larger by 1 compared to the correct number of transfers. The Pareto Dijkstra algorithm, as described in Sect. 2.2, still delivers correct results since the additional transfer is counted equally for all train connections. In order to compute correct results after adding taxi edges to the graph, if a direct taxi edge from the station node of $dep_st(q)$ to the station node of $dest_st(q)$ is added to the graph, we set its transfer cost to 1 (cf. Fig. 8.1).

8.4.5 The Pareto Search for Late-Night Connections

In order to compute attractive late-night connections, we apply the Pareto Dijkstra algorithm (cf. Sect. 2.2) on a time-dependent graph that models the timetable and is extended as described in Sect. 8.4.4. Since we have an on-trip scenario, we exactly create one start label at the station node of $dep_st(q)$ as described in Sect. 2.2.2 (cf. *Creating start labels*). The station node of $dest_st(q)$ is specified as the destination node for the search.

Note: If desired, the approach from this section can be combined with the approaches from Sect. 7.1.2, 7.1.3, and 7.2 in order to find reliable connections. However, for sake of convenience, we do not consider reliability in this section.

We use the criteria travel time and number of transfers in order to find attractive train connections. In order to minimize the travel time during nighttime as well as the costs to be paid by the transportation company (as described at the beginning of Sect. 8.4), we introduce two additional Pareto criteria denoted as the *tp-costs criterion* and the *nighttime criterion* respectively. These are defined below (cf. *The tp-costs criterion* and *The nighttime criterion*). We will see that an extension of the labels is required as will be discussed below (cf. *The hotel flag*). We also discuss an adjustment of the domination rules during the search (cf. *Adjusted domination rules*).

The tp-costs criterion This criterion stores the costs to be paid by the transportation company; this Pareto criterion is minimized during the search. As described in Sect. 8.4.3, costs are caused in case of taxi transfers or hotel stays. Moreover, the transportation company must pay compensations in case of delayed arrivals at the destination. The cost criterion for the start label equals zero. During the search, it is computed as follows. Recall the creation of the labels during the search (cf. Sect. 2.2.2, *Creating labels during the search*); given an existing label \hat{l} at a node $v \in V$ and an

outgoing edge $(v, w) \in A$ of v , a new label l is created at the node w . The cost criterion for the label l equals

$$tp_cost(l) = tp_cost(\hat{l}) + tp_cost(v, w).$$

The value $tp_cost(v, w)$ denotes the cost caused by taking the edge (v, w) ; it equals the delay compensation amount—incurring when taking (v, w) —plus

- the price for the hotel stay if (v, w) is a hotel edge,
- the price for the taxi transfer if (v, w) is a taxi edge, and
- zero otherwise.

As announced in Sect. 8.4.3, we assume that the function of computing the delay compensation is provided by the transportation company; see Sect. 9.5.1 (cf. *Model and parameters*) for an example.

The nighttime criterion This criterion stores the travel time during nighttime; it is denoted by *night_travel_time*. The nighttime is defined as a time interval; for example from midnight to 5:00 o'clock. This criterion is minimized as a Pareto criterion since traveling during nighttime is not desired by passengers. For the start label, it equals zero. Recall the creation of the labels during the search (cf. Sect. 2.2.2, *Creating labels during the search*); given an existing label \hat{l} at a node $v \in V$ and an outgoing edge $(v, w) \in A$ of v , a new label l is created at the node w . The nighttime criterion for the label l equals

$$night_travel_time(l) = night_travel_time(\hat{l}) + night_travel_time(t(\hat{l}), len)$$

where

- $t(\hat{l}) \in TIMES$ denotes the timestamp (cf. Sect. 2.2.2, *Labels*) of the label \hat{l} ,
- len denotes the length of the edge (v, w) , and
- the function $night_travel_time(t(\hat{l}), len)$ delivers
 - zero if (v, w) is a hotel edge and
 - the overlap time between the time interval $[t(\hat{l}), t(\hat{l}) + len]$ and the nighttime for all other edge types.

The hotel flag We assume that, after a hotel stay, the passenger travels by public transport to the destination; the transportation company does not pay for a hotel stay as well as a taxi transfer. Hence, we do not compute late-night connections which contain a hotel stay as well as a taxi transfer. This constraint is modeled as follows.

Each label for the Pareto Dijkstra search (cf. Sect. 2.2.2, *Labels*) is extended by a *hotel flag* $\in \{0, 1\}$. For the start label, it is set to 0. Recall the creation of the labels during the search (cf. Sect. 2.2.2, *Creating labels during the search*); given an existing label \hat{l} at a node $v \in V$ and an outgoing edge $(v, w) \in A$ of v , a new label l is created at the node w . The hotel flag for label l equals

$$\text{hotel_flag}(l) = \text{hotel_flag}(\hat{l}) + \text{is_hotel_edge}(v, w)$$

where $\text{is_hotel_edge}(v, w)$ delivers 1 if (v, w) is a hotel edge, and 0 otherwise.

We forbid a taxi transfer after a hotel stay as well as multiple hotel stays as follows. Let l, \hat{l} , and (v, w) be defined as above. If (v, w) is a taxi or a hotel edge, we create the label l as the successor of \hat{l} only if $\text{hotel_flag}(\hat{l})$ equals zero.

Adjusted domination rules We adjust the domination rules (cf. Sect. 2.2.2, *Label dominance*) for the comparison of labels at each node except for the destination node. A label l is not allowed to dominate another label l' if $\text{hotel_flag}(l) \neq \text{hotel_flag}(l')$. This constraint ensures that no Pareto optimal late-night connection is dominated during the search. More explanations follow.

If $\text{hotel_flag}(l)$ equals 1, the label l cannot use a taxi anymore. In contrast, the label l' could still use a taxi if $\text{hotel_flag}(l')$ equals 0. Hence, compared to l , the label l' has more options for the continuation of the journey. In this case, we forbid a domination of l' by l .

On the other side, if $\text{hotel_flag}(l)$ equals 0 and $\text{hotel_flag}(l')$ equals 1, in contrast to l , the label l' has already bridged the nighttime. Hence, connections which are constructed from the label l potentially require longer travel times during the nighttime compared to connections which are constructed from the label l' . Therefore, in this case, we forbid a domination of l' by l .

This concludes the extension of the Pareto Dijkstra algorithm that is required to find late-night connections. An evaluation of this approach can be found in Sect. 9.5.

8.5 Conclusion

Reliable intermodal connections We discussed a method of assessing the reliability of rides by individual modes of transport. Our method is based on predictions

which are computed using real, historical data. However, instead of our assessment method, even other assessment methods could be combined with our approaches to the search for reliable intermodal connections and complete connections. We discussed four different approaches to computing reliable train connections and complete connections. The search for optimal complete connections is left to future research.

We have implemented and tested the approaches from Sect. 8.3.2 and 8.3.4. However, in this work, we do not evaluate these approaches, not least because of the lack of data for bike sharing and car sharing services.

Late-night connections Train connections could break late at night such that the destination cannot be reached by public transport prior to the end of operations. We presented a solution that finds attractive late-night connections for this scenario. We apply the Pareto Dijkstra algorithm extended by additional criteria minimizing the travel time during nighttime and the costs to be paid by the transportation company. This approach is evaluated in Sect. 9.5.

Chapter 9

Computational Study

In this chapter, we present evaluations of approaches discussed in this work. We implemented the approaches as extensions of the existing timetable information system MOTIS (Multi-Objective Traffic Information System); it has been developed in the institute of Algorithm Engineering at the Technische Universität Darmstadt in Germany [Alg16]. The code is written in C++.

We first explain in Sect. 9.1 which data we used for our evaluations. We then present in Sect. 9.2 evaluations of computations of probability distributions describing the random times of events in the timetable. In Sect. 9.3, we evaluate the quality of our reliability assessments from Chapter 3. In Sect. 9.4, we demonstrate an extensive evaluation of our optimal approach from Chapter 5; the optimal approach is compared to the approaches from Sect. 7.1. We finally present in Sect. 9.5 an evaluation of the approach from Sect. 8.4 that computes late-night connections.

9.1 Setup Data

In this section, we explain on which data our evaluations are based.

Timetable We used real timetable data provided to us by the German railway company, Deutsche Bahn AG [Bah16c]. In addition, they provided to us the values for minimum dwell and transfer times as well as maximum waiting times (cf. Sect. 1.1 and 1.2). We used timetables with validity periods in the past; the real times (cf. Sect. 1.3) of a large portion of the events in the timetables have been provided to us by Deutsche Bahn AG as well.

Functions describing random prepared times From Sect. 3.3.2, recall that, for each departure event $e_{dep} \in E_{dep}$, there is a random prepared time with the probability

density function $\xi(e_{dep})$. We collected data about these functions from different sources as follows.

In our input data, provided by Deutsche Bahn AG, the function $\xi(e_{dep})$ is given for the very first departure events of the short-distance and long-distance trains of relevant and frequent train categories. We used the functions for short-distance trains (S-Bahn) also for streetcars. For the very first departure events of buses, we learned the respective functions as described in Sect. 3.4.1; for this learning procedure, we used a history of real delay data kindly provided to us by the local public-transport organization DADINA [DAD16]. For the rest of the departure events, we simply defined the following function:

$$\xi(e_{dep}, t) = \begin{cases} 1 & \text{if } t = \text{sched}(e_{dep}), \\ 0 & \text{otherwise.} \end{cases}$$

Functions describing random train move durations From Sect. 3.3.2, recall that there is a probability density function $\theta(tm, t_{dep})$ for the random duration of each train move $tm \in TM$ and for each possible departure time $t_{dep} \in TIMES$ of the departure event of tm . We collected data about these functions from different sources as follows.

For relevant and frequent classes of short-distance and long-distance trains, Deutsche Bahn AG provided to us probability density functions for the random durations of their train moves. We used the functions for short-distance trains (S-Bahn) also for streetcars. For buses, we learned the respective functions as described in Sect. 3.4.2; for this learning procedure, we used a history of real delay data kindly provided to us by the local public-transport organization DADINA [DAD16].

For train moves of the remaining train moves (which are not covered above), we proceeded as follows. For the evaluations in Sect. 9.2 and 9.3, we generated one-point probability distributions with a probability of one for the stochastic event that the real duration of the train move equals its scheduled duration. For the evaluations in Sect. 9.4, we generated realistic distributions as discussed in Sect. 3.4.3: for the random duration of such a train move (e_{dep}, e_{arr}) and for each time $t_{dep} \in \text{supp}(\phi(e_{dep}))$, we assumed a Cauchy distribution parameterized depending on t_{dep} and the scheduled duration of (e_{dep}, e_{arr}) ; the parameters were chosen such that the generated distributions were realistic; see Appx. A.3 for examples.

Hardware Our computations were carried out on a desktop PC with Intel Xeon CPU E3-1270v2 3.50GHz and 32 GB of RAM.

9.2 Distributions for Random Event Times

In Sect. 9.2.1, we present the evaluations of the procedures of computing probability density functions for random event times. In Sect. 9.2.2, we evaluate the procedure of updating precomputed functions according to real event times.

9.2.1 Computing the Probability Density Functions

In this section, we evaluate the procedure of computing the probability density functions describing the random times of events. We evaluate two variants which are as follows:

1. The pre-processing procedure as discussed in Sect. 3.8.2 using the extended version of the time-expanded graph from Sect. 3.13.1.
2. The hybrid processing procedure from Sect. 3.8.3 using the time-dependent graph from Sect. 2.1.2 combined with the dependency graph from Sect. 3.13.2.

Setup Using real timetable data from German railways, we prepared a time-expanded and a time-dependent graph for the first and the second variant respectively. With the time-expanded graph, we only modeled long-distance and short-distance trains. With the time-dependent graph, we additionally modeled local public transport such as buses and streetcars. We performed the evaluation for the following two days periods in November 2016: 21-22, 22-23, 23-24, 24-25, 25-26, 26-27, and 27-28; the very first day was a Monday.

Note: A two days period allowed us to search for train connections that start and end at the same day as well as train connections that end at the very next day.

Negligible probabilities As explained in Sect. 3.13.3, we dropped negligible probabilities; the parameter ϵ from that section is set to 10^{-5} in our computational study. The cardinality of $\text{supp}(\phi(e))$, for each event processed in the evaluation from this section, never exceeded 135 minutes.

Results The evaluation results are presented in Tables 9.1 and 9.2; the tables give reports on the number of the events for which we computed the probability density functions; the run time required for each procedure is listed as well. The number of all events in Table 9.2 is significantly higher than the number of all events in Table 9.1 since the latter additionally modeled local public transport. The results demonstrate that the setup procedure for our stochastic model from Chapter 3 is fast and efficient even for large timetables.

	unit	Mon. Tue.	Tue. Wed.	Wed. Thu.	Thu. Fri.	Fri. Sat.	Sat. Sun.	Sun. Mon.
# events	M	2.0	2.0	2.0	2.0	1.9	1.6	1.8
run time	s	11.8	11.8	11.8	11.8	10.8	9.5	10.5

Table 9.1: Processing events using the extension of the time-expanded graph model as defined in Sect. 3.13.1. “# events”: number (in millions) of departure and arrival events in the timetable for the respective two days interval. “run time”: run time (in seconds) required to process the events for the respective two days interval.

	unit	Mon. Tue.	Tue. Wed.	Wed. Thu.	Thu. Fri.	Fri. Sat.	Sat. Sun.	Sun. Mon.
# all events	M	58.8	58.8	58.8	58.8	47.9	32.4	43.2
# processed	M	2.6	2.6	2.6	2.6	2.4	2.1	2.3
run time	s	22.2	21.9	21.9	22.4	20.6	15.3	19.9

Table 9.2: Processing events using the time-dependent graph model combined with the dependency graph from Sect. 3.13.2. “# all events”: number (in millions) of all departure and arrival events for the respective two days period. “# processed”: number (in millions) of all events for which a probability density function is computed. “run time”: run time (in seconds) required to process the events for the respective two days interval.

Note: We measured the run times five times and listed the average values in Tables 9.1 and 9.2.

9.2.2 Updating the Probability Density Functions

In Sect. 3.9, we discussed the procedure of updating the function $\phi(e)$ for each event e where the real event time is known. We evaluated the update procedure from that section; more precisely, we evaluated the effort to keep all functions $\phi(e)$, $e \in E$, up-to-date.

Setup We used real timetable data and real delay data (the real event times as discussed in Sect. 1.3) from German railways. To model the timetable, we used the hybrid procedure from Sect. 3.8.3 using the time-dependent graph model combined with the dependency graph as defined in Sect. 3.13.2. We performed the evaluation for 21 to 27 November 2016; the very first day was a Monday. We set the parameter ϵ from Sect. 3.9 to 10^{-3} .

Note: In contrast to the evaluation in Sect. 9.2.1, in this section, we used one-day periods.

	unit	Mon.	Tue.	Wed.	Thu.	Fri.	Sat.	Sun.
# all events	M	29.5	30.0	30.0	30.0	29.5	18.7	14.0
# processed	M	1.3	1.3	1.3	1.3	1.3	1.1	1.0
run time	s	10.1	10.0	10.0	9.9	10.0	7.1	6.2

Table 9.3: Processing events with the time-dependent graph model combined with the dependency graph from Sect. 3.13.2. “# all events”: number (in millions) of all departure and arrival events in the timetable for the respective day. “# processed”: number (in millions) of all departure and arrival events for which a probability density function is computed. “run time”: run time (in seconds) required to process the events for the respective day.

	unit	Mon.	Tue.	Wed.	Thu.	Fri.	Sat.	Sun.
# events	K	189.8	187.2	184.6	184.9	181.1	156.2	136.9
# recomp.	M	13.6	12.9	12.6	13.2	13.2	9.8	8.6
run time	s	81.2	75.3	73.8	77.4	78.6	50.1	43.2
r. t. per min.	ms	56.4	52.3	51.2	53.7	54.6	34.8	30.0

Table 9.4: Updating events with the time-dependent graph model combined with the dependency graph from Sect. 3.13.2. “# events”: The number (in thousands) of all events—with precomputed functions—for which we received the real event times from Deutsche Bahn AG. “# recomp.”: The number (in millions) of recomputations of the functions. “run time”: the total run time (in seconds) required for all recomputations at the respective day. “r. t. per min.”: the run time (in milliseconds) per minute required to keep the functions $\phi(e)$, $e \in E$, up-to-date.

Results We performed the following evaluation for each day separately. We first computed the probability density functions according to the hybrid model from Sect. 3.8.3; the number of all events, the number of the events for which the probabilities were computed in pre-processing, and the run time required for the pre-processing step are listed in Table 9.3 (on page 157). Using real delay data for the respective day, we then updated the precomputed functions in one minute steps. This update interval ensures that all functions are always up-to-date such that no information in the graph is older than one minute. The results are listed in Table 9.4 (on page 157).

From Sect. 3.9, recall that once the function $\phi(e)$ is changed for an event e , the probability density functions of all events which depend on e need to be recomputed. This results in millions of recomputations per day as listed in Table 9.4. However, the run time required to keep the functions up-to-date once per minute is in the range of milliseconds; the update procedure is very efficient.

9.3 The Quality of the Reliability Assessments

In Sect. 3.7, we presented our approach to assessing the reliability of train connections. We evaluated the quality of our reliability assessments and present the results here.

Briefly, we evaluated a large set of train connections. We first assessed the reliability of each train connection prior to the time when the real event times become known. We then checked the feasibility of each train connection according to the real event times which were known to us owing to Deutsche Bahn AG. This comparison allowed us an evaluation of the quality of our reliability assessments. In the following, we explain the evaluation procedure more precisely.

The procedure The evaluation procedure was is follows.

- The setup process to model the timetable is exactly as in Sect. 9.2.2.
- After the setup, for each day from 21 to 27 November 2016, we randomly created a number of 100,000 queries consisting of a departure station, a destination station, and a one hour time interval for the departure. We obtained 700,000 queries in total.
- Using our timetable information system MOTIS, we computed the set of all Pareto optimal train connections (cf. Sect. 2.2) for each query; travel time and number of transfers were the Pareto criteria. We obtained 1,183,174 train connections in total.
- We computed the probability of the success of each of the train connections by our method for reliability assessment from Sect. 3.7. On average, our method required 317 microseconds to compute the probability of the success of a train connection.
- Last, for each train connection, we checked its feasibility according to real event times (cf. Sect. 1.3.2) using real delay data—from Deutsche Bahn AG—for the period from 21 to 27 November 2016. More precisely, for each transfer activity (e_{arr}, e_{dep}) in each of the train connections, we checked whether

$$real(e_{arr}) + min_dur(e_{arr}, e_{dep}) \leq real(e_{dep})$$

is true. In words, we checked whether passengers had sufficient time to transfer from the first train to the second one.

The feasibility of a train connection—according to real times—can be verified only if, for each transfer activity (e_{arr}, e_{dep}) in the train connection, the real times of e_{arr} and e_{dep} are known to us. From Deutsche Bahn AG, we did not receive the real times of all events in the timetable. We consequently could not check the feasibility of many train connections which we had computed in the second step of the evaluation procedure. In

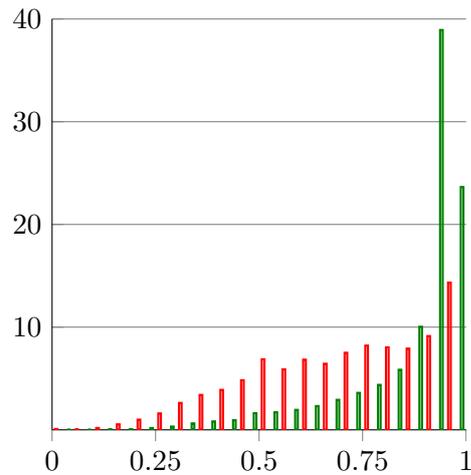


Figure 9.1: The x-axis represents the values for the *probability of success* computed by our reliability assessment method. The y-axis represents the *percentage* of the train connections. The *green* and the *red* histograms are computed over all train connections which were *feasible* and *infeasible* according to real event times respectively.

total, we could verify the feasibility of 136,601 train connections; this builds the set of train connections for the rest of our computational study in the following.

Results In Fig. 9.1 (on page 159) and Fig. 9.2 (on page 160), we analyze our reliability assessments (the probabilities of success) for the train connections which were feasible according to the real times as well as for the train connections which were infeasible. The results demonstrate that our method for reliability assessment frequently delivered high probabilities of success for train connections which were actually feasible according to real event times. Train connections which broke usually had lower probabilities of success.

Note: On average, the evaluated train connections had scheduled travel times of 351 minutes and 2.8 transfer activities (cf. Sect. 1.2).

Receiver operating characteristic For a further quality measurement of our method for reliability assessment, we computed the receiver operating characteristic (ROC) curve [Faw06]. Fig. 9.3 (on page 161) demonstrates the result; the area under the curve (AUC) equals 0.81. This curve is computed as follows.

For each $\rho \in [0, 1] \subset \mathbb{R}$, for all train connections with a probability of success lower than ρ , we *predicted* that they become *infeasible* according to the real event times; for the rest, we *predicted* that they will be *feasible* according to the real event times. The x-axis represents the false positive rate which equals FP/N ; the y-axis represents the true positive rate which equals TP/P ; these values are defined as follows. FP is

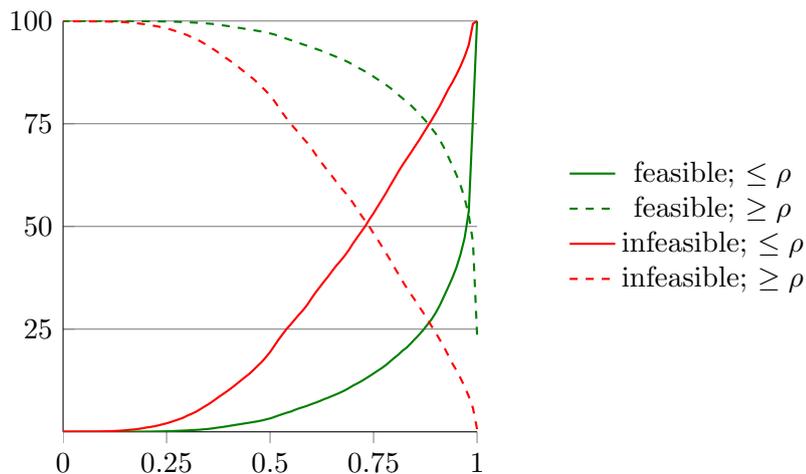


Figure 9.2: The x-axis represents the values for the *probability of success* $\rho \in [0, 1] \subset \mathbb{R}$ computed by our reliability assessment method. The y-axis represents the *percentage* of the train connections. The *green* and the *red* curves are computed over all train connections which were *feasible* and *infeasible* according to real event times respectively. The *solid* curves represent the percentage of the train connections with a probability of success *lower* than or equal to the respective value for ρ . The *dashed* curves represent the percentage of the train connections with a probability of success *greater* than or equal to the respective value for ρ . To compute the curves, we evaluated 100 values for ρ and connected the measurement points.

the number of all *feasible* train connections (according to real event times) for which our predictions were *incorrect*. TP is the number of all *infeasible* train connections (according to real event times) for which our predictions were *correct*. N and P are the numbers of all train connections which were *feasible* and *infeasible* according to real event times respectively. For a better understanding, the function $y = x$ represents the prediction strategy to guess randomly [Faw06].

Discussion The evaluation demonstrates that our reliability assessments are accurate and precise to a significant extent. However, we believe that better results can be obtained by our reliability assessment method as follows.

- The quality of our reliability assessments strongly depends on the quality of the input data used for the random prepared times and the random train move durations. The functions which we used for this evaluation were not as accurate as desired; we even had to generate some functions as explained in Sect. 9.1. An incorporation of information about the tracks and the stations of the train moves could improve the quality of the reliability assessments (cf. Sect. 3.4).
- The maximum waiting times (cf. Sect. 1.3.2) which we could use as input data were not exactly the same as applied by Deutsche Bahn AG. We found out that

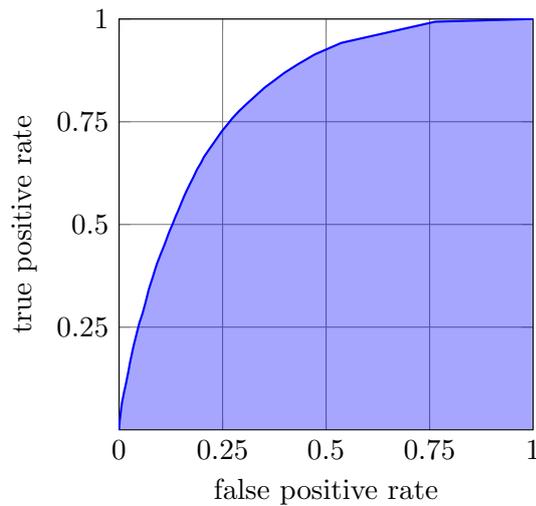


Figure 9.3: The figure illustrates the receiver operating characteristic (ROC) curve [Faw06] which is computed as explained in Sect. 9.3 (cf. *Receiver operating characteristic*).

the waiting times which we used for this computational study did not improve our reliability assessments. Again, better input data is required.

- We performed this evaluation using the time-dependent graph model from Sect. 2.1.2 in order to model short-range local public transport in addition to short-distance and long-distance trains. However, this model—as presented in this work—does not support platform-specific minimum transfer durations for trains that arrive and depart at platforms which are close to each other (cf. Sect. 1.2). Extending our graph model to support platform-specific minimum transfer durations would improve the reliability assessments.

This concludes the evaluation of the reliability assessment method from Sect. 3.7.

Note: In [KSWZ12], we published a similar evaluation using other timetable data; we obtained similar results.

9.4 On-Time Arrival Guarantee

We addressed the problem of finding optimal complete connections in Sect. 4.3. In this section of our computational study, we evaluate our approach to the search component from Chapter 5. For this evaluation, we focus on the problem variant “latest departure subject to arrival guarantee” from Sect. 4.3.3.1; it is the most interesting and relevant problem variant.

Overview In Sect. 9.4.1, we specify the objectives of this evaluation. We discuss in Sect. 9.4.2 and 9.4.3 how we proceeded to achieve these objectives. In Sect. 9.4.4, we explain the setup data for the evaluation. In Sect. 9.4.5–9.4.10, we present and discuss the evaluation results. Last, in Sect. 9.4.11 and 9.4.12, we evaluate the heuristics which are applicable to our search approach as well as the effect of our methods of discovering relevant events in order to accelerate the search.

9.4.1 Objectives of the Computational Study

Our computational study is based on a real-world, national train network (in Germany) with real timetable data. From our computational study, we draw the following conclusions:

1. If meeting the deadline is the primary criterion for the passenger, then (I) it must be the sole primary criterion for the search component as well, and (II) alternative continuations must be taken into account and optimized at the booking stage (cf. Sect. 4.4).
2. State-of-the-art traffic information systems do not provide solutions for the problem specified in Sect. 4.3. To overcome this problem, some passengers choose an attractive train connection—computed with no regard for the probability of success—and change to the next available alternative continuation in case that a transfer becomes infeasible due to delays. We demonstrate that this strategy frequently yields suboptimal and even infeasible outcomes. Hence, it is well worth the effort to search for *optimal* outcomes through our approach.
3. The simpler approach to incorporate buffer times at critical points in the journey yields suboptimal outcomes.
4. Outcomes of our approach are attractive in terms of *travel time* and the *number of transfers*. Thus, our approach delivers outcomes with a high travel convenience, and may be attractive to passengers.
5. Our approach has efficient response times and is suitable from a practical point of view.

In the rest of this section, we present the evaluations of the above claims.

9.4.2 The Evaluated Approaches

We evaluated the claims from Sect. 9.4.1 by comparing our optimal approach from Chapter 5 to the alternative approaches from Sect. 7.1.1–7.1.3. In the following, we list all evaluated approaches and review their outcomes for each query q (cf. Sect. 4.3.1).

OPT We denote by *OPT* our optimal approach to the search component from Chapter 5; we combined it with the recursive formula from Sect. 5.3 and the incremental approach from Sect. 5.9. The approach *OPT* delivers the optimal complete connection according to the objective function from Eq. 4.3.1 (on page 73).

Note: The approach OPT uses the recursive formula from Sect. 5.3; in Sect. 9.4.9, we will additionally evaluate the extended recursive formula from Sect. 5.4.

DP-L This approach has been introduced in Sect. 7.1.1; without regard for probability of success, it delivers the train connection with the latest scheduled departure time at $dep_st(q)$. We compare *OPT* to this approach in order to verify the first and the fourth claim from Sect. 9.4.1.

DP-LP This approach has been introduced in Sect. 7.1.1; without regard for probability of success, it delivers the train connection with the latest scheduled departure time at $dep_st(q)$ among Pareto optimal train connections. In Sect. 7.1.1, we introduced the parameter δ to specify the time interval for the arrival at $dest_st(q)$; for δ , we used 100 minutes in our computational study. We compare *OPT* to this approach in order to verify the first and the fourth claim from Sect. 9.4.1.

CEA This approach has been introduced in Sect. 7.1.3; it simulates the behavior of passengers which is described in the second claim from Sect. 9.4.1. We introduced in Sect. 7.1.3 the parameter δ in order to specify the time interval for the arrival at $dest_st(q)$; for δ , we used 100 minutes in our computational study. We compare *OPT* to this approach in order to verify the first and the second claim from Sect. 9.4.1.

BT This approach has been introduced in Sect. 7.1.2; it delivers the train connection with the latest scheduled departure time at $dep_st(q)$ such that a minimum buffer time is satisfied for each transfer and prior to the deadline at $dest_st(q)$. The minimum buffer time as required for *BT* was denoted by τ in Sect. 7.1.2. In our computational study, we evaluated $\tau \in \{10, 15, 20, 30\}$ (in minutes); we denote these approaches by *BT-10*, *BT-15*, *BT-20*, and *BT-30* respectively. We compare *OPT* to this approach in order to verify the third claim from Sect. 9.4.1.

Applied heuristics For *OPT* as well as for all approaches for comparison, we applied the heuristic “maximum waiting time of passengers” from Sect. 5.11.1; we limited the maximum waiting time at a station for a transfer to 150 minutes. This is the only heuristic used for *OPT* in the evaluations presented in Sect. 9.4.5–9.4.9. We will discuss the effect of the heuristics from Sect. 5.11 in Sect. 9.4.11.

Note: Equally for all approaches, we dropped negligible probabilities as discussed in Sect. 3.13.3 and 9.2.1.

9.4.3 Classifying the Outcomes

In the previous section, we listed the approaches we evaluated. For each query q , the outcome of each of the approaches OPT and CEA is a complete connection as specified in Sect. 4.3.2; the outcome of each of the approaches $DP-L$, $DP-LP$, and BT is a train connection as specified in Sect. 1.2. In order to evaluate the approaches, we mainly focused on two relevant properties of their outcomes: probability of success and scheduled departure time. For complete connections, both properties have been discussed in Sect. 5.5. For train connection, the scheduled departure time has been introduced in Sect. 1.2; the probability of success is computed by the method from Sect. 3.7.2.

From Sect. 4.3.1, recall that, for a query q , the value $\rho_{req}(q)$ denotes the probability of success required by q . Moreover, the value $\rho(o)$ denotes the probability of success of the outcome o of an approach—the outcome o is either a train connection or a complete connection. For each evaluated query q , we classified the outcome o of each approach as follows:

- *void*: if the approach did not deliver any outcome to the query, its outcome is called *void*;
- *infeasible*: the outcome o is called *infeasible* if $\rho(o) < \rho_{req}(q)$;
- *feasible*: the outcome o is called *feasible* if $\rho(o) \geq \rho_{req}(q)$.

Furthermore, *feasible* outcomes are sub-classified as follows:

- *optimal*: a *feasible* outcome is called *optimal* if there was no other *feasible* outcome for query q which had a later scheduled departure time at $dep_st(q)$;
- *suboptimal*: a *feasible* outcome is called *suboptimal* if there was at least one other *feasible* outcome for query q which had a later scheduled departure time at $dep_st(q)$.

The approaches OPT and CEA deliver outcomes which are either *feasible* or *void* while $DP-L$, $DP-LP$, and BT even deliver *infeasible* outcomes for some queries.

To avoid misunderstandings: In Chapter 3 and Sect. 9.3, we used the terms “feasible” and “infeasible” in a different context and with a different meaning. In Sect. 9.4, the terms “feasible” and “infeasible” are defined according to Sect. 4.3.

9.4.4 Evaluation Setup and Test Queries

Timetable Using real timetable data from German railways, we prepared a time-expanded graph (cf. Sect. 2.1.1) for a two days period, 5-6 August 2014, with 2.2M departure and arrival nodes. This two days period allowed us to evaluate complete connections that start and end at the same day as well as complete connections that end at the very next day. We extended our time-expanded graph as discussed in Sect. 3.13.1 in order to provide an access to the functions $\phi(e)$, $e \in E$, and the interdependencies between the events in constant time.

Queries The queries (cf. Sect. 4.3.1) for the computational study were designed as follows. As departure and destination stations of the queries, we used 8,000 different station pairs extracted from real customer queries (provided by German railways). We combined each station pair with 10 different deadlines for the arrival at the destination:

$$deadline(q) \in \left\{ \begin{array}{l} 14:00, 17:00, 19:00, 21:00, 23:00, \\ 01:00, 07:00, 08:00, 11:00, 13:00 \end{array} \right\}$$

where the first five deadlines were on the first day and the last five deadlines were on the second day of the timetable's two-days period. This setting yielded 80,000 triples, each consisting of a departure and a destination station as well as a deadline. We combined each triple with five different values for the required probability of success:

$$\rho_{req}(q) \in \{ 0.95, 0.96, 0.97, 0.98, 0.99 \}.$$

We finally obtained 400,000 queries in total.

9.4.5 The Outcomes

We processed all queries, specified in Sect. 9.4.4, with our optimal approach *OPT* as well as with all alternative approaches from Sect. 7.1. The results of this evaluation are detailed in the following and in the rest of Sect. 9.4.

Feasibilities of the outcomes For each query, our approach *OPT* always delivered the *optimal* outcome unless there was no *feasible* one (details about the heuristic from Sect. 5.11.1 will be explained later in Sect. 9.4.11). For 5.3% of all queries, there were no *feasible* outcomes; thus, the outcomes of *OPT* were *void*. It comes as no surprise that for these queries no other approach could deliver *feasible* outcomes either.

In Table 9.5 (on page 166), we evaluate the feasibilities of the outcomes of the alternative approaches from Sect. 7.1. More precisely, for each alternative approach, we compare “the percentage of the queries for which the approach—as well as *OPT*—delivered *feasible* outcomes” to “the percentage of the queries for which *OPT* was

$\rho(q)$	outcome	<i>DP-I</i>	<i>DP-LP</i>	<i>BT-10</i>	<i>BT-15</i>	<i>BT-20</i>	<i>BT-30</i>	<i>CEA</i>
99%	<i>OPT</i> solely	59.8	59.8	16.8	8.7	5.1	5.5	12.8
	both feasible	34.9	32.9	77.9	86.0	89.6	89.2	79.9
98%	<i>OPT</i> solely	55.3	55.6	10.7	5.2	4.4	5.5	10.6
	both feasible	39.4	37.1	84.0	89.5	90.3	89.2	82.1
97%	<i>OPT</i> solely	52.3	52.8	7.8	4.2	4.2	5.4	9.3
	both feasible	42.4	39.9	86.9	90.5	90.5	89.3	83.4
96%	<i>OPT</i> solely	49.7	50.3	5.8	3.9	4.2	5.4	8.4
	both feasible	45.0	42.4	88.9	90.8	90.5	89.3	84.3
95%	<i>OPT</i> solely	47.3	47.9	4.7	3.7	4.1	5.4	7.6
	both feasible	47.4	44.8	90.0	91.0	90.6	89.3	85.1

Table 9.5: The outcomes of the approaches: The number in each cell is the percentage of all queries with the $\rho(q)$ -value as listed in the first column (i.e. 80k queries for each $\rho(q)$ -value).

“*OPT* solely”: queries for which *OPT* delivered *feasible*—and necessarily *optimal*—outcomes while the outcomes of the respective approach for comparison were either *void* or *infeasible*.

“both feasible”: queries for which *OPT* and the respective approach for comparison delivered *feasible* outcomes.

Note that, for 5.3% of the queries, no approach could deliver feasible outcomes. Moreover, for the comparison of *OPT* to *DP-LP* and *CEA*, 2.0% of the queries have been excluded as explained in Sect. 9.4.5 (cf. *Excluded queries*).

the only approach that delivered *feasible* outcomes”. We can see that the alternative approaches have performances inferior to *OPT*.

Excluded queries From Sect. 7.1.1 and 7.1.3, recall that *DP-LP* and *CEA* use a time interval for the arrival at $dest_st(q)$. To obtain a fair and valid comparison to *OPT*, we excluded each query for which the outcomes of *CEA/DP-LP* and *OPT* were as follows: *DP-LP* and *CEA* failed to find any Pareto optimal train connection arriving at $dest_st(q)$ within the defined arrival interval while *OPT* delivered an outcome with an alternative continuation arriving at $dest_st(q)$ earlier than this time interval. These were 2.0% of all queries; thus, in Table 9.5 (on page 166), the percentages in the respective cells (plus 5.3% for the queries without any feasible outcomes) add up to 98%. These queries are only excluded from the comparison of *OPT* to *DP-LP* and *CEA* and not from the rest of our computational study. Note that for the excluded queries the outcomes of *OPT* were *optimal*.

In Sect. 9.4.6–9.4.10, we discuss the objectives of our computational study from Sect. 9.4.1 and give empirical evidence for them.

$\rho(q)$	<i>OPT</i>	<i>BT-10</i>	<i>BT-15</i>	<i>BT-20</i>	<i>BT-30</i>	<i>CEA</i>
99%	100.0	56.8	39.9	29.1	17.1	76.0
98%	100.0	53.8	37.2	26.9	15.9	77.1
97%	100.0	51.8	35.6	25.8	15.2	77.8
96%	100.0	50.3	34.3	24.8	14.7	78.4
95%	100.0	48.9	33.4	24.1	14.3	78.9

Table 9.6: Optimality evaluation: For each approach, the table reports how often the *feasible* outcomes were *optimal*. The numbers are percentages computed over all *feasible* outcomes of the respective approach.

9.4.6 Relevance of Probability of Success and Optimality

Relevance of probability of success The evaluation of the outcomes of *DP-L*, *DP-LP*, and *CEA* supports our first claim from Sect. 9.4.1. Table 9.5 (on page 166) demonstrates that for about half of all queries, the approaches *DP-L* and *DP-LP* delivered *infeasible* outcomes while *feasible* outcomes existed (cf. “*OPT* solely”). Even the *CEA* approach failed to deliver *feasible* outcomes for a significant number of the queries. We conclude that for the problem from Sect. 4.3, where a probability of success has to be guaranteed, the probability of success must be the primary criterion for the search. Moreover, alternative continuations must be optimized already during the search.

Relevance of optimality Our second claim from Sect. 9.4.1 is proven by the comparison of *OPT* to the approach *CEA*. Above, we already demonstrated that *CEA* failed to deliver *feasible* outcomes while *feasible* complete connections existed. We now focus on the optimality of the *feasible* outcomes of *CEA*. For this, as presented in Table 9.6 (on page 167), we evaluated how often the *feasible* outcomes of each approach were actually *optimal*. As listed in Tables 9.5 and 9.6, for a significant number of queries, the *CEA* approach failed to deliver *optimal* or even any *feasible* outcomes. Moreover, details about the differences between the departure times of the outcomes of *OPT* and *CEA* can be found in Fig. 9.4 (on page 168). Compared to *CEA*, our approach *OPT* delivered complete connections with significantly better—i.e. later—departure times.

Altogether, we can see that it is well worth the effort to search for *optimal* outcomes with our dynamic programming approach *OPT*; the strategy of the passengers simulated by *CEA* is inferior to our optimal approach *OPT*.

9.4.7 Minimum Buffer Times

We now address our third claim from Sect. 9.4.1. Table 9.5 (on page 166) demonstrates that the *BT* approach delivers *feasible* outcomes for quite a large fraction of the queries

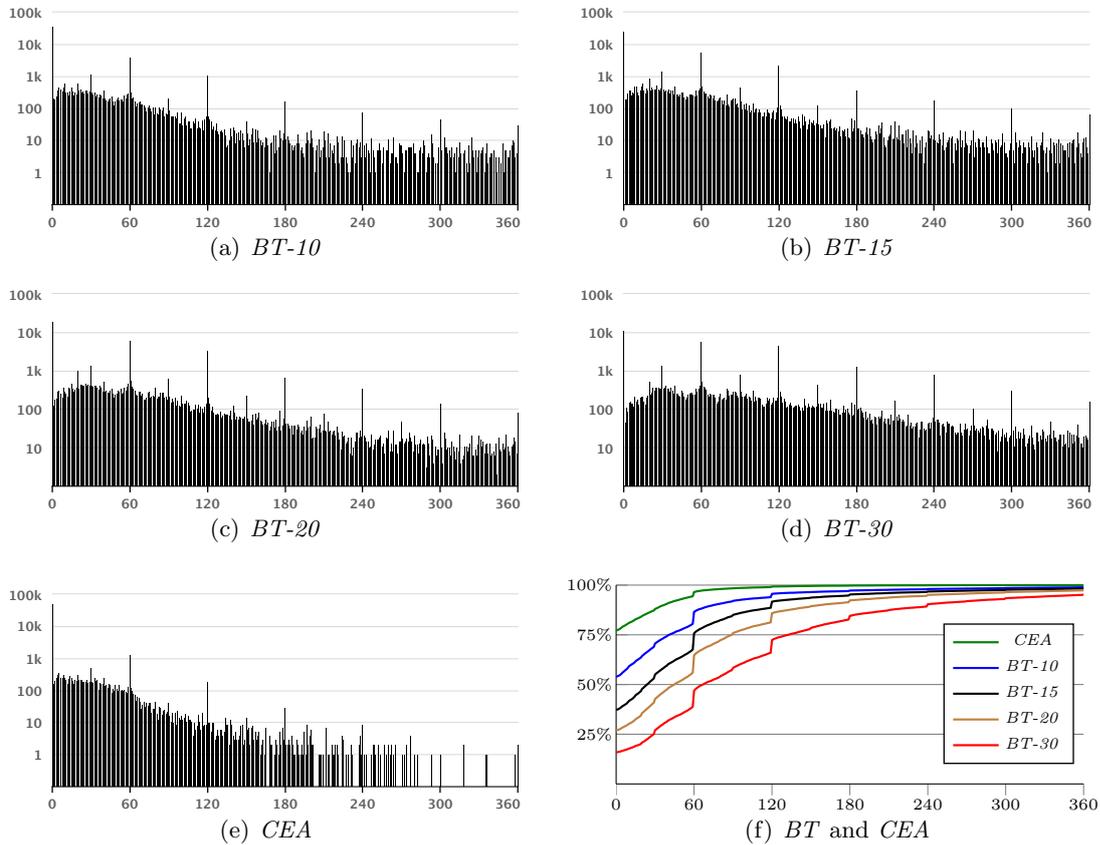


Figure 9.4: Detailed differences between the departure times of *OPT*'s *optimal* outcomes compared to the departure times of the *feasible* outcomes of *BT* and *CEA*. For each query q , the value δ (in minutes) equals “the departure time of *OPT*'s outcome” minus “the departure time of the outcome of *BT/CEA*”. For each value for δ , each of the histograms (a)–(e) illustrates the absolute number of the queries with that δ -value. The figure (f) is a cumulative plot; the x-axis represents the values for δ ; the y-axis represents the percentage of the queries with δ -values smaller than or equal to the respective δ -value on the x-axis. The maximum value illustrated for δ is 360. There are 706, 1197, 1846, 3345, and 4 outcomes of the approaches *BT-10*, *BT-15*, *BT-20*, *BT-30*, and *CEA* that are not illustrated (they have a δ -value greater than 360). Each histogram in the figure comprises all queries for which *OPT* as well as *BT/CEA* delivered a *feasible* outcome (cf. Table 9.5 (on page 166)). This figure comprises solely the queries with $\rho(q) = 98\%$ (we obtained similar results for the other values for $\rho(q)$). For the histograms (a)–(e), the y-axis has a logarithmic scale.

$\rho(q)$	earlier	average	median	$\rho(q)$	earlier	average	median
99%	49.1%	0.13	0.0	99%	77.6%	0.21	0.13
98%	44.2%	0.12	0.0	98%	75.6%	0.20	0.11
97%	41.0%	0.10	0.0	97%	74.1%	0.19	0.11
96%	38.3%	0.10	0.0	96%	72.9%	0.18	0.10
95%	36.0%	0.09	0.0	95%	72.1%	0.18	0.09

Table 9.7: The price of the on-time arrival guarantee; more precisely, the table reports on the earlier departure times of *optimal* outcomes (computed by *OPT*) compared to the latest departure times without regard for probability of success (computed by *DP-L*). The left table evaluates all queries for which the outcomes of *DP-L* were either *feasible* or *infeasible*. The right table evaluates only those queries for which the outcomes of *DP-L* were *infeasible*.

“earlier”: queries for which the *optimal* outcomes of *OPT* depart earlier than the *infeasible* outcomes of *DP-L*; the numbers are percentages of all evaluated queries.

“average”: the *relative duration increase*, as defined in Sect. 9.4.8, averaged over all evaluated queries.

“median”: the median of the *relative duration increase*, over all evaluated queries.

but not as frequent as *OPT*. The superiority of *OPT* over *BT* becomes more clear when comparing the departure times of the outcomes of both approaches. Table 9.6 (on page 167) focuses on this comparison; the results reveal that the *BT* approach frequently delivered *suboptimal* outcomes. Moreover, in Fig. 9.4 (on page 168), histograms demonstrate the detailed differences between the departure times of *OPT* and the *BT* approach. As we can see, *OPT* delivered significantly better—i.e. later—departure times for a large number of the queries. So, our computational study suggests that our third claim from Sect. 9.4.1 is valid.

9.4.8 The Price of On-Time Arrival Guarantee

So far, we have demonstrated that *OPT* delivers complete connections which guarantee the required probability of success and have scheduled departure times as late as possible. We now discuss the price of this on-time arrival guarantee; more precisely, we analyzed how much earlier departure times must be accepted by the passengers in order to meet the required probability of success. For this, we compared the departure times of the outcomes of *OPT* to the departure times of the outcomes of *DP-L*; recall that, without regard for probability of success, the latter delivers the latest departure time such that an arrival prior to the deadline is still possible.

The results of this evaluation are illustrated in Table 9.7 (on page 169) and Fig. 9.5 (on page 170). There, for each query q , the *minimum duration* denotes the absolute difference between $deadline(q)$ and the scheduled departure time of the outcome of *DP-L* for the query q . Moreover, the *duration increase* denotes the absolute time difference between the scheduled departure times of the outcomes of *OPT* and *DP-L* for the query q . The *relative duration increase* denotes the ratio of *duration increase*

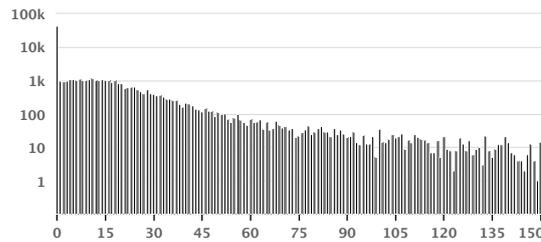


Figure 9.5: The price of the on-time arrival guarantee; more precisely, the histogram reports on the earlier departure times of *optimal* outcomes (computed by *OPT*) compared to the latest departure times without regard for probability of success (computed by *DP-L*). For each value for *relative duration increase*, as defined in Sect. 9.4.8, the histogram illustrates the number of the queries with that value; the values are multiplied by 100 for better readability. The maximum value that is illustrated in the histogram is 150; there are 581 (=0.7%) outcomes that are not illustrated since they have a value greater than 150. The histogram comprises all queries where the outcomes of *OPT* were *optimal* and the outcomes of *DP-L* were either *feasible* or *infeasible*. The histogram is exemplary; solely for queries with $\rho(q) = 98\%$ (we obtained similar results for the other values for $\rho(q)$). The y-axis has a logarithmic scale. For more details, we refer to Table 9.7 (on page 169).

to *minimum duration*.

For a fraction of the queries where the outcomes of *DP-L* were *infeasible*, we see that earlier departure times—such as computed by *OPT*—must be accepted in order to guarantee the required probability of success $\rho_{req}(q)$. For these queries, there were no *feasible* outcomes with later departure times. On the other hand, for all queries where *DP-L* delivered *feasible* outcomes, the departure times of the outcomes of *OPT* and *DP-L* were the same. As Table 9.7 (on page 169) reports, averaged over all queries for which *DP-L* delivered *feasible* or *infeasible* outcomes, the relative duration increase is acceptable. Moreover, *OPT* still delivered better departure times compared to the alternative approaches *BT* and *CEA*.

The approach *OPT* consequently computes outcomes with an on-time arrival guarantee and travel times which are still attractive even for busy people. The comparison above was the first step to prove the fourth claim from Sect. 9.4.1; in Sect. 9.4.9, we present further evaluations of the criteria travel time and number of transfers.

9.4.9 Travel Convenience

In the previous section, we discussed the price of the on-time arrival guarantee. In this section, we further on address the fourth claim from Sect. 9.4.1; we analyze the number of transfers and the travel time of the outcomes. For train connections, both properties are defined as described in Sect. 1.2; for complete connections, both are expected values as explained in Sect. 5.5.

The approach *OPT* optimizes these properties as secondary criteria; more precisely, they are used as tie-breakers. In Sect. 5.3 and 5.4, we presented two recursive functions

	<i>OPT-1</i>	<i>OPT-2</i>	<i>DP-L</i>	<i>DP-LP</i>	<i>BT-10</i>	<i>BT-15</i>	<i>BT-20</i>	<i>BT-30</i>	<i>CEA</i>
transfers	1.6	1.7	1.7	1.8	1.6	1.5	1.5	1.5	1.5
durations	257	247	244	240	273	283	293	311	291

Table 9.8: The number of transfers and the travel times (in minutes) of the approaches, averaged over all (feasible and infeasible) outcomes of each approach.

“*OPT-1*”: our optimal approach *OPT* with the recursive function from Sect. 5.3, the number of transfers as tie-breaker, and a tolerance of 10^{-4} for the comparison of the probabilities as explained in Sect. 5.3.4; see Table 9.9 (on page 171) for the evaluation of further tolerance values.

“*OPT-2*”: our optimal approach *OPT* with the extended recursive function from Sect. 5.4, the weighted sum as described in Sect. 9.4.9 as tie-breaker, and a tolerance value of 10^{-4} for the comparison of the probabilities.

Note: The expected travel time of the outcomes of *OPT-1* could only be computed by the recursive function from Sect. 5.4 in a post-processing step after the computation of the outcomes.

tolerance	0	10^{-5}	10^{-4}
transfers	3.3	1.7	1.6
suboptimal	0%	0.0065%	0.04%

Table 9.9: The expected number of transfers of *OPT*’s outcomes obtained by the recursive function from Sect. 5.3, averaged over all feasible outcomes.

“tolerance”: the tolerance value for the comparison of the probabilities as described in Sect. 5.3.4.

“transfers”: the expected number of transfers of the complete connections (on average).

“suboptimal”: the percentage of the queries for which *OPT* delivered suboptimal outcomes because of the applied tolerance.

that compute optimal complete connections; the first one allows us to optimize the number of transfers as a secondary criterion (cf. Sect. 5.3.4); the second one allows us to optimize a weighted sum of the number of transfers and the travel time as a secondary criterion (cf. Sect. 5.4.2). Both are evaluated in the following.

Optimizing number of transfers solely By the recursive formula from Sect. 5.3, we optimized the number of transfers as the secondary criterion. The results are listed in Table 9.8 (on page 171); see the values for *OPT-1*. We evaluated different tolerance values as discussed in cf. Sect. 5.3.4; the results are presented in Table 9.9 (on page 171). We see that the evaluated tolerance values significantly reduce the expected number of transfers but affect the optimality only marginally. As claimed in Sect. 9.4.1, *OPT* delivers outcomes with attractive numbers of transfers, comparable with *DP-L* and *DP-LP*.

Optimizing number of transfers and travel time We evaluated the extended recursive function from Sect. 5.4. As tie-breaker, we used the weighted sum from Eq. 5.4.5 (on page 90); for each arrival event e_{arr} and some time $t \in \text{supp}(\phi(e_{arr}))$, if

two or more successors obtain the maximum probability of success, recall that we select the successor with the better weighted sum. In the weighted sum, we set the weight $w_a = 1$; the weight w_t for the number of transfers is set to

$$w_t = \max \{ 15.0, 0.15 \cdot (\text{deadline}(q) - \text{sched}(e_{arr})) \}$$

where e_{arr} is the arrival event for which we determine the best successors according to Eq. 5.4.1 (on page 88). In words, the weighted sum requires that each additional transfer decreases the expected travel time by 15% of the time difference between $\text{deadline}(q)$ and $\text{sched}(e_{arr})$ but at least by 15 minutes.

The results are presented in Table 9.8 (on page 171); see the values for *OPT-2*. As claimed in Sect. 9.4.1, *OPT* delivers outcomes with attractive travel times, comparable with *DP-L* and *DP-LP*.

Note: Recall that the expected travel time can only be computed using the extended recursive function from Sect. 5.4. In order to see the improvement of the travel times obtained by that approach compared to the approach from Sect. 5.3, we calculated the travel times of the outcomes obtained by the recursive function from Sect. 5.3 in a post-processing step using the extended recursive function from Sect. 5.4. This calculation can be accomplished analogous to an ex-post calculation of the probability of the success of a complete connection as explained in Sect. 6.2.2.1 (cf. Recalculation of the probability of success).

9.4.10 Efficiency and Practicability

We finally address the last claim from Sect. 9.4.1. The response times of the approaches are listed in Table 9.10 (on page 173). For *OPT*, this table lists the response times which are obtained by applying our incremental approach from Sect. 5.9. We listed the times for all 80k queries with $\rho_{req}(q) = 98\%$ as an example; for all approaches for comparison, we measured similar response times for the other values for $\rho_{req}(q)$; for *OPT*, we noticed 3-4% longer response times when processing queries with $\rho_{req}(q) = 99\%$ compared to queries with $\rho_{req}(q) = 95\%$.

We see that the optimization of the travel time by the extended recursive formula from Sect. 5.4 is more expensive than applying the recursive function from Sect. 5.3; which recursive function is more appropriate, should be decided depending on the requirements and preferences of the passengers.

A desktop PC as specified in Sect. 9.4.4 is sufficient to run up to four instances of our implementation of *OPT* in parallel. The results presented in this section and the average response time of 2.0s are indicative that our optimal approach *OPT* is efficient and practicable as claimed in Sect. 9.4.1.

	<i>OPT-1</i>	<i>OPT-2</i>	<i>DP-1</i>	<i>DP-1P</i>	<i>BT-10</i>	<i>BT-15</i>	<i>BT-20</i>	<i>BT-30</i>	<i>CEA</i>
average	2,025	2,477	157	183	171	171	169	169	1,594
90%-q.	4,773	5,424	329	396	366	364	360	364	3,369

Table 9.10: For each approach, the table lists the average response time as well as the 90%-quantile of the response times over all 80k queries with $\rho(q) = 98\%$. All times are in milliseconds.

“*OPT-1*”: our optimal approach *OPT* with the recursive function from Sect. 5.3.

“*OPT-2*”: our optimal approach *OPT* with the extended recursive function from Sect. 5.4.

“average”: the response time of the respective approach on average.

“90%-q.”: the 90%-quantile of the response times of the respective approach.

9.4.11 Heuristics

In this section, we discuss the evaluation of the heuristics from Sect. 5.11. The only heuristic we used in our evaluations presented in Sect. 9.4.5–9.4.10 was “maximum waiting time of passengers”.

Maximum waiting time of passengers In Sect. 5.11.1, we discussed the maximum waiting time of passengers at a station for a transfer. This assumption and heuristic results in *suboptimal* outcomes for about 2.3% of all queries when limiting the maximum waiting time to 150 minutes. Fortunately, this loss of optimality is negligible as will be explained in the following. For the *optimal* outcomes for these queries, the passengers are expected to wait at some station for a time longer than 150 minutes; for 99% of these outcomes, this waiting window is during the night (12:00 midnight – 05:00 a.m.). Such journeys are not attractive, and, in such cases, passengers usually decide to travel a day earlier or even by other means of transport. Moreover, in such cases, the *suboptimal* outcomes which are computed by demanding a maximum waiting time are potentially more attractive than the optimal ones that we miss; the passengers have the possibility to sit or even sleep in a train instead of waiting at a station during the nighttime. In summary, this loss of optimality is negligible.

Without this heuristic, our search component would have up to six times longer response times on average. The time saving obtained by this heuristic has two main reasons. First, the number of the events discovered as relevant (cf. set $E_{relevant}^{inc}$ in Algorithm 5.4 on page 103) is reduced by about 14%. Second, the numbers of the predecessors and successors of each event visited during the procedure of discovering relevant events is significantly reduced.

The further heuristics We also evaluated the three heuristics from Sect. 5.11.2–5.11.4: “earliest arrival time at the destination”, “earliest departure time”, and “cycles through the departure station”. In this evaluation, we set the earliest arrival time for

each query q to $deadline(q) - 100$ minutes. The earliest departure time was set to the *earliest arrival time at the destination* minus the travel time of the longest *Pareto optimal* connection (cf. Sect. 2.2) that arrives no earlier than the *earliest arrival time at the destination* and no later than $deadline(q)$; the criteria optimized by this Pareto search were travel time and number of transfers.

We evaluated these three heuristics not individually but together. Applying them reduced the number of the events discovered as relevant (cf. set $E_{relevant}^{inc}$ in Algorithm 5.4 on page 103) by 7%. On the other side, these heuristics resulted in 3.9% queries with *suboptimal* outcomes. The loss of optimality is significant such that the heuristics from Sect. 5.11.2–5.11.4 do not confer an advantage on our incremental approach from Sect. 5.9. Therefore, we did not use them in our evaluations presented in Sect. 9.4.5–9.4.10.

9.4.12 The Discovery of Relevant Events

In Sect. 9.4.10, we presented and discussed the response times of our approach *OPT* combined with the incremental approach from Sect. 5.9. In this section, we compare our different methods of discovering relevant events from Sect. 5.8 and 5.9; we evaluated three variants of our optimal approach *OPT*:

- *OPT*: the approach *OPT* combined with the incremental approach from Sect. 5.9,
- *OPT-P*: the approach *OPT* combined with the pre-processing method from Sect. 5.8.2, and
- *OPT-PE*: the approach *OPT* combined with the pre-processing method from Sect. 5.8.2 as well as the early termination from Sect. 5.7.2.

Exclusively for this evaluation, we used all heuristics from Sect. 5.11 for all three above variants. The reason is that the set $E_{relevant}$ of all relevant events (cf. Sect. 5.8) could be very large such that the approaches *OPT-P* and *OPT-PE*—which discover all relevant events in pre-processing—would be too expensive for the evaluated problem variant from Sect. 4.3.3.1 (cf. *Latest Departure Subject to Arrival Guarantee*). Particularly for *OPT-P* and *OPT-PE*, earliest arrival and departure times need to be defined in order to significantly limit the number of discovered relevant events and facilitate an evaluation of a large number of queries.

The results of the comparison are listed in Table 9.11 (on page 175). As we see, the early termination from Sect. 5.7.2 significantly accelerates the response times. However, for the problem variant from Sect. 4.3.3.1 (cf. *Latest Departure Subject to Arrival Guarantee*), our optimal approach *OPT* from Chapter 5 obtains efficient response times when using the incremental approach from Sect. 5.9.

	function	<i>OPT-P</i>	<i>OPT-PE</i>	<i>OPT</i>
average time	Sect. 5.4	10.0	6.8	1.2
	Sect. 5.3	4.3	3.5	1.0
90%-quantile	Sect. 5.4	11.7	7.7	1.1
	Sect. 5.3	5.0	4.0	1.0
#events	Sect. 5.4	16.9	11.2	1.0
	Sect. 5.3	16.9	11.2	1.0

Table 9.11: Comparison of the response times of *OPT* combined with the different methods of discovering relevant events. All values are listed as multiples of the respective values for the approach *OPT* with our incremental approach from Sect. 5.9 and the recursive function from Sect. 5.3.

“average time”: the response time of the respective approach on average.

“90%-quantile”: the 90%-quantile of the response times of the respective approach.

“#events”: the number of the events which have been processed by Algorithm 5.1 (on page 94).

“function”: the recursive function applied to compute the outcomes.

Note: We evaluated all 80k queries with $\rho(q) = 98\%$.

*Note: Without the heuristics, the speed-up factors would be even larger since *OPT* with the incremental approach does not profit from the heuristics as much as the slower variants *OPT-P* and *OPT-PE*.*

9.5 Late-Night Connections

In this section, we present the evaluation of our approach of computing late-night connections which has been discussed in Sect. 8.4. The evaluation setup is explained in Sect. 9.5.1; the results are presented in Sect. 9.5.2.

9.5.1 Evaluation Setup

We first explain the evaluation procedure; we then discuss parameters and functions we defined for this evaluation.

The procedure In short, we detected *critical situations* in the timetable; these are situations where, after a (late) connection break, no attractive alternative continuations to the destination could be found by employing the basic Pareto Dijkstra algorithm from Sect. 2.2. For each critical situation, we applied our late-night connection search from Sect. 8.4. We then compared the late-night connections to the alternative continuations computed by the basic search from Sect. 2.2 in order to evaluate the performance of our search approach. The evaluation procedure is detailed in the following.

1. First, we generated random queries as follows. We used the timetable from Sect. 9.2.1 for the two-days period 21-22 November 2016. We generated a set of 100,000 queries where each query was as defined in Sect. 2.2. For each query

q , as the time interval $[begin(q), end(q)]$ for the departure, we randomly selected a one hour time interval between 16:00 o'clock and 23:59 o'clock on November 21. As departure and destination stations, we selected random train stations.

2. For each query, we computed the set of all Pareto optimal train connections by employing the basic Pareto Dijkstra algorithm from Sect. 2.2; the travel duration and the number of transfers were the Pareto criteria. We obtained 137,639 train connections.
3. Among the train connections computed, we then selected the train connections with late arrival times at the destination. More precisely, from the 137,639 train connections computed in the previous step, we selected each train connection with an arrival time between 16:00 o'clock on November 21 and 1:00 o'clock on November 22. We obtained 6,891 train connections.
4. For each train connection c with a late arrival time, we simulated connection breaks and detected critical situations as follows. For this purpose, for each departure event e_{dep} in c , we assumed that the train move starting with e_{dep} is not available anymore. By employing the basic Pareto Dijkstra algorithm from Sect. 2.2 (in the on-trip scenario), we subsequently searched for alternative continuation from the station of e_{dep} to the destination station of c ; we call these alternative continuations the *basic alternative continuations*. To perform this search, we set $ontrip_time(q) = sched(e_{dep}) + 1$.

For the train connection c and a basic alternative continuation computed, let t and t' respectively denote the arrival times of c and the basic alternative continuation (the arrival time at the destination station); we continued as follows.

- If a basic alternative continuation could be found such that $t' - t < 60$ minutes, we assumed that the passenger will take it, and the late-night connection search would be pointless in that situation. Thus, we continued with next departure events in the train connection c in order to find critical situations.
 - If no basic alternative continuation could be found such that $t' - t < 60$ minutes, a situation was detected where the basic search approach from Sect. 2.2 had failed to deliver an attractive alternative continuation. We collected all these critical situations; in total, we found 73,999 critical situations.
5. For each critical situation detected in the previous step, we performed an on-trip search by employing our late-night connection search from 8.4; the query was specified as for the search for basic alternative continuations (see the previous

step). The search delivered *late-night alternative continuations* to the destination station from the station where the original train connection c were assumed to be broken.

6. For each situation detected in the fourth step, we then compared the basic alternative continuations to the late-night alternative continuations. According to the Pareto Dijkstra algorithm, for each query, a set of train connections is delivered. For the comparison, from the set of the basic alternative continuations computed, we selected the alternative continuation with the earliest arrival time at the destination station; in the rest of this section, we call this alternative continuation the *basic alternative continuation*. From the set of the late-night alternative continuations computed, we selected four alternative continuations as follows:

- *Earliest*: the late-night alternative continuation with the earliest arrival time at the destination station.
- *Min. costs*: the late-night alternative continuation minimizing the tp-costs criterion (cf. Sect. 8.4.5).
- *Min. night*: the late-night alternative continuation minimizing the nighttime criterion (cf. Sect. 8.4.5).
- *Min. night**: the late-night alternative continuation minimizing the nighttime criterion among all late-night alternative continuations which had a tp-cost not greater than the tp-cost of the basic alternative continuation.

The comparison results can be found in Sect. 9.5.2.

Model and parameters To facilitate this evaluation, we specified the following parameters and functions.

- *Delay compensation*: According to [Bah16b], we assumed the following delay compensation rule. The delay compensation is calculated depending on the scheduled arrival time of the original, broken train connection at the destination and the actual arrival time of the passenger at the destination. Let c denote the original train connection that is broken in the evening hours. Let $\gamma(c)$ and δ respectively denote the price of c and the delay of the passenger for the arrival at the destination—when taking an alternative connection to the destination. We

calculated the compensation as follows:

$$\text{delay_compensation}(c) = \begin{cases} \text{€}0.0 & \text{if } \delta < 60, \\ \gamma(c)/4 & \text{if } \delta \in [60, 120), \text{ and} \\ \gamma(c)/2 & \text{if } \delta \geq 120. \end{cases}$$

- *Estimating the ticket price:* Since the functions of computing ticket prices, employed by Deutsche Bahn AG, are unknown to us, we estimated the ticket prices. Let $\beta(c)$ denote the straight-line distance (in kilometers) from the departure station to the destination station of the train connection c for which the price should be computed. We estimated the ticket price $\gamma(c)$ of the train connection c as follows:

$$\gamma(c) = \min \{ \text{€}123.0, \beta(c) \cdot 1.5 \cdot \text{€}0.30 \}.$$

The factor 1.5 is applied as a correction factor since $\beta(c)$ is a straight-line distance.

- *Hotels:* We incorporated into our model 36 hotels close to train stations [Int16]. We assumed a fixed price of €70.0 for an overnight stay at a hotel. We set the parameters *minimum stay duration* and *earliest check-out time* to 7 hours and 7:00 o'clock respectively (cf. Sect. 8.4, *Taxi transfers and hotel stays*).
- *Taxis:* Given a destination station s , we computed taxi transfers to s from each station that was in the 50km radius of s . These taxi transfers were incorporated into the search for late-night connections (cf. Sect. 8.4.4). For a taxi transfer, let β denote the straight-line distance (in kilometers) from the station where the passenger is collected to the destination station. In order to estimate the duration of the taxi transfer, we proceeded as follows. We assumed speeds of 40 km/h and 100 km/h in cities and on highways respectively. Moreover, we assumed that a 5km distance of β is always traveled in cities, and the remaining distance is covered on highways. Hence, we obtain:

$$\beta_{city} = \min\{\beta, 5\text{km}\}$$

and

$$\beta_{highway} = \max\{\beta - 5\text{km}, 0\}$$

where

$$\beta_{city} + \beta_{highway} = \beta.$$

We estimated the duration (in minutes) of the taxi transfer by the following

function:

$$duration = 10\text{min} + \left(\frac{\beta_{city}}{40\text{km/h}} + \frac{\beta_{highway}}{100\text{km/h}} \right) \cdot 1.5 \cdot 60\text{min/h}.$$

In the formula above, 10 minutes is the time required to move from the train station to the taxi as well as for entering and leaving the taxi. The factor 1.5 is applied as a correction factor since β is a straight-line distance.

The price of the taxi transfer is computed by the following function:

$$price = \text{€}2.50 + \beta \cdot 1.5 \cdot \text{€}2.00.$$

We assumed that the transportation company can provide taxi transfers in the time between 20:00 o'clock and 04:00 o'clock (on the very next day).

- *Nighttime*: We defined the time between 01:00 o'clock and 06:00 o'clock as the nighttime (cf. Sect. 8.4.5, *The nighttime criterion*); recall that traveling in this time period is undesired by passengers.

9.5.2 Results

We compared the basic alternative continuation—with the earliest arrival time at the destination—to four late-night alternative continuations as specified in the sixth step of the evaluation procedure in Sect. 9.5.1. The results are presented in Table 9.12 (on page 180) as well as in Fig. 9.6 and 9.7.

Tp-costs and nighttime Compared to the basic alternative continuation, the tp-cost (cf. Sect. 8.4.5) is reduced by 13% when taking the late-night alternative continuation with the minimum tp-cost. A higher reduction of the tp-cost could not be obtained since hotels and taxis are associated with costs (in addition to a delay compensation). Note that a overnight stay at a hotel costs €70,0; in addition, the maximum delay compensation has to be paid since the traveler reaches the destination with a delay of more than two hours.

The travel time during nighttime can be reduced by 92% when taking the late-night alternative continuations that minimizes the nighttime criterion. This is a significant improvement from the point of view of the travelers. On the other side, such alternative continuations use taxis and hotels; this consequently results in higher costs to be paid by the transportation company—more than twice as high as the earliest basic alternative continuation.

Note: In many cases, the tp-cost—to be paid by the transportation company—is mini-

	basic alt.	earliest arrival	min. costs	min. night	min. night*
tp-costs	34.97	80.79	30.54	78.63	31.36
nighttime	216	66	172	17	168
#transfers	1.83	1.05	1.93	1.55	1.78
travel time	406	234	370	411	361

Table 9.12: All numbers are average values. The values “tp-costs” and “nighttime” are as defined in Sect. 8.4.5. The values “transfers” and “travel time” are as defined in Sect. 1.2. The column “basic alt.” represents the basic alternative continuation with the earliest arrival time at the destination. The other four columns represent the late-night alternative continuations as specified in the sixth step of the evaluation procedure in Sect. 9.5.1.

mized if the passenger waits at a train station during nighttime; the delay compensation is usually lower than the costs incurred by a taxi transfer or a overnight stay in a hotel.

Run times The run time of the basic search approach from Sect. 2.2 was 183.5ms on average. In contrast, our late-night connection search required 561.4ms on average. The additional Pareto criteria slow down the search. However, the late-night search is still efficient and practicable.

Discussion We believe that passengers prefer either the late-night connection with the earliest arrival time at the destination or the late-night connection with the minimum travel time during nighttime. Both are more expensive for the transportation company compared to the basic alternative continuation. The late-night alternative continuation with the minimum tp-costs is an attractive solution in many cases—for both the passengers as well as the transportation companies.

In summary, as claimed, by employing the late-night connection search, a set of attractive alternative continuations can be computed. The transportation company has the possibility to decide which options are provided to the passengers. Last, we note that the results of our computational study significantly depend on the parameters and the functions specified in Sect. 9.5.1 (cf. *Model and parameters*).

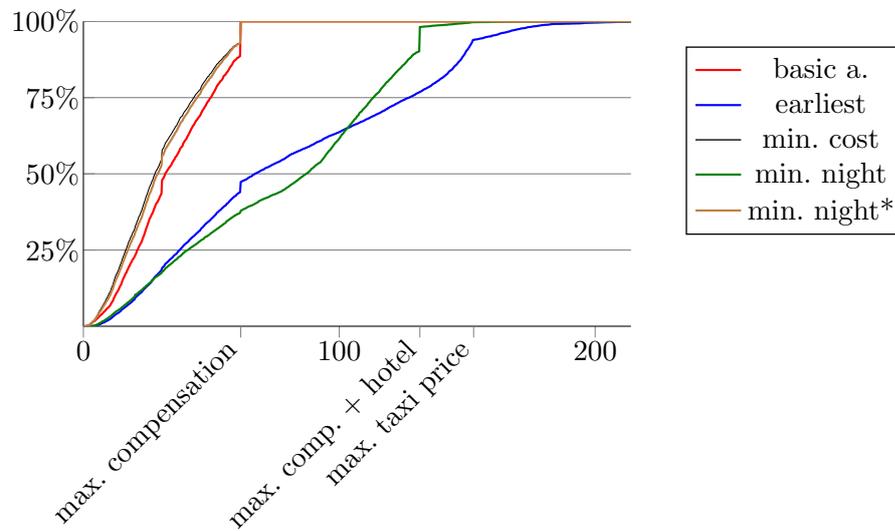


Figure 9.6: The x-axis represents the tp-cost value (cf. Sect. 8.4.5). The y-axis represents the percentage of the alternative continuations with a tp-cost lower than or equal to the respective x-value. The graph “basic a.” represents the basic alternative continuation with the earliest arrival time at the destination. The other graphs represent the late-night alternative continuations as specified in the sixth step of the evaluation procedure in Sect. 9.5.1. Since the maximum ticket price equals €123,0, the maximum delay compensation equals €61.50. The longest taxi transfer has a distance of 50km; this results in a maximum price of €152.50. The maximum tp-cost is paid for a connection with the maximum taxi price and the maximum delay compensation; it equals €214.

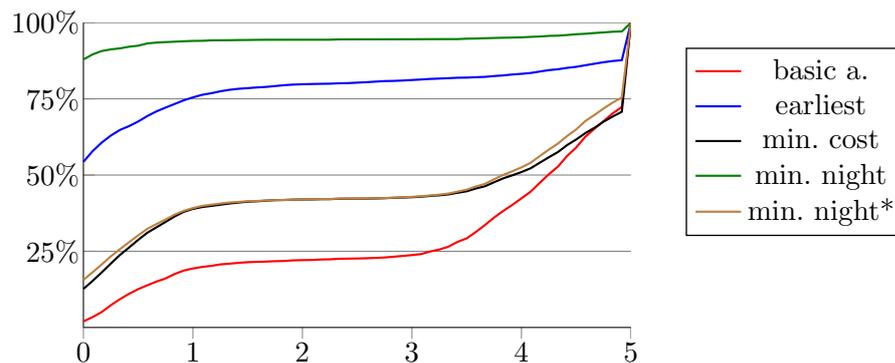


Figure 9.7: The x-axis represents the nighttime value (cf. Sect. 8.4.5). The y-axis represents the percentage of the alternative continuations with a nighttime value lower than or equal to the respective x-value. The graph “basic a.” represents the basic alternative continuation with the earliest arrival time at the destination. The other graphs represent the late-night alternative continuations as specified in the sixth step of the evaluation procedure in Sect. 9.5.1.

Chapter 10

Conclusion and Outlook

10.1 Conclusion

In this work, we focused on the computation of reliable train journeys.

Reliability assessments We first introduced a method of computing the probability of the success of a transfer activity in a train connection. We then extended this method to train connections that potentially contain more than one transfer activity. We measured the probability that no transfer activity in the train connection becomes infeasible and passengers can reach their destination.

Our method of assessing the reliability of train connections uses probability distributions for random event times. For the calculation of these probability distributions, we use formulas presented by [BGMHO11]. We demonstrated how to efficiently process the departure and arrival events in the timetable in order to compute their probability distributions. We also presented an efficient procedure of updating precomputed probability distributions according to current delays in the railway network.

To evaluate the accuracy of our reliability assessments, we computed the probability of the success of each train connection in a large set. After the real times of the events in the timetable became known, we checked the feasibility of each train connection. This computational study revealed that our reliability assessments had been realistic and accurate. However, the quality of our method depends on the quality of the input data, particularly the probability distributions for the random train move durations as well as the maximum waiting times.

As demonstrated in our computational study, the run time of our method for reliability assessment is on the order of microseconds on a desktop PC. It is fast and efficient even for large railway networks such as in Germany. The timetable evaluated had millions of departure and arrival events, and comprised short- and long-distance trains as well as local public transport such as buses and streetcars.

We finally demonstrated that reliability is an important criterion of train connections. Train connections with low reliabilities often break. This fact motivated us to develop an efficient approach to facilitate reliable journeys.

Highly reliable complete connections We extended the definition of train connection to complete connection; the latter is a train connection along with alternative continuations. More precisely, for each arrival event and each real time expected for that event, a complete connection has an instruction concerning how to continue the journey. The optimization of the alternative continuations at the booking stage facilitates computing complete connections which are highly reliable and have attractive travel times as well.

We presented an efficient approach to computing a complete connection—for a given query—that guarantees a probability of success in reaching the destination prior to the specified deadline. Subject to this guarantee, our approach maximizes the scheduled departure time at the departure station. Moreover, we discussed further problem variants which can be solved by our approach. To the best of our knowledge, we are the first to address this fundamental problem and to present an optimal, efficient solution.

Our approach can be combined with two different recursive functions. One of them is faster and allows the optimization of the expected number of transfers as a secondary criterion. The other one is more compute-intensive but facilitates the optimization of a weighted sum of the travel time and the expected number of transfers. We translated these recursive functions to a dynamic programming approach. Moreover, we discussed the discovery of events in the timetable which are relevant for a query; the expensive computations can be limited to these events. Eventually, we obtained the required efficiency by an approach that incrementally discovers relevant events and computes the optimal complete connection.

In our extensive computational study, we first demonstrated that it is necessary to regard the probability of the success as well as the alternative continuations of each train connection at the booking stage. We then demonstrated that our approach to computing complete connections is better than conservative methods which incorporate buffer times at critical points in the journey. The evaluations have revealed that it is well worth the effort to compute optimal complete connections as presented in this work. Our approach computes journeys with attractive travel times and numbers of transfers, particularly for busy individuals.

Our approach delivers the optimal outcome for any query that allows one. The average run time of two seconds per query is indicative that our approach is appropriate from a practical point of view. It can be integrated into timetable information systems in order to answer large numbers of queries per day.

Travel guidance component For highly reliable complete connections, we used the search component and the travel guidance component. As described above, the search component computes the optimal complete connection for a given query so that—given all information available at the time of planning—the required probability of success is met. The probability of the success of a complete connection can consequently be guaranteed only at the time when the search component is executed. Whenever the network state has changed, for example due to delays, the probability of the success of a formerly computed complete connection could be affected. To overcome this problem, the travel guidance component checks the computed complete connection and, if necessary, updates it by a re-invocation of the search component.

Moreover, the travel guidance component is responsible for a comprehensive presentation of the complete connections to travelers. We demonstrated sample implementations of the travel guidance component. The travel guidance component can also be combined with alternative approaches that compute train connections or complete connections; we discussed such approaches in Chapter 7.

Further search approaches We also discussed further methods of searching for reliable train connections and complete connections. Reliability can be optimized as a Pareto criterion for the Pareto search. In this way, we obtain train connections which have higher probabilities of success compared to train connections which are computed without regard for reliability. Moreover, we introduced an alternative approach to the computation of complete connections. However, all alternative approaches which we discussed were inferior to our optimal approach to the search for complete connections.

Reliable intermodal connections We extended our reliability assessment method to intermodal connections. We discussed predictions for the availability of bike sharing and car sharing services; the predictions were based on historical availability data. We then discussed methods of computing reliable intermodal connections: we either optimized reliability as a Pareto criterion or excluded unreliable rides by individual modes of transport. We also discussed an approach to finding reliable intermodal complete connections with alternative continuations. However, this approach is not optimal; the computation of optimal intermodal complete connections is left to future research as discussed in 10.2.

Late-night connections We addressed the problem of connection breaks late at night. The challenge in this regard is that the destination station cannot be reached by public transport prior to the end of operations. Our solution was to search for intermodal connections which potentially use taxi rides to the destination; alternatively,

the passenger is recommended to visit a hotel. The costs—to be paid by the transportation company—and the travel time during the night were minimized as additional criteria for the Pareto Dijkstra algorithm. The computational study demonstrated that our approach could help transportation companies to provide better services to their passengers. It could be applied as a tool in order to find numerous attractive options in cases of connection breaks late at night. The transportation companies could still decide which options are finally offered to the passengers.

10.2 Outlook

In this section, we discuss potential improvements and extensions to the approaches presented in this work.

Better reliability assessments The distributions for the departure and arrival events of the trains and, consequently, the probability of success could be computed more accurately by the inclusion of further factors influencing train delays. Our model would particularly benefit from data on the railroad network's infrastructure. As an example, some stations are connected by only one track which must be shared by trains traveling in both directions. Delays in one direction may cause delays in the other. Once the necessary data is available, dependencies like these could be seamlessly integrated into our approach and modeled as additional random variables. Alternatively, they could be taken into account when computing the probability distributions for the random durations of the train moves or the random prepared times.

An extension of the time-dependent graph model is required in order to support special transfer times for specific train pairs; for example, for trains at tracks which are close to each other. These more accurate minimum transfer times would improve the accuracy of the reliability assessments.

Last, in the timetable, there are run-through services and trains which are coupled together. These rules cause further dependencies which must be integrated into our model. The formulas to compute the probability distributions for the events must be extended in order to support these exceptional cases.

Further speed-up techniques The search for optimal complete connections could be accelerated by developing further speed-up techniques. We noticed that the procedure of discovering relevant events requires a significant portion of the overall run time. An investigation of more elaborate heuristics and speed-up techniques could accelerate the discovery of relevant events. Moreover, through techniques such as parallelization, we could accelerate the computations of the probabilities which are required to

construct the complete connections.

Support for local public transport in complete connections In Sect. 9.4, we evaluated an implementation of our approach to the search for optimal complete connections. In our implementation, we used the time-expanded graph model, and limited the timetable to short- and long-distance trains. To support local public transport, the time-dependent graph model is more appropriate. However, supporting dense local public transport is a challenge since the number of events and, consequently, the number of random variables increases by orders of magnitude. A potential solution is to contract high-frequency train (or bus) moves which can be used periodically. This is nevertheless insufficient for the management of such large networks, and this heuristic could result in suboptimal outcomes. The support for local public transport is left to future research.

Optimal intermodal complete connections The optimal approach to computing complete connections could be extended to intermodal complete connections. For example, after the arrival at a station close to the destination location, depending on the respective situation, the intermodal complete connection could suggest walking to a bike sharing station to collect a bike, or taking a bus to a station which is closer to the destination location and walking from there. We believe that the extension of our algorithm to intermodal complete connections can be accomplished without any major obstacles; analogously to the extension of the Pareto Dijkstra algorithm, we could add additional edges to the graph which represent the use of individual modes of transport. However, the support for local public transport is a prerequisite to find attractive intermodal complete connections.

Generalization and application to other problems The problem addressed may be seen as a novel generic algorithmic problem in directed acyclic graphs. The approach we presented is evidently of much broader practical value, especially in the fields of transportation, logistics, and scheduling. For example, in stochastic scheduling, we could represent project states and milestones by graph nodes and progress steps by graph arcs. Jobs have stochastic durations. With our dynamic programming approach, we could solve the stochastic scheduling problem and guarantee a probability of success. However, this is left to future research.

Appendix A

Appendices

A.1 Reliability of Transfers

In our publication [KSWZ12], we addressed the computation of the probability of success of a transfer. For a transfer activity (e_{arr}, e_{dep}) and its minimum transfer time δ , we introduced a formula to compute the probability $P(X_{e_{arr}} + \delta \leq X_{e_{dep}})$. In the following, we demonstrate that the formula from [KSWZ12] and the formula from Eq. 3.6.2 are equivalent.

Terminology First, we discuss the terminology from [KSWZ12]. In that work, the events e_{arr} and e_{dep} involved in a transfer activity (e_{arr}, e_{dep}) are denoted by dep and $arr_{tr_1, s}$ respectively. The probability $P(X_{e_{arr}} + \delta \leq X_{e_{dep}})$ from Sect. 3.6 is denoted by $P(X_{dep} = d)$ (see [KSWZ12], Sect. 3.2.1). The set FD from [KSWZ12] equals

$$FD = E_{arr}^{depend}(e_{dep}) \setminus \{e_{arr}, \widehat{e_{arr}}\}$$

where (e_{arr}, e_{dep}) is the transfer activity—for which we compute the probability of success—and $\widehat{e_{arr}}$ is the arrival event of the dwell activity $(\widehat{e_{arr}}, e_{dep}) \in DW$. Last, the probability $P_{noWaitingForFeeders}(tr, s, d)$ from [KSWZ12] equals $\psi_{e_{arr}}(e_{dep}, d) - \psi_{e_{arr}}(e_{dep}, d - 1)$.

Proof First, we define a variable $b \in \{0, 1\}$ where

$$b = \begin{cases} 1 & \text{if } d \leq d_{\max}, \\ 0 & \text{otherwise} \end{cases}$$

and d_{\max} is the latest departure time for e_{dep} up to which e_{dep} waits for e_{arr} . Let $\widehat{e_{arr}} \in E_{arr}$ be the arrival event of the dwell activity $(\widehat{e_{arr}}, e_{dep}) \in DW$. Let $\psi_{e_{arr}}(e_{dep}, d)$

be as defined in Sect. 3.6. In addition, we define $\psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d)$ that is computed with the formula introduced for $\psi(e_{dep}, d)$ (in Eq. 3.5.2) with the difference that the product in the formula is over all events in the set $E_{arr}^{depend}(e_{dep}) \setminus \{e_{arr}, \widehat{e_{arr}}\}$. Observe that $\psi_{e_{arr}}(e_{dep}, d)$ from Eq. 3.5.2 (on page 45) equals

$$\psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) \cdot P(X_{e_{arr}} + \min_dur(e_{arr}, e_{dep}) \leq d)$$

since the probability

$$P(X_{e_{arr}} + \min_dur(e_{arr}, e_{dep}) > \max\{d, d^{latest}(e_{arr}, e_{dep})\})$$

equals 0 for $(\widehat{e_{arr}}, e_{dep}) \in DW$. In the following, let δ be equal to $\min_dur(e_{arr}, e_{dep})$.

$$\begin{aligned} & P(X_{e_{arr}} + \delta \leq d \cap X_{e_{dep}} = d) \\ &= \left[P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}}(e_{dep}, d) - P(X_{e_{arr}} + \delta \leq d-1) \cdot \psi_{e_{arr}}(e_{dep}, d-1) \right] \cdot b \\ &\quad + \left[P(X_{e_{arr}} + \delta \leq d) \cdot [\psi_{e_{arr}}(e_{dep}, d) - \psi_{e_{arr}}(e_{dep}, d-1)] \right] \cdot (1-b) \\ &= P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}}(e_{dep}, d) \cdot b \\ &\quad - P(X_{e_{arr}} + \delta \leq d-1) \cdot \psi_{e_{arr}}(e_{dep}, d-1) \cdot b \\ &\quad + P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}}(e_{dep}, d) \cdot (1-b) \\ &\quad - P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}}(e_{dep}, d-1) \cdot (1-b) \\ &= P(X_{e_{arr}} + \delta \leq d) \cdot [\psi_{e_{arr}}(e_{dep}, d) \cdot b + \psi_{e_{arr}}(e_{dep}, d) \cdot (1-b) - \psi_{e_{arr}}(e_{dep}, d-1) \cdot (1-b)] \\ &\quad - P(X_{e_{arr}} + \delta \leq d-1) \cdot \psi_{e_{arr}}(e_{dep}, d-1) \cdot b \\ &= P(X_{e_{arr}} + \delta \leq d) \cdot [\psi_{e_{arr}}(e_{dep}, d) - \psi_{e_{arr}}(e_{dep}, d-1) \cdot (1-b)] \\ &\quad - P(X_{e_{arr}} + \delta \leq d-1) \cdot \psi_{e_{arr}}(e_{dep}, d-1) \cdot b \\ &= P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}}(e_{dep}, d) \\ &\quad - P(X_{e_{arr}} + \delta \leq d-1) \cdot \psi_{e_{arr}}(e_{dep}, d-1) \cdot b \\ &\quad - P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}}(e_{dep}, d-1) \cdot (1-b) \\ &= P(X_{e_{arr}} + \delta \leq d) \cdot P(X_{\widehat{e_{arr}}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) \\ &\quad - P(X_{e_{arr}} + \delta \leq d-1) \cdot P(X_{\widehat{e_{arr}}} + \delta \leq d-1) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d-1) \cdot b \\ &\quad - P(X_{e_{arr}} + \delta \leq d) \cdot P(X_{\widehat{e_{arr}}} + \delta \leq d-1) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d-1) \cdot (1-b) \\ &= P(X_{\widehat{e_{arr}}} + \delta \leq d) \cdot P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) \\ &\quad - P(X_{\widehat{e_{arr}}} + \delta \leq d-1) \cdot P(X_{e_{arr}} + \delta \leq d-1) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d-1) \cdot b \\ &\quad - P(X_{\widehat{e_{arr}}} + \delta \leq d-1) \cdot P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d-1) \cdot (1-b) \end{aligned}$$

$$\begin{aligned}
 &= P(X_{\widehat{e_{arr}}} + \delta = d) \cdot P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) \\
 &\quad + P(X_{\widehat{e_{arr}}} + \delta \leq d - 1) \cdot P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) \\
 &\quad - P(X_{\widehat{e_{arr}}} + \delta \leq d - 1) \cdot P(X_{e_{arr}} + \delta \leq d - 1) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \cdot b \\
 &\quad - P(X_{\widehat{e_{arr}}} + \delta \leq d - 1) \cdot P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \cdot (1 - b) \\
 &= P(X_{\widehat{e_{arr}}} + \delta = d) \cdot P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) \\
 &\quad + P(X_{\widehat{e_{arr}}} + \delta < d) \cdot [P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) \\
 &\quad\quad - P(X_{e_{arr}} + \delta \leq d - 1) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \cdot b \\
 &\quad\quad - P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \cdot (1 - b)] \\
 &= P(X_{\widehat{e_{arr}}} + \delta = d) \cdot P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) \\
 &\quad + P(X_{\widehat{e_{arr}}} + \delta < d) \cdot \xi
 \end{aligned} \tag{A.1.1}$$

The value ξ is defined as follows.

$$\begin{aligned}
 \xi &= P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) \\
 &\quad - P(X_{e_{arr}} + \delta \leq d - 1) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \cdot b \\
 &\quad - P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \cdot (1 - b) \\
 &= P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) \\
 &\quad - P(X_{e_{arr}} + \delta \leq d - 1) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \cdot b \\
 &\quad - P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \cdot (1 - b) \\
 &\quad + P(X_{e_{arr}} + \delta = d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \cdot b \\
 &\quad - P(X_{e_{arr}} + \delta = d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \cdot b \\
 &= P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) \\
 &\quad - P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \cdot b \\
 &\quad - P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \cdot (1 - b) \\
 &\quad + P(X_{e_{arr}} + \delta = d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \cdot b \\
 &= P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) \\
 &\quad - P(X_{e_{arr}} + \delta \leq d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \\
 &\quad + P(X_{e_{arr}} + \delta = d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \cdot b \\
 &= P(X_{e_{arr}} + \delta \leq d) \cdot [\psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) - \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1)] \\
 &\quad + b \cdot P(X_{e_{arr}} + \delta = d) \cdot \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) \\
 &= P(X_{e_{arr}} + \delta \leq d) \cdot [\psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) - \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1)] \\
 &\quad + b \cdot P(X_{e_{arr}} + \delta = d) \cdot [\psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1) + \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) - \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d)]
 \end{aligned}$$

$$\begin{aligned}
 &= P(X_{e_{arr}} + \delta \leq d) \cdot [\psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) - \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1)] \\
 &\quad + b \cdot P(X_{e_{arr}} + \delta = d) \cdot [\psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) - [\psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d) - \psi_{e_{arr}, \widehat{e_{arr}}}(e_{dep}, d - 1)]]
 \end{aligned}
 \tag{A.1.2}$$

In [KSWZ12], we distinguished between the cases: “departure on time”, “departure during the waiting interval”, and “departure after the waiting interval”. Eq. A.1.1 demonstrates the equality to the formula presented for the case “departure during the waiting interval” (see [KSWZ12], Sect. 3.2.1.2). In Eq. A.1.2, we demonstrated that ξ equals $P_{waiting}$ from [KSWZ12] (see appendix A.2 in that work). The formulas for the other two cases are simplifications of the formula for “departure during the waiting interval” and can be obtained trivially.

A.2 Travel Guidance Component

In Fig. A.2 (on page 193), we present screenshots of our implementation of the travel guidance component from Sect. 6.1.1. Our implementation of the mobile application from Sect. 6.1.2 is demonstrated in Fig. A.3 (on page 194). Finally, Fig. A.4 (on page 195) presents a visualization of two intermodal connections (with bike sharing) computed as discussed in Sect. 8.3. The first and the second intermodal connection use bike sharing services with reliability assessments of 3 and 4 respectively. The former is classified as unreliable while the later is assumed to be reliable.

A.3 Generated Probability Distributions

In Fig. A.1 (on page 192), we illustrate examples of generated probability distributions which we used in our computational study in Sect. 9.4.

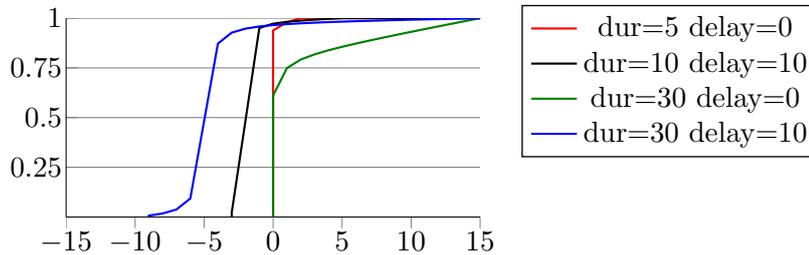


Figure A.1: Examples of generated probability distributions describing random train move durations. The x-axis represents the deviation (in minutes) from the scheduled duration of the train move. The y-axis represents the probability (cumulative). All train moves with the same scheduled duration and the same delay of their departure events are described by the same probability distribution. In the legend, “dur” and “delay” respectively denote the scheduled duration and the departure delay of the train moves for which the random durations are described by the respective probability distribution.

A.3. GENERATED PROBABILITY DISTRIBUTIONS

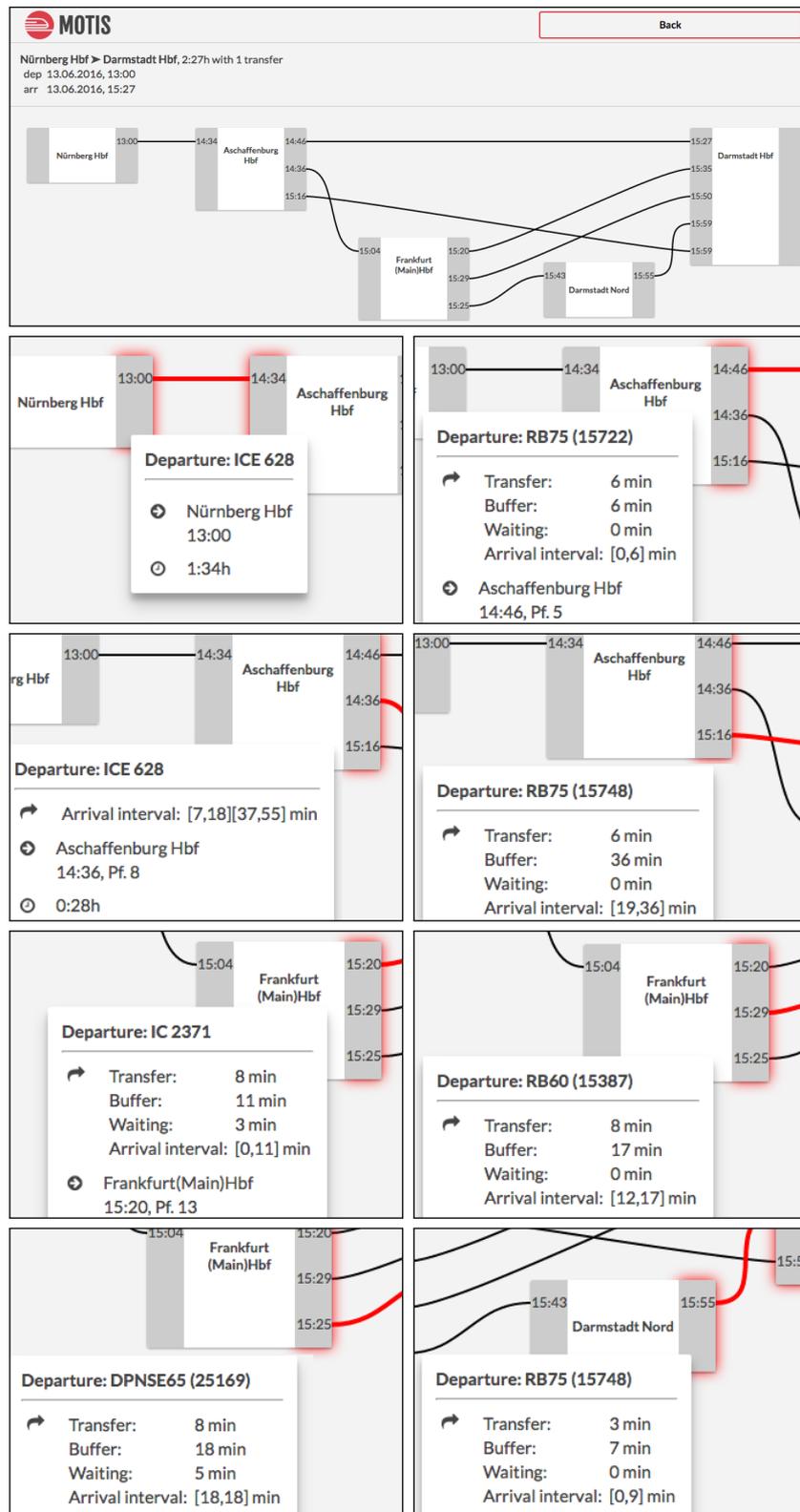


Figure A.2: Travel guidance component: the web interface

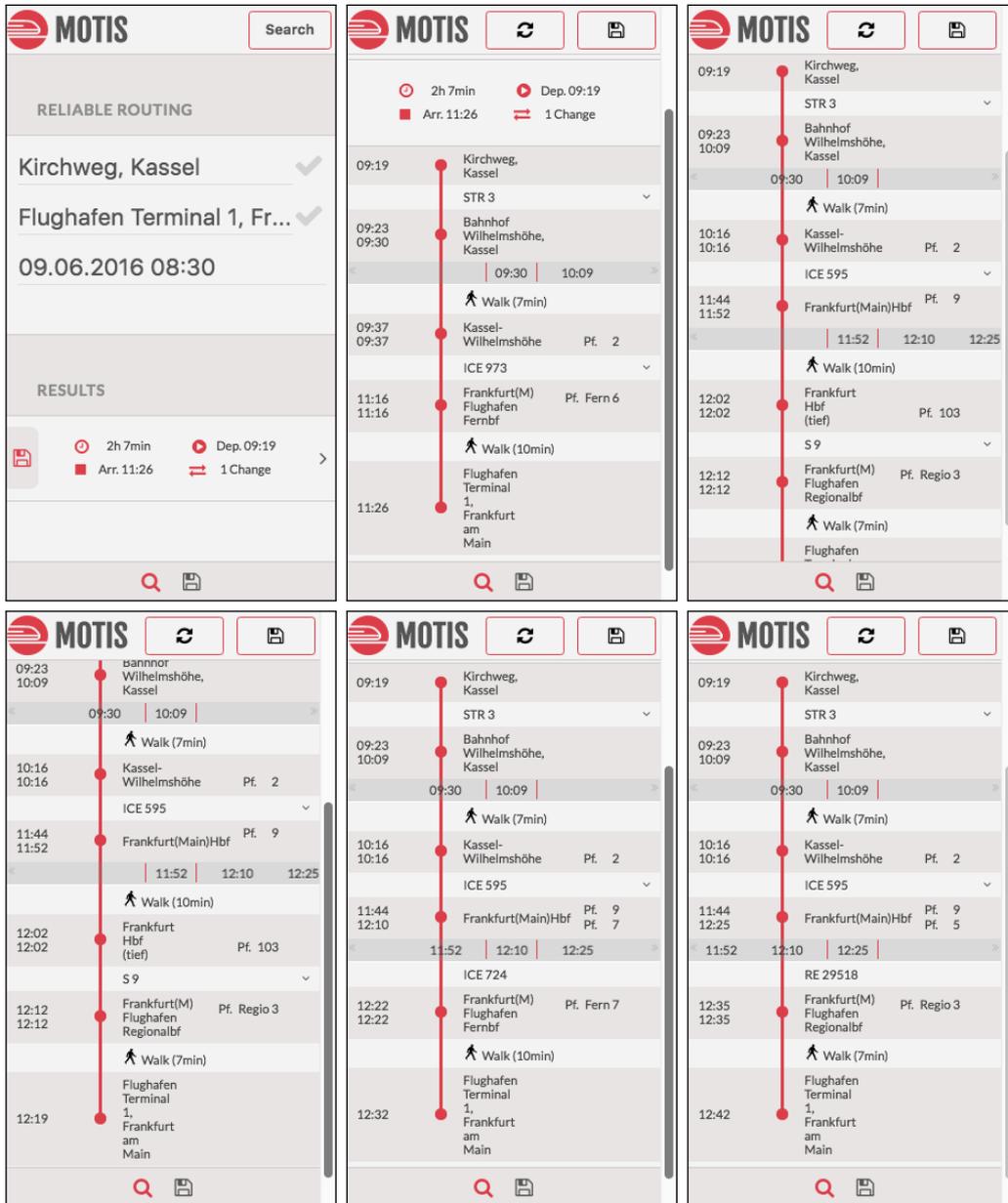


Figure A.3: Travel guidance component: the mobile application

A.3. GENERATED PROBABILITY DISTRIBUTIONS

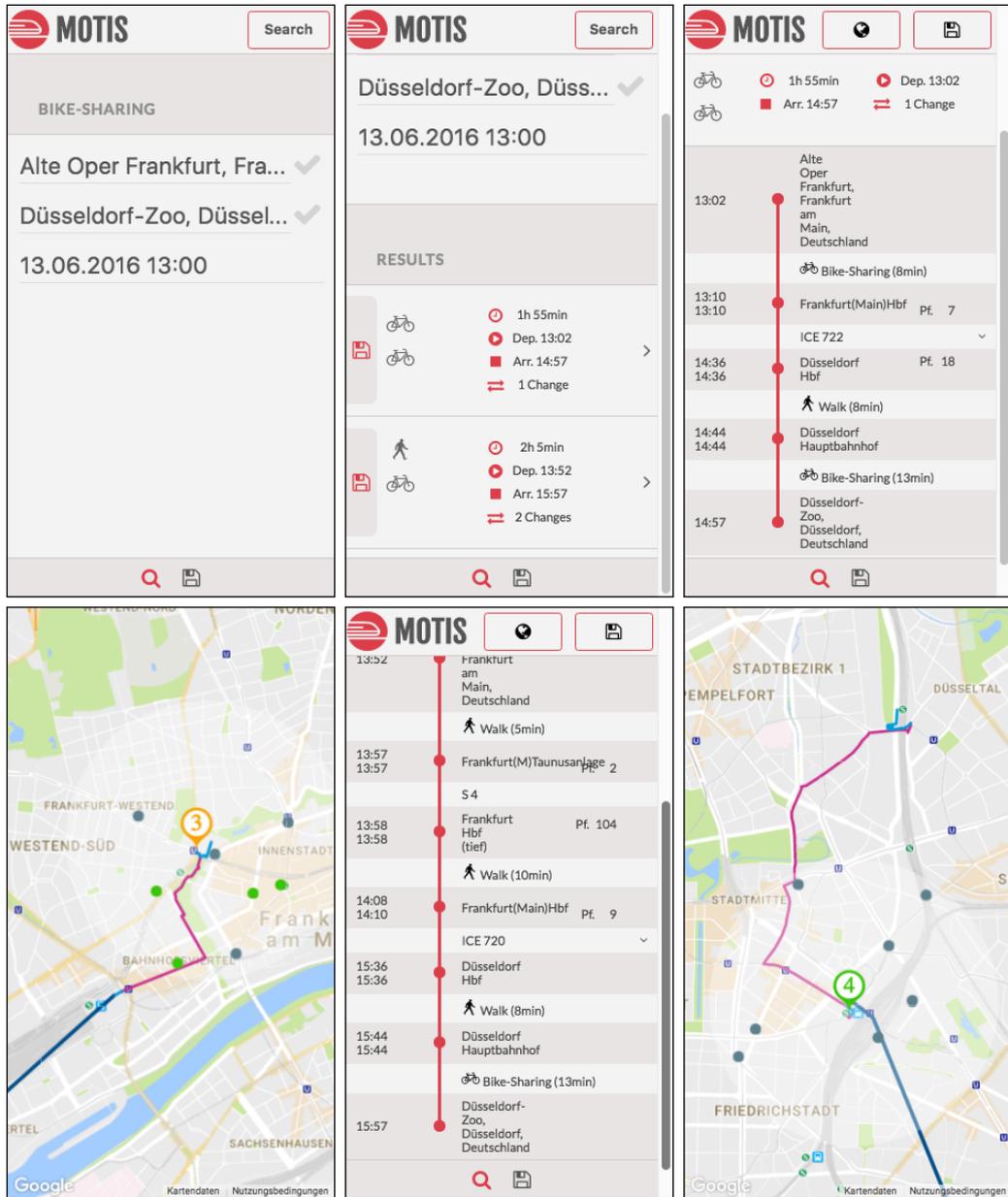


Figure A.4: Travel guidance component: intermodal connections

Nomenclature

Timetable

c	A train connection, page 7
C	A set of train connections, page 7
dw	A dwell activity in the set DW , page 6
DW	The set of all dwell activities in the timetable, page 6
e	a departure or an arrival events in the set E , page 6
$e_{arr}^{s,tr}$	The arrival event of the train tr at the station s , page 6
$e_{dep}^{s,tr}$	The departure event of the train tr at the station s , page 6
E	The set of all departure and arrival events in the timetable, page 6
E_{arr}	The set of all arrival events in the timetable, page 6
E_{arr}^s	The set of all arrival events at the station $s \in S$, page 6
E_{dep}	The set of all departure events in the timetable, page 6
E_{dep}^s	The set of all departure events at the station $s \in S$, page 6
$(e_{arr}^{s,tr1}, e_{dep}^{s,tr2})$	A transfer activity in the set $TRANS$ from the train $tr1$ to the train $tr2$ at the station s , page 8
$(e_{arr}^{s1,tr1}, e_{dep}^{s2,tr2})$	A transfer activity with a walk trip in the set $TRANS$ from the train $tr1$ at the station $s1$ to the train $tr2$ at the station $s2$, page 9
$(e_{arr}^{s2,tr}, e_{dep}^{s,tr})$	A dwell activity in the set DW of the train tr at the station s , page 6

NOMENCLATURE

$(e_{dep}^{s,tr}, e_{arr}^{s2,tr})$	A train move in the set TM of the train tr from the station s to the station $s2$, page 6
$max_wait(e_{arr}^{s,tr}, e_{dep}^{s,tr2})$	The maximum waiting time of $tr2$ for tr at the station s , page 10
$min_dur(e_{arr}^{s,tr1}, e_{dep}^{s,tr2})$	The minimum transfer time required for the transfer activity $(e_{arr}^{s,tr1}, e_{dep}^{s,tr2})$, page 8
$min_dur(e_{dep}^{s,tr}, e_{arr}^{s2,tr})$	The scheduled duration of the train move $(e_{dep}^{s,tr}, e_{arr}^{s2,tr}) \in TM$, page 6
$min_dur(tm)$	The scheduled duration of the train move $tm \in TM$, page 6
$min_dur(walk)$	The duration of the walk trip $walk$, page 7
$real(e)$	The real event time of the event e , page 10
s	A station in the set S , page 5
S	The set of all stations in the timetable, page 5
$sched(e)$	The scheduled event time of the event $e \in E$, page 6
(s, s')	A walk activity from the station s to the station s' , page 7
t_{arr}	A point in time in the set $TIMES$, page 5
t_{dep}	A point in time in the set $TIMES$, page 5
t	A point in time in the set $TIMES$, page 5
$TIMES$	The set of all points in time in the timetable's validity period, page 5
TM	The set of all train moves in the timetable, page 6
tm	A train move in the set TM , page 6
TR	The set of all trains in the timetable, page 5
tr	A train in the set TR , page 5
$TRANS$	The set of all transfer activities in the timetable, page 8
$trans$	A transfer activity in the set $TRANS$, page 8
$trip$	A train trip, page 7
TT	A timetable, page 5

WALKS Set of all walk trips in the timetable, page 7

walk A walk trip in the set *WALKS*, page 7

Time-Expanded Graph Model

A The set of all edges in the graph, page 15

G The graph, page 15

sched(v) The scheduled time of the event represented by the node *v*, page 16

V The set of all nodes in the graph, page 15

v A node in the set *V*, page 16

V_{arr} The set of all arrival nodes in the graph, page 16

v_{arr} An arrival node in the graph, page 16

V_{dep} The set of all departure nodes in the graph, page 16

v_{dep} A departure node in the graph, page 16

V_{arr}^s The set of all arrival nodes at the station *s*, page 16

V_{dep}^s The set of all departure nodes at the station *s*, page 16

(v, w) An edge in the set *A*, page 16

(v_{arr}, v_{dep}) A dwell edge or a special-transfer edge in the set *A*, page 16

(v_{dep}, v_{arr}) A train edge in the set *A*, page 16

w A node in the set *V*, page 16

Time-Dependent Graph Model

A The set of all edges in the graph, page 19

G The graph, page 19

$length_{(v,w)}(t)$ The time-dependent length of the edge (v, w) , page 20

r A route in the set *R*, page 18

R The set of all routes in the timetable, page 18

TR(r) The set of all trains on the route *r*, page 18

V The set of all nodes in the graph, page 19

v A node in the set V , page 19

(v, w) An edge in the set A , page 19

w A node in the set V , page 19

Pareto Dijkstra

$[begin(q), end(q)]$ A time interval for the departure at $dep_st(q)$, page 21

$cost_i$ A cost value from the tuple of costs, page 23

$dep_st(q)$ The departure station of the query q , page 21

$dest_st(q)$ The destination station of the query q , page 21

l A search label, page 23

$ontrip_time(q)$ A point in time for the departure at $dep_st(q)$, page 21

q A query, page 21

Stochastic Model

$E_{arr}^{depend}(e_{dep}^{s,tr})$ The set of all arrival events that depend on $e_{dep}^{s,tr}$, page 43

$e_{artific}^{s,tr}$ An artificial event modeling the random prepared time of a departure event, page 43

$\xi(e_{dep})$ The probability density function for the random prepared time of the event e_{dep} , page 36

$\xi(e_{dep}, t)$ The probability that the real prepared time of e_{dep} equals t , page 36

$\phi(e)$ The probability density function for the random event time of the event e , page 35

$\phi(e, t)$ The probability of e taking place at time t , page 35

$P(X_e = t)$ The probability of e taking place at time t , page 35

$\psi(e_{dep}, t_{dep})$ The probability that, for the departure event e_{dep} and at the time t_{dep} , the necessary condition from Sect. 3.5.2.1 (cf. *Informal description*) is fulfilled, page 45

$\psi_{e_{arr}}(e_{dep}, t_{dep})$	The probability that, for the departure event e_{dep} as well as at the time t_{dep} , the necessary condition from Sect. 3.5.2.1 (cf. <i>Informal description</i>) is fulfilled given that e_{arr} is excluded from the set $E_{arr}^{depend}(e_{dep})$, page 47
$supp(\phi(e))$	The support of the function $\phi(e)$, page 35
$t_{dep}^{latest}(e_{arr}, e_{dep})$	The latest real event time of e_{dep} that can be caused by $e_{arr} \in E_{arr}^{depend}(e_{dep})$ as defined in Eq. 3.5.1, page 44
$\theta(tm, t_{dep})$	The probability density function for the random duration of the train move tm given that the departure event of tm takes place at the time t_{dep} , page 36
$\theta(tm, t_{dep}, t_{arr})$	The probability that the departure event of the train move tm takes place at the time t_{dep} and its arrival event takes place at the time t_{arr} , page 36
X_e	The random variable modeling the random time of the event e , page 35
$X_{e_{dep}}^p$	The random variable modeling the random prepared time of e_{dep} , page 36
X_{tm}	The random variable modeling the random duration of the train move tm , page 36

Complete Connections

c	A complete connection, page 72
C	A set of complete connections, page 72
$C(q)$	The set of all complete connections for a query q , page 73
$C_{best}(q)$	The set of the best complete connections for the query q , page 81
$deadline(q)$	A deadline for the arrival time at $dest_st(q)$ of the query q , page 71
$dep_st(q)$	The departure station of the query q , page 71
$dep_time(c)$	The scheduled departure time of the complete connection c , page 72

$dest_st(q)$	The destination station of the query q , page 71
$\hat{e}_{dep}^{cont}(e_{arr}, t)$	The succeeding departure event selected for the arrival event e_{arr} and the time t , page 79
$\tilde{E}_{dep}^{dep_st(q)}$	A set of relevant events at $dep_st(q)$ as defined in Sect. 5.8.1, page 97
$\tilde{E}_{arr}^{dest_st(q)}$	A set of relevant events at $dest_st(q)$ as defined in Sect. 5.8.1, page 97
e_{dep}^{init}	The initial departure event of a complete connection, page 78
$E_{relevant}$	The set of all events that are relevant for a given query, page 96
$E_{relevant}^{inc}$	A set of relevant events as discovered till the current iteration of the incremental approach from Sect. 5.9, page 100
$E_{succ}(e, t)$	The set of all successor events of the event e for the time t , page 84
$E_{succ}^{max}(e, t)$	The set of all events in $E_{succ}(e, t)$ that maximize the probability from Eq. 5.4.2 (on page 88) for the event e and the time t , page 88
$\mu(e)$	The expected number of transfers which are required to reach $dest_st(q)$ from the event e , page 86
$\rho_{req}(q)$	The required probability of success of the query q , page 71
$\rho(c)$	The probability of the success of the complete connection c , page 72
$\rho(e)$	The probability of success for the event e , page 77
$\varrho(e, t)$	The probability that from e the destination station $dest_st(q)$ can be reached no later than $deadline(q)$ given that e takes place at the time t , page 85
$\varrho(e, t, t_{arr})$	The probability that from e the destination station $dest_st(q)$ can be reached at the time t_{arr} given that e takes place at the time t , page 87
$\varrho(e, t, \tilde{e}, t_{arr})$	The probability that from e the destination station $dest_st(q)$ can be reached at the time t_{arr} given that e takes place at the time t and the journey is continued via \tilde{e} , page 88

Reliable Intermodal Connections

$[begin(q), end(q)]$	A time interval for the departure at $dep_st(q)$, page 136
IR_{end}	The set of all individual routes for the end parts of individual connections for a given query, page 137
IR_{init}	The set of all individual routes for the initial parts of individual connections for a given query, page 137
q	A query, page 136
(t_1, t_2, a)	A reliability assessment tuple, page 137

List of Algorithms

2.1	Pareto Dijkstra	29
3.1	Computing the probability density functions of events	54
3.2	Updating the probability density functions of events	58
5.1	Computing optimal complete connections by dynamic programming	94
5.2	Relevant events: backward marking procedure	99
5.3	Relevant events: forward marking procedure	99
5.4	Discovering and processing relevant events incrementally	103

List of Figures

1.1	Example of a train trip	6
1.2	Example of a train connection	8
1.3	Example of an intermodal connection	12
1.4	Station-based individual modes of transport	12
2.1	The time-expanded graph model	16
2.2	The time-dependent graph model	19
2.3	Ranges of individual modes of transport	27
3.1	Interdependencies between departure and arrival events	43
3.2	The probability of the success of a transfer activity	48
3.3	The probability of the success of a train connection	52
4.1	The idea of complete connections	68
4.2	The idea of optimizing complete connections	69
5.1	The structure of complete connections	79
5.2	Assumption for the dynamic programming approach	96
6.1	The travel guidance component: a web interface	115
6.2	The travel guidance component: a mobile application	115
6.3	The travel guidance component: updating a complete connection	121
8.1	The structure of late-night connections	148
9.1	Evaluation: reliability assessments (histogram)	159
9.2	Evaluation: reliability assessments (cumulative plot)	160
9.3	Evaluation: reliability assessments (ROC curve)	161
9.4	Evaluation: the departure times of the outcomes	168
9.5	Evaluation: the price of the on-time arrival guarantee	170
9.6	Evaluation: late-night connections (tp-costs)	181
9.7	Evaluation: late-night connections (nighttime)	181

LIST OF FIGURES

A.1	Examples of generated distributions	192
A.2	Travel guidance component: the web interface	193
A.3	Travel guidance component: the mobile application	194
A.4	Travel guidance component: intermodal connections	195

List of Tables

2.1	Examples for Pareto dominance	23
8.1	Examples for the reliability assessments of intermodal connections	143
9.1	Evaluation: processing events (time-expanded graph model)	156
9.2	Evaluation: processing events (time-dependent graph model with a de- pendency graph)	156
9.3	Evaluation: processing events (one day only)	157
9.4	Evaluation: updating events	157
9.5	Evaluation: the outcomes of the approaches	166
9.6	Evaluation: the optimality of the outcomes	167
9.7	Evaluation: the price of the on-time arrival guarantee	169
9.8	Evaluation: the number of transfers and the travel times	171
9.9	Evaluation: the number of transfers (optimal complete connections) . .	171
9.10	Evaluation: the run times of the approaches	173
9.11	Evaluation: different approaches of discovering relevant events	175
9.12	Evaluation: late-night connections	180

Publications

- [KSW17] Mohammad Hossein **Keyhani**, Mathias Schnee, and Karsten Weihe. Arrive in Time by Train with High Probability. *Transportation Science*. Accepted manuscript. 2017.
- [FGKS16] Sebastian Fahnenschreiber, Felix Gündling, Mohammad Hossein **Keyhani**, and Mathias Schnee. A Multi-modal Routing Approach Combining Dynamic Ride-sharing and Public Transport. *Transportation Research Procedia*. 2016.
- [KSW14] Mohammad Hossein **Keyhani**, Mathias Schnee, and Karsten Weihe. Path Finding Strategies in Stochastic Networks. Technical report, Technische Universität Darmstadt. 2014.
- [GKSW14] Felix Gündling, Mohammad Hossein **Keyhani**, Mathias Schnee, and Karsten Weihe. Fully Realistic Multi-Criteria Multi-Modal Routing. Technical report, Technische Universität Darmstadt. 2014.
- [KSWZ12] Mohammad Hossein **Keyhani**, Mathias Schnee, Karsten Weihe, and Hans-Peter Zorn. Reliability and Delay Distributions of Train Connections. 12th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, ATMOS. 2012.

Bibliography

- [AJ14] Yossiri Adulyasak and Patrick Jaillet. Models and Algorithms for Stochastic and Robust Vehicle Routing with Deadlines. *Transportation Science*, 2014. Cited on page 70.
- [Alg16] Algorithm Engineering. Website of the institute of Algorithm Engineering, Technische Universität Darmstadt, Germany. <http://www.algo.informatik.tu-darmstadt.de/>, 2016. Accessed: 2016-12-10. Cited on page 153.
- [AMZ09] Ravindra K. Ahuja, Rolf H. Möhring, and Christos D. Zaroliagis, editors. *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*, volume 5868 of *LNCS*. Springer, 2009. Cited on page 65.
- [Arn14] Cornelius Arndt. Auxiliary Routing Services for an Intermodal Travel Information System, 2014. Student Research Project, Technische Universität Darmstadt. Cited on pages 11, 25, 132, and 133.
- [Bah16a] Deutsche Bahn, 2016. Daten & Fakten 2015. Cited on page 1.
- [Bah16b] Deutsche Bahn. Fahrgastrechte. http://www.bahn.de/p/view/service/auskunft/fahrgastrechte/nationale_regelungen.shtml, 2016. Accessed: 2016-12-28. Cited on pages 144 and 177.
- [Bah16c] Deutsche Bahn. German railway company. <http://www.bahn.de/>, 2016. Accessed: 2016-11-16. Cited on pages 8, 125, and 153.
- [BGMHO11] 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 Modeling, Optimization, and Systems, ATMOS*, pages 100–111, 2011. Cited on pages 2, 32, 33, 59, and 183.

- [BMP⁺13] Kateřina Böhmová, Matúš Mihalák, Tobias Pröger, Rastislav Šrámek, and Peter Widmayer. Robust Routing in Urban Public Transportation: How to Find Reliable Journeys Based on Past Observations. In *13th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, ATMOS*, volume 33 of *OASICS*, pages 27–41, 2013. Cited on page 69.
- [CBM12] Felix Caicedo, Carola Blazquez, and Pablo Miranda. Prediction of Parking Space Availability in Real Time. *Expert Systems with Applications*, 39(8):7281 – 7290, 2012. Cited on page 132.
- [CK94] Malachy Carey and Andrzej Kwieciński. Stochastic Approximation to the Effects of Headways on Knock-On Delays of Trains. *Transportation Research Part B: Methodological*, 28(4):251–267, 1994. Cited on page 33.
- [CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001. Cited on pages 53, 54, 55, 93, and 95.
- [DAD16] DADINA. Website of Darmstadt-Dieburger Nahverkehrsorganisation. <http://www.dadina.de/>, 2016. Accessed: 2016-12-10. Cited on page 154.
- [Dar09] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, Cambridge, New York, 2009. Cited on page 43.
- [DMS08] Yann Disser, Matthias Müller-Hannemann, and Mathias Schnee. Multi-criteria Shortest Paths in Time-Dependent Train Networks. In *Experimental Algorithms, 7th International Workshop, WEA 2008, Provincetown, MA, USA, May 30-June 1, 2008, Proceedings*, pages 347–361, 2008. Cited on pages 5, 15, 18, and 21.
- [DPSW13] Julian Dibbelt, Thomas Pajor, Ben Strasser, and Dorothea Wagner. Intriguingly Simple and Fast Transit Routing. In *12th International SEA*, pages 43–54, 2013. Cited on page 69.
- [DSW14] Julian Dibbelt, Ben Strasser, and Dorothea Wagner. Delay-Robust Journeys in Timetable Networks with Minimum Expected Arrival Time. In *14th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, ATMOS 2014, September 11, 2014, Wroclaw, Poland*, pages 1–14, 2014. Cited on page 69.
- [Faw06] Tom Fawcett. An Introduction to ROC Analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006. Cited on pages 159, 160, and 161.

- [FGKS16] Sebastian Fahnenschreiber, Felix Gündling, Mohammad H. Keyhani, and Mathias Schnee. A Multi-modal Routing Approach Combining Dynamic Ride-sharing and Public Transport. *Transportation Research Procedia*, 13:176 – 183, 2016. Towards future innovative transport: visions, trends and methods 43rd European Transport Conference Selected Proceedings. Cited on pages 25 and 211.
- [FMHS08] Lennart Frede, Matthias Müller-Hannemann, and Mathias Schnee. Efficient On-Trip Timetable Information in the Presence of Delays. In *8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, ATMOS*, 2008. Cited on pages 58 and 114.
- [FN06] Y. Fan and Y. Nie. Optimal Routing for Maximizing the Travel Time Reliability. *Networks and Spatial Economics*, 6(3-4):333–344, sep 2006. Cited on page 70.
- [FR98] Liping Fu and L. R. Rilett. Expected Shortest Paths in Dynamic and Stochastic Traffic Networks. *Transportation Research Part B: Methodological*, 32(7):499–516, 1998. Cited on page 70.
- [FS09] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, Cambridge, New York, 2009. Autre tirage: 2010. Cited on page 60.
- [GHMH⁺13] Marc Goerigk, Sacha Heße, Matthias Müller-Hannemann, Marie Schmidt, and Anita Schöbel. Recoverable Robust Timetable Information. In *13th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, ATMOS*, volume 33 of *OASiCs*, pages 1–14, 2013. Cited on page 70.
- [GKMH⁺11] Marc Goerigk, Martin Knoth, Matthias Müller-Hannemann, Marie Schmidt, and Anita Schöbel. The Price of Robustness in Timetable Information. In *11th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, ATMOS*, volume 20 of *OASiCs*, pages 76–87, 2011. Cited on page 70.
- [GKSW14] Felix Gündling, Mohammad Hossein Keyhani, Mathias Schnee, and Karsten Weihe. Fully Realistic Multi-Criteria Multi-Modal Routing. Technical report, Technische Universität Darmstadt, Dezember 2014. Cited on pages 25 and 211.

- [Glö15] Peter Glöckner. Parallelization for the Search for Reliable Connections. Master's thesis, Technische Universität Darmstadt, Darmstadt, 2015. Cited on pages vii, 56, and 108.
- [Gro15] Julian Gross. Estimation of Travel Time Distributions for Trains Based on a Real Delay History, 2015. Bachelor's thesis, Technische Universität Darmstadt. Cited on pages vii and 37.
- [GSS⁺14] Marc Goerigk, Marie Schmidt, Anita Schöbel, Martin Knoth, and Matthias Müller-Hannemann. The Price of Strict and Light Robustness in Timetable Information. *Transportation Science*, 48(2):225–242, 2014. Cited on page 70.
- [Gün14] Felix Gündling. An Intermodal Travel Information System, 2014. Student Research Project, Technische Universität Darmstadt. Cited on pages 11, 25, and 132.
- [Gün15] Felix Gündling. Modelling and Speedup Techniques for an Intermodal Travel Information System. Master's thesis, Technische Universität Darmstadt, Darmstadt, 2015. Cited on pages 5, 11, 15, 18, 22, 24, 25, 27, 28, and 132.
- [HaC16] HaCon. Official website. <http://www.hacon.de/>, 2016. Accessed: 2016-12-02. Cited on page 1.
- [Hal86] Randolph W. Hall. The Fastest Path through a Network with Random Time-Dependent Travel Times. *Transportation Science*, 20(3):182–188, August 1986. Cited on page 69.
- [Int16] IntercityHotel. Official website. <https://www.intercityhotel.com/hotels/alle-hotels/deutschland>, 2016. Accessed: 2017-01-20. Cited on page 178.
- [Key09] Mohammad Hossein Keyhani. Delay Management in Timetable Information Systems, 2009. Bachelor's thesis, Technische Universität Darmstadt. Cited on page 58.
- [KSW14] Mohammad Hossein Keyhani, Mathias Schnee, and Karsten Weihe. Path Finding Strategies in Stochastic Networks. Technical report, Technische Universität Darmstadt, Dezember 2014. Cited on pages 71 and 211.
- [KSW17] Mohammad Hossein Keyhani, Mathias Schnee, and Karsten Weihe. Arrive in Time by Train with High Probability. Accepted manuscript, Trans-

- portation Science journal: <http://pubsonline.informs.org/journal/trsc>, 2017. Cited on pages 5, 70, 71, and 211.
- [KSWZ12] Mohammad Hossein Keyhani, Mathias Schnee, Karsten Weihe, and Hans-Peter Zorn. Reliability and Delay Distributions of Train Connections. In *12th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, ATMOS*, volume 25 of *OASICs*, pages 35–46, 2012. Cited on pages 33, 161, 189, 192, and 211.
- [LRR⁺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 Stefan Funke and Matúš Mihalák, editors, *14th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, ATMOS*, volume 42 of *OpenAccess Series in Informatics (OASICs)*, pages 122–137, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. Cited on page 33.
- [LV11] Dennis Luxen and Christian Vetter. Real-time Routing with OpenStreetMap Data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS '11*, pages 513–516, New York, NY, USA, 2011. ACM. Cited on pages 26 and 147.
- [MDnP10] Pavankumar Murali, Maged Dessouky, Fernando Ordóñez, and Kurt Palmer. A Delay Estimation Technique for Single and Double-Track Railroads. *Transportation Research Part E: Logistics and Transportation Review*, 46(4):483 – 495, 2010. Selected papers from the Second National Urban Freight Conference, Long Beach, California, December 2007. Cited on page 33.
- [Meh07] Kai Mehringskötter. Effiziente Fahrplanauskunft unter Optimierung der Umstiegssicherheit, 2007. Bachelor’s thesis, Technische Universität Darmstadt. Cited on page 33.
- [MHS04] Matthias Müller-Hannemann and Mathias Schnee. Finding All Attractive Train Connections by Multi-Criteria Pareto Search. In *4th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, pages 246–263, 2004. Cited on pages 5, 15, 17, and 21.
- [MHS09] Matthias Müller-Hannemann and Mathias Schnee. Efficient Timetable Information in the Presence of Delays. In *Robust and Online Large-Scale*

- Optimization*, pages 249–272. Springer, 2009. Cited on pages 5, 33, 63, and 70.
- [MM07] Ludolf E. Meester and Sander Muns. Stochastic Delay Propagation in Railway Networks and Phase-Type Distributions. *Transportation Research Part B: Methodological*, 41(2):218 – 230, 2007. Advanced Modelling of Train Operations in Stations and Networks. Cited on page 33.
- [Mül16] Maximilian Müller. Tool for Analyzing the Current Situation in a Train Network, 2016. Bachelor’s thesis, Technische Universität Darmstadt. Cited on page vii.
- [NBK06] Evdokia Nikolova, Matthew Brand, and David R. Karger. Optimal Route Planning under Uncertainty. In *ICAPS*, pages 131–141, 2006. Cited on page 70.
- [NIS17a] NIST/SEMATECH. E-Handbook of Statistical Methods. <http://www.itl.nist.gov/div898/handbook/eda/section3/eda3663.htm>, 2017. Accessed: 2017-01-16. Cited on page 42.
- [NIS17b] NIST/SEMATECH. E-Handbook of Statistical Methods. <http://www.itl.nist.gov/div898/handbook/apr/section1/apr1a.htm>, 2017. Accessed: 2017-01-16. Cited on page 46.
- [NPA04] L.R. Nielsen, D. Pretolani, and K.A. Andersen. *K* Shortest Paths in Stochastic Time-Dependent Networks. Technical Report WP-L-2004-05, Department of Accounting, Finance and Logistics, Aarhus School of Business, 2004. Cited on page 70.
- [NPA06] L.R. Nielsen, D. Pretolani, and K.A. Andersen. Bicriterion A Priori Route Choice in Stochastic Time-Dependent Networks. Technical Report WP-L-2006-10, Department of Business Studies, Aarhus School of Business, 2006. Cited on page 70.
- [Pla16] Marco Plaue. Probabilistic Delay Prognoses and Reliable Train Connections in Time-Dependent Graphs, 2016. Bachelor’s thesis, Technische Universität Darmstadt. Cited on pages vii, 37, 127, 128, and 130.
- [Pre98] Daniele Pretolani. A Directed Hypergraph Model for Random Time Dependent Shortest Paths. *European Journal of Operational Research*, 123:2000, 1998. Cited on page 70.

- [PSG13] Yiyong Pan, Lu Sun, and Minli Ge. Finding Reliable Shortest Path in Stochastic Time-dependent Network. *Procedia - Social and Behavioral Sciences*, 96(0):451 – 460, 2013. (CICTP2013). Cited on page 70.
- [Raf15] Tobias Raffel. A Framework for Black-Box Testing and its Application to Timetable Information Systems, 2015. Bachelor’s thesis, Technische Universität Darmstadt. Cited on page vii.
- [RBW16] Tarun Rambha, Stephen D. Boyles, and S. Travis Waller. Adaptive Transit Routing in Stochastic Time-Dependent Networks. *Transportation Science*, 50(3):1043–1059, 2016. Cited on pages 69 and 70.
- [SBB11] S. Samaranayake, S. Blandin, and A. Bayen. A Tractable Class of Algorithms for Reliable Routing in Stochastic Networks. *Procedia - Social and Behavioral Sciences*, 17(0):341 – 363, 2011. 19th ISTTT. Cited on page 70.
- [SBB12] Samitha Samaranayake, Sebastien Blandin, and Alexandre M. Bayen. Speedup Techniques for the Stochastic On-Time Arrival Problem. In *12th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, ATMOS*, volume 25 of *OASICs*, pages 83–96, 2012. Cited on page 70.
- [Sch09] Mathias Schnee. *Fully Realistic Multi-Criteria Timetable Information Systems*. PhD thesis, Technische Universität Darmstadt, 2009. Cited on pages 1, 5, 15, 18, 22, 24, 33, 58, 107, and 114.
- [Sch16] Kai Schwierczek. Computing Joint Delay Distributions for Trains in Railway Networks, 2016. Bachelor’s thesis, Technische Universität Darmstadt. Cited on pages vii, 59, and 60.
- [SWW00] Frank Schulz, Dorothea Wagner, and Karsten Weihe. Dijkstra’s Algorithm On-line: An Empirical Case Study from Public Railroad Transport. *J. Exp. Algorithmics*, 5, December 2000. Cited on page 5.

Wissenschaftlicher Werdegang des Verfassers¹

2012 – 2017	Wissenschaftlicher Mitarbeiter, Doktorand Fachgebiet Algorithmik Fachbereich Informatik Technische Universität Darmstadt
2012	Abschluss als Master of Science im Studiengang Informatik Abschlussarbeit: <i>Super-Resolution für einzelne medizinische 3D-Bilddatensätze</i>
2009 – 2012	Masterstudium der Informatik Technische Universität Darmstadt
2009	Abschluss als Bachelor of Science im Studiengang Informatik Abschlussarbeit: <i>Verspätungsbehandlung in Bahnfahrplanauskunftssystemen</i>
2005 – 2009	Bachelorstudium der Informatik Technische Universität Darmstadt

¹Gemäß §20 Abs. 3 der Promotionsordnung der TU Darmstadt

Erklärung²

Hiermit erkläre ich, die vorgelegte Arbeit zur Erlangung des akademischen Grades “Dr. rer. nat.” selbstständig und ausschließlich unter Verwendung der angegebenen Quellen und Hilfsmittel erstellt zu haben. Ich habe alle Stellen, die dem Wortlaut oder Sinne nach anderen Werken entnommen sind, durch Angabe der Quellen als Entlehnungen kenntlich gemacht. Ich habe bisher noch keinen Promotionsversuch unternommen.

Darmstadt, 2. Februar 2017

Mohammad Hossein Keyhani

²Gemäß §9 Abs. 1 der Promotionsordnung der TU Darmstadt

