Timetable Information: Models and Algorithms*

Matthias Müller-Hannemann[1], Frank Schulz[2], Dorothea Wagner[2], and Christos Zaroliagis[3]

[1] Darmstadt University of Technology, Department of Computer Science, Algorithmics Group, Hochschulstr. 10, 64289 Darmstadt, Germany
muellerh@algo.informatik.tu-darmstadt.de
[2] University of Karlsruhe, Department of Computer Science, P.O. Box 6980, 76128 Karlsruhe, Germany
{fschulz,dwagner}@ira.uka.de
[3] Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece, and Department of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece
zaro@ceid.upatras.gr

Abstract. We give an overview of models and efficient algorithms for optimally solving timetable information problems like "given a departure and an arrival station as well as a departure time, which is the connection that arrives as early as possible at the arrival station?" Two main approaches that transform the problems into shortest path problems are reviewed, including issues like the modeling of realistic details (e.g., train transfers) and further optimization criteria (e.g., the number of transfers). An important topic is also multi-criteria optimization, where in general all attractive connections with respect to several criteria shall be determined. Finally, we discuss the performance of the described algorithms, which is crucial for their application in a real system.

1 Introduction

The first electronic timetable information systems were established in the late eighties of the last century. Current systems are for example HAFAS [13], which is used by many European railway companies, or EFA [8], which is mainly used for local traffic limited to smaller regions in Europe. Empirically, the resulting connections are satisfying in the majority of cases. There are cases, however, for which the suggested itineraries are clearly not optimal (given some optimization criterion). The main reason for such non-optimal connections is that the algorithms behind the systems employ heuristic methods to reduce the search space (in order to achieve an acceptable response time) that do not always guarantee optimal solutions. Such heuristic approaches often work in two phases as described below.

In the last few years the question arose whether models and algorithms for optimally solving timetable information problems are feasible. In this work we

* Partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

F. Geraets et al. (Eds.): Railway Optimization 2004, LNCS 4359, pp. 67–90, 2007.

want to give an overview of such approaches, which solve timetable information queries by finding a shortest path in an appropriately defined graph. Hence, the problems are directly transformed into shortest path problems.

In the remainder of the introduction we give a brief overview of heuristic two-phase approaches and direct shortest-path approaches. In Section 2 the timetable information problems are formally specified. Two main approaches for modeling timetable information directly as shortest paths are described in detail in Section 3, where first a simplified problem specification is considered. Later on, in Section 4, extensions of the approaches that cover realistic details are outlined. Multi-criteria optimization is discussed in Section 5, and studies investigating the performance of the algorithms described in this paper are summarized in Section 6. We conclude the survey with some final remarks in Section 7.

1.1 Two-Phase Approaches

We want to mention two predecessors of "real" timetable information systems. Around the year 1988, the Dutch and German train companies started to use electronic timetable information systems. Heuristics that usually yield good solutions, but cannot always guarantee an optimal solution, are used to keep the search spaces small enough. The two systems we describe work both in two phases, where the first phase heuristically restricts the search space.

TRAINS

Tulp and Siklòssy [37] describe the TRAINS system, which was used by the Dutch railways (NS) at that time as a prototype: It is based on a graph where nodes represent cities. They distinguish two levels of the network, a "static" level which consists of arcs between nodes representing distances, and a "dynamic" level where the arcs include information about the departure and arrival times of trains. The algorithm uses the static level to cut out the "interesting" part of the network, without considering any information about time. Note that this cutting is heuristic in the sense that optimal connections may be lost by that step, which is not permitted in the models investigated later in this paper. Then, a train connection is calculated by a modification of Dijkstra's algorithm [7] trying to incorporate time for train changes at stations. Once a connection to the destination station is found, a backward search tries to improve the result (e.g., to find a connection that departs later and has the same arrival time).

ARIADNE

Baumann and Schmidt [2] outline an algorithm called ARIADNE, which can be regarded as ancestor of HAFAS [13], the timetable information system that is nowadays used by the German railway company Deutsche Bahn AG and many other railway companies worldwide. As in TRAINS, the algorithm considers two different networks: a static graph representing the topographic railway network, and a dynamic network including time, traffic days, train classes etc. The ARIADNE algorithm works in two phases: The first phase ("Wegesuche") searches

feasible paths in the static network by a bidirectional version of Dijkstra's algorithm and outputs a subgraph of the network to be considered in the second phase. Note that again—as in the TRAINS algorithm—optimal solutions may be lost by this step. The second phase ("Zeitsuche") computes on the dynamic, time-dependent version of the network, limited by the subgraph computed in the first phase, several feasible train connections. These are rated according to measures like travel time, quality of trains, direct connection, etc.

1.2 Direct Shortest Path Approaches

Two main approaches have been proposed for modeling timetable information as shortest path problem: the *time-expanded* [19,20,21,22,26,30,28,32,33,34], and the *time-dependent* approach [4,5,16,23,24,25,30,28,29,32]. The common characteristic of both approaches is that a query is answered by applying some shortest path algorithm to a suitably constructed graph.

The Time-Expanded Approach

A time-expanded graph is constructed in which every node corresponds to a specific time event (departure or arrival) at a station and edges between nodes represent either elementary connections between the two events (i.e., served by a train that does not stop in-between), or waiting within a station. Depending on the optimization criterion, the construction assigns specific fixed lengths to the edges. This naturally results in the construction of a very large (but usually sparse) graph. The simplified version of the earliest arrival problem—where details like transfer rules and traffic days are neglected—has been extensively studied:

In [33], Schulz, Wagner and Weihe explicitly use the time-expanded approach to model a simplified version of the earliest arrival problem as a shortest path problem in a static graph, and solve the problem optimally. An extensive experimental study has been conducted and—at least in the simplified scenario—it could be demonstrated that the running time of the time-expanded approach on state-of-the-art computers is acceptable. To achieve this result, several speed-up techniques, which guarantee optimal solutions, were applied to Dijkstra's algorithm for computing the shortest path. More details on the speed-up techniques are provided in Section 6.

An extension of the time-expanded approach incorporating train transfers and an extensive experimental study focused on multi-criteria problems is presented by Müller-Hannemann and Weihe in [22]. The results of this study are quite promising: in practice (among other data also the time-expanded graph was considered) the number of Pareto-optimal paths is often very small, and labeling approaches are feasible. In [21], Müller-Hannemann, Schnee and Weihe focus on more realistic and complex real-world scenarios for timetable information, in particular with respect to space limitations. Further extensions towards realistic models and also further optimization criteria as well as bicriteria problems are presented by Pyrga, Schulz, Wagner and Zaroliagis in [30,28] (see

also [32]), where the authors also conducted an experimental comparison with the time-dependent approach (see below). Multi-criteria optimization in the time-expanded graph by a labeling approach is extensively investigated by Müller-Hannemann and Schnee [20]; the notion of Pareto-optimal connections is relaxed (cf. 5.2). Möhring suggests the time-expanded model as a graph-theoretic concept for timetable information in [19]. He further discusses algorithms for solving multi-criteria problems, and focuses on a distributed approach for timetable information, which is also the topic of the recent projects DELFI [6] and EU-Spirit [11]: the railway network is considered as consisting of several (overlapping) subnetworks (e.g., each subnetwork is operated by a different company or institution), and a global solution is constructed from several subqueries to the conventional timetable information systems operated on the respective subnetworks. In a sense such new systems operate like meta search engines for the web.

The Time-Dependent Approach

The idea is to avoid the maintenance of a node per event. Instead, the time-dependent graph is used in which every node represents a station, and two nodes are connected by an edge if the corresponding stations are connected by an elementary connection. The lengths on the edges are assigned "on-the-fly": the length of an edge depends on the time in which the particular edge will be used by the shortest path algorithm to answer the query. Dynamic programming approaches for a time-dependent shortest-path problem have first been studied by Cooke and Halsey [5]. Later, Kostreva and Wiecek [16] generalized this approach towards multiple criteria. However, no performance guarantees are given for these dynamic programming approaches. Orda and Rom [24,25] thoroughly investigated the complexity of time-dependent shortest path problems and give efficient algorithms for special cases. Brodal and Jacob [3,4] argued that in the simplified case of the earliest arrival problem, Dijkstra's algorithm considers many redundant edges in the time-expanded approach. They suggest to use a time-dependent network instead and proved by a detailed theoretical analysis of operation counts in both approaches that a variant of a time-dependent shortest-path algorithm introduced by Orda and Rom is more efficient than the time-expanded approach. Pyrga, Schulz, Wagner and Zaroliagis extended the time-dependent model to cope with realistic problem specifications [29]. A subsequent study [30,28,32] compares these models experimentally with the time-expanded models, where also bicriteria problems are considered.

The work of Nachtigal [23] can also be classified as a time-dependent approach to timetable information. The problem specification he uses is different to most other approaches: given a source station, for all other stations arrival functions depending on the departure time shall be computed. Hence, the departure time is not part of the query, and solutions are computed for all possible departure times.

2 Problem Specification

2.1 Data

A *timetable* consists of data concerning: stations (or bus stops, ports, etc), trains (or busses, ferries, etc), connecting stations, departure and arrival times of trains at stations, and traffic days. More formally, we are given a set of *trains* $\mathcal{Z}$, a set of stations $\mathcal{B}$, and a set of *elementary connections* $\mathcal{C}$ whose elements c are 5-tuples of the form $c = (Z, S_1, S_2, t_d, t_a)$. Such a tuple (elementary connection) is interpreted as train Z leaves station S_1 at time t_d, and the next stop of train Z is station S_2 at time t_a. If x denotes a tuple's field, then the notation $x(c)$ specifies the value of x in the elementary connection c.

The *departure* and *arrival times* $t_d(c)$ and $t_a(c)$ of an elementary connection $c \in \mathcal{C}$ within a day are integers in the interval $[0, 1439]$ representing time in minutes after midnight. The *length* of an elementary connection c, denoted by *length*(c), is the time that passes between the departure and the arrival of c.

A timetable is valid for a number of N *traffic days*, and every train is assigned a bit-field of N bits determining on which traffic day the train operates (for overnight trains the departure of the first elementary connection counts).

At a station $S \in \mathcal{B}$ it is possible to transfer from one train to another only if the time between the arrival and the departure at that station S is larger than or equal to a given, station-specific, *minimum transfer time*, denoted by *transfer*(S). There may also be more complicated transfer rules, for example the transfer time can be smaller for trains that depart from the same platform. The most general notion is to specify a station-specific minimum transfer time, and exceptions in the form of a set of feasible and a set of forbidden transfer trains for each arrival of a train at a station.

Between stations that are located close to each other it is possible to walk by foot. Such data is available through so-called *foot-edges* between stations. Each foot-edge is associated with a natural number representing the time in minutes needed for the walk. Formally, we treat a foot-edge like an elementary connection c, where the train Z and the departure and arrival times t_d and t_a are invalid, and *length*(c) is the associated walking time. Foot-edges are independent of traffic days.

2.2 Connections

Let $P = (c_1, \ldots, c_k)$ be a sequence of elementary connections (and foot-edges) together with departure times $dep_i(P)$ and arrival times $arr_i(P)$ for each elementary connection c_i, $1 \leq i \leq k$. We assume that the times $dep_i(P)$ and $arr_i(P)$ include data regarding also the departure/arrival day by counting time in minutes from the first day of the timetable. A time value t is of the form $t = a \cdot 1440 + b$, where $a \in [0, 364]$ and $b \in [0, 1439]$. Hence, the actual time within a day is $t \pmod{1440}$ and the actual day is $\lfloor t/1440 \rfloor$.

Such a sequence P is called a *consistent connection* from station $A = S_1(c_1)$ to station $B = S_2(c_k)$ if it fulfills some consistency conditions: the departure station

of c_{i+1} is the arrival station of c_i, and the time values $dep_i(P)$ and $arr_i(P)$ correspond to the time values t_d and t_a, resp., of the elementary connections (modulo 1440) and respect the transfer times at stations. More formally, P is a *consistent connection* if the following conditions are satisfied:

$$\begin{aligned} c_i \quad & \text{is valid on day } \lfloor dep_i(P)/1440 \rfloor, \\ S_2(c_i) &= S_1(c_{i+1}), \\ dep_i(P) &\equiv t_d(c_i) \pmod{1440}, \\ arr_i(P) &= dep_i(P) + length(c_i), \\ dep_{i+1}(P) - arr_i(P) &\geq \begin{cases} 0 & \text{if } Z(c_{i+1}) = Z(c_i) \text{ or} \\ & c_i \text{ is a foot-edge, and} \\ transfer(S_2(c_i)) & \text{otherwise.} \end{cases} \end{aligned}$$

2.3 Criteria and Queries

For the timetable information problems we are additionally given a large, on-line sequence of *queries*. A query generally defines a set of valid connections, and an optimization criterion (or criteria) on that set of connections. The problem is to find the optimal connection (or a set of optimal connections) w.r.t. the specific criterion or criteria.

The Basic Query

The most fundamental query is also referred to as the *earliest arrival problem*. A query (A, B, t_0) consists of a departure station A, an arrival station B, and a departure time t_0. Connections are *valid* if they do not depart before the given departure time t_0, and the optimization criterion is to minimize the difference between the arrival time and the given departure time. Additionally, one may ask among all connections that are solutions to such a query for the connection that departs as late as possible.

Extended Queries

Other important optimization criteria involve the *number of transfers* and the *price* of a connection. In the *minimum number of transfers problem*, the query is to ask, given two stations A and B, for a connection with as few transfers as possible, which doesn't involve a departure or arrival time at all. Similarly, one can ask for a connection with the lowest price.

A query can also contain a sequence of via stations together with the duration of the stays at the respective stations. Further, certain trains or train classes can be excluded from the set of trains, e.g., if one intends to use a ticket that is valid only for local trains the intercity trains should be excluded. Also, instead of specifying the departure time, as in the earliest arrival problem, the aspired arrival time may be given. Alternatively, such time specifications can be given as time intervals.

3 Basic Modeling: The Earliest Arrival Problem

In this section, we review models for solving the first and most fundamental basic query, namely the earliest arrival problem, in both the time-expanded and the time-dependent approach. We consider throughout this section a simplified specification of train connections: We assume that (i) a transfer between trains at a station takes negligible time, i.e., $transfer(S) = 0$ for each station S, (ii) every train is operated daily, i.e., every day is the same in the timetable, and (iii) there are no foot-edges. In the following section we show how the models can be extended to comply with the realistic specification, and also consider the extended types of queries.

3.1 Time-Expanded Model

The time-expanded model [33] is based on the directed *time-expanded graph* which is constructed as follows. There is a node for every time *event* (departure or arrival) at a station, and there are two types of edges. For every elementary connection (Z, S_1, S_2, t_d, t_a) in the timetable, there is a *train-edge* in the graph connecting a *departure node*, belonging to station S_1 and associated with time t_d, with an *arrival node*, belonging to station S_2 and associated with time t_a. In other words, the endpoints of the train-edges induce the set of nodes of the graph. For each station S, all nodes belonging to S are ordered according to their time values. Let $v_1, \ldots, v_k$ be the nodes of S in that order. Then, there is a set of *stay-edges* (v_i, v_{i+1}), $1 \leq i \leq k-1$, and (v_k, v_1) connecting the time events within a station and representing waiting within that station. The length of an edge (u, v) is $t_v - t_u$ (for edges over midnight the length is $1440 + t_v - t_u$, respectively), where t_u and t_v are the time values associated with u and v, respectively. Figure 1 illustrates this definition.

A shortest path in the time-expanded graph from the first departure node s at the departure station A with departure time later than or equal to the given start time t_0 to one of the arrival nodes of the destination station B constitutes a solution to the earliest arrival problem in the time-expanded model. The actual path can be found by Dijkstra's algorithm [7].

3.2 Time-Dependent Model

The time-dependent model [4] is also based on a digraph, called *time-dependent graph*. In this graph there is only one node per station, and there is an edge e from station A to station B if there is an elementary connection from A to B. The set of elementary connections from A to B is denoted by $\mathcal{C}(e)$. The definition is illustrated in Figure 1. The length of an edge $e = (v, w)$ depends on the time at which this particular edge will be used during the algorithm. In other words, if T is a set denoting time, then the length of an edge (v, w) is given by $f_{(v,w)}(t) - t$, where t is the departure time at v, $f_{(v,w)} : T \to T$ is a function such that $f_{(v,w)}(t) = t'$, and $t' \geq t$ is the earliest possible arrival time at w. The time-dependent model is based on the assumption that overtaking of trains on

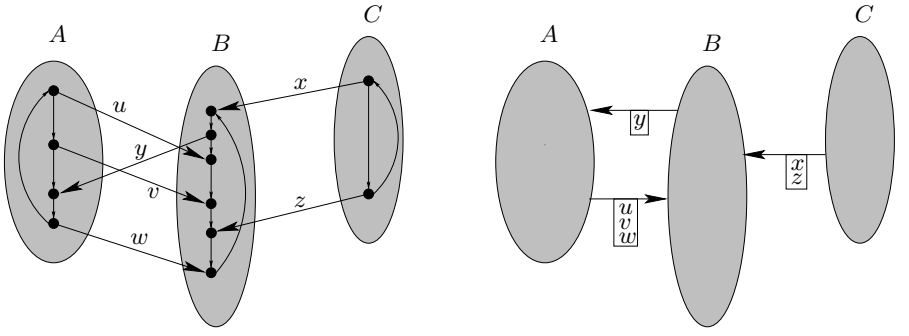

Fig. 1. The time-expanded graph (left) and the time-dependent graph (right) of a timetable with three stations A, B, C. There are three trains that connect A with B (elementary connections u,v,w), one train from C via B to A (x,y) and one train from C to B (z).

an edge is not allowed, i.e., for any two given stations A and B, there are no two trains leaving A and arriving to B such that the train that leaves A second arrives first at B.

A modification of Dijkstra's algorithm [7] can be used to solve the earliest arrival problem in the time-dependent model [4]. Let D denote the departure station and t_0 the earliest departure time. The differences, w.r.t. Dijkstra's algorithm, are: set the distance label $\delta(D)$ of the starting node corresponding to the departure station D to t_0 (and not to 0), and calculate the edge lengths "on-the-fly". The edge lengths (and implicitly the time-dependent function f) are calculated as follows. Since Dijkstra's algorithm is a label-setting shortest-path algorithm, whenever an edge $e = (A, B)$ is considered the distance label $\delta(A)$ of node A is optimal. In the time-dependent model, $\delta(A)$ denotes the earliest arrival time at station A. In other words, we indeed know the earliest arrival time at station A whenever the edge $e = (A, B)$ is considered, and therefore we know at that stage of the algorithm which train has to be taken to reach station B via A as early as possible: the first train that departs later than or equal to the earliest arrival time at A. The particular connection $c \in C(e)$ can be easily found by binary search if the elementary connections $\mathcal{C}(e)$ are maintained in a sorted array (or with more sophisticated techniques in constant time). The edge length of e, $\ell_e(t)$, is then defined to be the time to wait for the departure of c plus *length*(c). Consequently, $f_e(t) = t + \ell_e(t)$. The correctness of the algorithm is based on the fact that f is *non-decreasing* ($t \leq t' \Rightarrow f(t) \leq f(t')$) and has *non-negative delay* ($\forall t, f(t) \geq t$).

3.3 Comparison of Models

In the simplified scenario we are investigating in this section, the graphs that are used in the two approaches are strongly related: Contracting all nodes that belong to the same station in the time-expanded graph and deleting parallel

edges afterwards yields the time-dependent graph. Further, the algorithm used in the time-dependent approach can be viewed as an improved implementation of the simple shortest-path search by Dijkstra's algorithm in the time-expanded approach: If the first edge from some station A to another station B has already been processed by Dijkstra's algorithm in the time-expanded graph, all other edges e'_{AB} from station A to station B do not have to be considered anymore. The reason is that such an edge doesn't provide an improvement since the path through the first edge extended by some stay-edges to the head of the edge e'_{AB} has the same length. In a sense, the time-dependent algorithm implements this observation.

Note, however, that on the one hand the edge lengths have still to be computed in the time-dependent algorithm, which consumes running time as well, so that it is not immediately clear which algorithm is faster. We will discuss this question in Section 6. On the other hand, the similarity of the graphs and the algorithms is disturbed when the realistic specifications are incorporated into the models in the following section.

4 Realistic Modeling

In this section, we explain how the approaches introduced so far for the simplified earliest arrival problem can be extended towards realistic problem specifications and other optimization criteria.

4.1 Transfers Rules

We summarize first how transfer times at stations can be incorporated in the time-expanded and the time-dependent models, and after that discuss the case of extended transfer rules.

Time-Expanded Model

To incorporate transfer times in the time-expanded model the *realistic time-expanded graph* is constructed as follows (cf. [22,30,28]). Based on the time-expanded graph, for each station, a copy of all departure and arrival nodes in the station is maintained which we call *transfer-nodes*; see Figure 2. The stay-edges are now introduced between the transfer-nodes. For every arrival node there are two additional outgoing edges: one edge to the departure of the same train, and a second edge to the transfer-node with time value greater than or equal to the sum of the time of the arrival node and the minimum time needed to change trains at the given station. If the earliest arrival problem shall be solved, the edge lengths are defined as in the definition of the original model (see Section 3.1).

Time-Dependent Model

The original time-dependent model is extended in [29] (cf. also [4]) using information on the routes that trains may follow. Hence, we assume that we are given a set

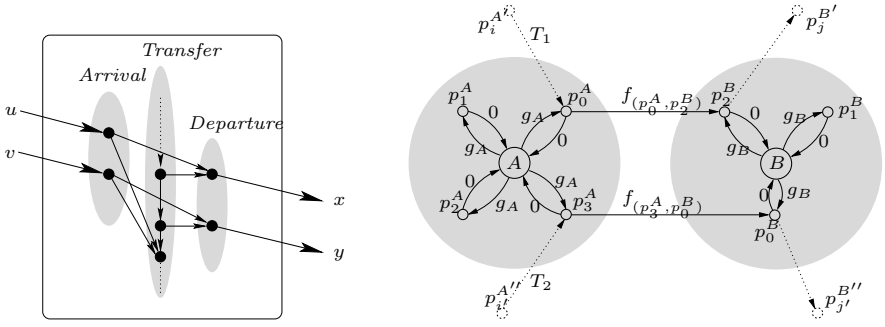

Fig. 2. Modeling transfer times in the time-expanded approach using the realistic time-expanded graph (left) and in the time-dependent approach using the train-route graph (right)

of train routes and their respective time schedules. In the following, we describe the construction of the *train-route graph*. We say that stations $S_0, S_1, \ldots, S_{k-1}$, $k > 0$, form a *train route* if there is some train starting its journey from S_0 and visiting consecutively $S_1, \ldots, S_{k-1}$ in turn. If there are more than one trains following the same schedule (with respect to the order in which they visit the above nodes), then we say that they all belong to the same train route.

The node-set of the train-route graph consists of the station-nodes $\mathcal{S}$ representing the stations, and for each station S of one additional route-node per route that passes through the station S, denoted by p_i^S, where i is an index of the specific route passing through station S. There are three types of edges: (i) edges from each station-node to the route-nodes belonging to the same station model boarding a train belonging to the specific route; (ii) edges from each route-node to the station-node model getting off a train at that station; (iii) for each train route $S_0, \ldots S_{k-q}$ edges that connect the corresponding route-nodes model the actual train trips.

To solve the earliest arrival problem with transfer times, edge lengths are defined as follows. The edges modeling boarding a train at a station S are assigned the transfer time $g_S = \mathit{transfer}(S)$, edges modeling getting off a train are assigned zero length, and the edges representing the train routes have time-dependent lengths as in the basic modeling described in Section 3.2. Given the query to solve, all internal edges are assigned zero length, and the modified version of Dijkstra's algorithm (cf. Section 3.2) is applied.

Extended Transfer Rules

Additionally to the transfer times at stations, exceptions which explicitly allow specific transfers can be modeled in the realistic time-expanded graph as additional edges connecting the arrival with the departure node of the corresponding elementary connections. Concerning the time-dependent approach, in [29] a graph similar to the train-route graph is constructed that allows to model also variable transfer times.

4.2 Foot-Edges

In the time-dependent approach a foot-edge from station A to station B can be directly modeled as an edge in the train-route graph between the two nodes representing the stations A and B. For the earliest arrival problem, such an edge is assigned a constant edge length: the walking time.

The straight-forward modeling of foot-edges in the time-expanded case is of course done by time expansion. For each transfer node of A in the (realistic) time-expanded graph an additional edge is maintained to the first possible transfer node at B. Another solution for the time-expanded approach is to apply the time-dependent idea and compute the additional edges during the algorithm (the node at B has to be calculated depending on the arrival time at B) instead of explicitly constructing them.

4.3 Traffic Days

Edges representing elementary connections of trains that are not valid can be simply ignored during Dijkstra's algorithm in the time-expanded approach, and the test whether an elementary connection is valid or not can be done by a lookup in the traffic day bit-mask of the corresponding train, if the day of departure is known. However, the algorithm has to be slightly modified because it may happen that an optimal connection stays more than a day at a station, and such connections would not be found otherwise. See [30] for details concerning the modification of the algorithm. In the time-dependent approach, the traffic days have to be considered in the calculation of the time-dependent edge lengths.

Problems with traffic days occur when speed-up techniques for Dijkstra's algorithm are applied that require preprocessing (cf. Section 6), because then every day is different and the preprocessing basically has to be done separately for each day.

4.4 The Minimum Number of Transfers Problem

The realistic time-expanded graph as well as the train-route graph (cf. Section 4.1) can be used to solve a minimum number of transfers query with a similar method (cf. [20,22,30,28]): edges that model transfers are assigned a length of one, and all the other edges are assigned length zero. In the time-expanded case all incoming edges of transfer nodes have length one, whereas in the time-dependent case the edges that represent getting off a train, except those belonging to the departure station, are assigned length one, and all other edges have length zero. Note that the edge lengths in the time-dependent train-route graph are all static here.

A shortest path in one of the graphs from a node belonging to (resp. representing) the departure station to a node belonging to (resp. representing) the arrival station provides a solution to the minimum number of transfers problem.

4.5 Extended Queries

Latest Departure

Determining a connection optimized for the latest departure combined with the earliest arrival can be done in the time-expanded case by introducing the latest departure as second criterion and determining the lexicographically first solution. In the time-dependent model the standard approach is to carry out a backward search from the destination station to the arrival station once the earliest arrival at the destination station is known.

Time Intervals

In pre-trip planning one often seeks the fastest connection which starts within a certain departure interval $[t_0, t_1]$ (or arrives within a certain arrival time interval). This variant can also be solved by Dijkstra's algorithm. With a single starting point, Dijkstra's algorithm starts by labeling the corresponding event node with distance 0 and puts it into the priority queue. With a time interval the only difference is that one initially inserts a label for each departure event between t_0 and t_1 into the priority queue and marks all corresponding nodes with a distance label of 0.

Excluding Trains

The exclusion of specific trains or of train classes, the exclusion or required inclusion of train attributes with respect to a given query can be handled like traffic days: We simply mark train edges as *invisible* for the search if they do not meet all requirements of the given query.

Via Stations

A query may contain one (or more) so called *vias*, i.e., stations the connection has to visit and where at least a specified amount of time can be spent. Assume that we have a query $(A = S_0, B = S_k, t_0)$ with via stations S_1, S_2, S_{k-1} and corresponding stay durations $d_1, d_2, \ldots d_k$. To answer such a query, one can simply split the query into basic queries without vias. More precisely, we first answer the query (A, S_1, t_0) and may find out that the earliest arrival time at S_1 is t_1. Then we answer the query $(S_1, S_2, t_1 + d_1)$. If t_i denotes the earliest arrival time at S_i (provided that we have visited $S_1, S_2, \ldots, S_{i-1}$ before), we continue with basic queries of the form $(S_i, S_{i+1}, t_i + d_i)$ for $i = 1, \ldots, k-1$. Finally, we concatenate the connections found in each of the basic queries to get the connection which solves the earliest arrival problem.

Cheapest Connections

Many customers are interested in finding a cheapest connection from station A to B within a certain time interval. Unfortunately, given the complicated fare regulations in most countries, this goal seems to be intractable. Recall that a

shortest path problem on a given digraph usually assumes that the length of a path can be calculated as the sum over the edge lengths which constitute the path. Given nonnegative lengths, the separability of the objective function then suffices to apply Dijkstra's algorithm.

Even in the standard tariff, the fare of a connection is usually not additive based on its elementary connections. The situation becomes substantially worse if one would also like to consider the many exceptions and special offers which exist and frequently change. Hence, there is no hope to solve the cheapest connection problem exactly and simultaneously efficiently.

However, what one can do is to use fare estimations based on a simpler model. Müller-Hannemann and Weihe [22] and Müller-Hannemann and Schnee [20] use a simplified fare model which assumes that the basic fare is proportional to the distance traveled. Depending on the train class an extra supplementary fare is charged. For the fastest trains like German ICE and French TGV, this supplement is assumed to be proportional to the speed of the train, whereas certain trains like Eurocity and Intercity trains have a constant surcharge which has to be paid at most once. With such a simplified fare model we can again use Dijkstra's algorithm if we store in our distance labels also flags indicating which train classes have been used.

Müller-Hannemann and Schnee [20] use these fare estimates to find low cost connections in a framework which does not only look for a single best connection but for several attractive connections.

5 Multi-criteria Optimization

In the previous sections we focused on single-criterion optimization, and in particular on finding fastest connections. In practice, however, one wishes to find optimal connections under several criteria. For instance, a customer may want to ask for a connection with a small number of transfers that departs later than a given time and does not arrive at the destination too late.

Other additional criteria of interest are fares, convenience (for example, measured by the used train classes in a connection), stability of a connection in case of delays (for example, measured by the minimum buffer time of a transfer within a connection, where the *buffer time* of a transfer is the difference between the waiting time and the minimum transfer time), seat reservability (is it possible to get a seat reservation for all those parts of a connection where the used trains do in principle allow a seat reservation), etc.

Computing optimal connections under multiple criteria reduces (in a completely analogous to the single-criterion case) to the *multi-criteria or multi-objective shortest path* (MOSP) problem – a fundamental problem in the area of multi-criteria or multi-objective optimization [10]. An instance of a multi-criteria optimization problem is associated with a set of feasible solutions Q and a d-vector function $\mathbf{f} = [f_1, \ldots, f_d]^T$ (d is typically a constant) associating each feasible solution $q \in Q$ with a d-vector $\mathbf{f}(q)$ (w.l.o.g we assume that all objectives f_i, $1 \leq i \leq d$, are to be minimized). In multi-criteria optimization we are

interested not in finding a single optimal solution, but in computing the *trade-off* among the different objective functions, called the *Pareto set (or curve)* $\mathcal{P}$, which is the set of all feasible solutions in Q whose vector of the various objectives is *not* dominated by any other solution (a solution p *dominates* another solution q iff $f_i(p) \leq f_i(q)$, for all $1 \leq i \leq d$, and $f_j(p) < f_j(q)$, for at least one j, $1 \leq j \leq d$).

Multi-objective optimization problems are usually NP-hard (as indeed is the case for MOSP), since the Pareto set is typically exponential in size (even in the case of two objectives). Hence, exact methods (i.e., methods that find all Pareto optimal solutions) may be efficient under certain circumstances that depend on the particular instance of a given problem. On the other hand, even if a decision maker is armed with the entire Pareto set, s/he is still left with the problem of which is the "best" solution for the application at hand. Consequently, three natural approaches to deal with multiobjective optimization problems are to: (i) study approximate versions of the Pareto set that result in (guaranteed) near optimal but smaller Pareto sets; (ii) optimize one objective while bounding the rest (*constrained approach*); and (iii) proceed in a normative way and choose the "best" solution by introducing a utility (often non-linear) function on the objectives (*normalization or decision maker's approach*).

In the following, we will discuss the above exact and non-exact approaches in computing optimal connections under multiple criteria.

5.1 Exact Approaches

The straightforward approach in dealing with multiple criteria is to find the entire Pareto set, that is, all Pareto-optimal (i.e., feasible and undominated) solutions (connections). As mentioned above, this may be a hard problem. Even very simple instances of graphs – actually chains of parallel arcs with just two criteria – may have exponentially many different Pareto optima [14]. In certain cases, however, the cardinality of the Pareto set may not be too large and the Pareto set can be computed efficiently. In the following, we discuss such cases.

Size of the Pareto Set

If the range of valid values of some criterion is restricted to a small discrete set, adding such a criterion cannot lead to an exponential blow-up. For example, for the number of transfers we can safely assume a small constant k as an upper bound on its number in practice. Hence, if we add the number of transfers as an additional criterion to one or more other criteria, the size of the Pareto set can only increase by a factor of k.

Empirical results indicate that one may have small Pareto sets in timetable information. In a recent study with 25000 queries to the server of Deutsche Bahn AG, Müller-Hannemann and Schnee observed only 3.6 Pareto optimal connections on average if the three criteria travel time, number of transfers, and fares are used. The observed maximum was 19 Pareto-optimal connections.

Müller-Hannemann and Weihe [22] studied the size of the Pareto set also from a theoretical point of view. They considered certain characteristics found in the

application scenario in an attempt to explain the huge gap between the potentially exponentially sized Pareto set and the small sizes observed in practice.

An important characteristic of our application is that we can partition the edge set of our graph models into a small number of different *edge classes* such that each edge class has a certain semantics. Naturally, we can take the different types of edges as individual classes. Moreover, we can refine the class of those edges modeling elementary train connections into further classes derived from the type of train (*train classes*). A similar way to partition the edge set is to group them by average speed.

Average speed as the ratio of travel distance and travel time relates two criteria together. If we now assume that there are only k different average speeds (and therefore only k different edge classes), we arrive at the *ratio-restricted lengths model*. More generally than just considering speed, but still with two criteria, we assume in this model that every edge class is equipped with a value r which denotes the ratio between the length values of the first and second criterion, respectively. But even in a bicriteria model with at most $k > 1$ different ratios, the number of Pareto optima can still be exponentially large (Lemma 2.1 in [22]).

Another important characteristic of our application is that Pareto-optimal connections typically show a certain pattern with respect to the order of used train classes. Namely, if we order the train classes by their maximum speed, from slowest to fastest means of transportation, then most connections turn out to be *bitonic*: they consist of one acceleration and one deceleration phase. In the acceleration phase, we start with a slower train and from transfer to transfer we monotonically use trains with higher average speed until we reach a maximum. Then the deceleration phase starts and we use again slower and slower trains. Of course, there are exceptions: for example, in cities like Paris or London one may have to use the subway when changing between two high-speed trains. But it seems reasonable to assume that the number of changes between acceleration and deceleration phases is rather limited for Pareto-optimal connections. This model has been called *restricted non-monotonical*.

Combining the ratio-restricted lengths model with the restricted non-monotonical case it is possible to give tight polynomial worst case bounds on the size of the Pareto set (Lemma 2.3 and Lemma 2.7. in [22]).

Finding the Pareto Set

The standard approaches to the case that all Pareto optima have to be computed are generalizations of the standard algorithms for the single-criterion case (for a survey see Ehrgott and Gandibleux [10]).

The main difference is that instead of one scalar distance label, each node v maintains a list of d-dimensional distance labels assuming that we work with d criteria. Such a list contains a set of Pareto-optimal paths from the start node to v.

This immediately leads to a "Pareto version" of Dijkstra's algorithm (first described by Hansen [14] and Martins [18]). Instead of storing temporarily labeled

nodes, the priority queue now maintains d-dimensional labels. In each iteration of the main loop, we extract the lexicographic smallest label from the priority queue instead of choosing the node with smallest distance label. If v is the corresponding node to the extracted label one updates for all feasible edges of type (v, w) the list of Pareto optima stored at the head node w. More precisely, a tentative new d-dimensional label is created and compared to all labels in the list of Pareto-optima held at node w. It is only inserted into that list if it is not dominated by any other label in the list. Moreover, labels dominated by the new label are removed. For a more detailed description of the generalized Dijkstra algorithm and a correctness proof we refer to [19] and [35].

An adaption of this algorithm has been used by Müller-Hannemann and Schnee to build the timetable information server PARETO [20] which relies on the time-expanded graph model.

In [30,28], Pareto-optimal connections concerning the earliest arrival and minimum number of transfers have been considered for the time-dependent approach. In this work, it is further shown that the Pareto-set in the special case of a bicriteria problem involving the earliest arrival as one criterion can be determined in the time-expanded approach by running Dijkstra's algorithm on the realistic time-expanded graph with lexicographically ordered distance labels. The Pareto-optimal solutions are enumerated by the solutions that Dijkstra's algorithm reports at the destination station (i.e., the algorithm is not terminated until all Pareto-optimal solutions have been found).

5.2 Approximation Approaches

The ultimate goal of a traffic information system is to offer a small set of highly attractive connections as an answer to a customer query. In that respect finding the whole set of Pareto-optimal solutions bears two problems:

1. Not every Pareto-optimal solution is really noteworthy for a customer.
2. Many attractive connections are dominated only slightly.

The first of these two problems can be solved easily by filtering out unattractive connections. To tackle the second problem, we need a proper notion of approximate Pareto optimal solutions. Current research concentrates along two directions: the (recently re-investigated) concept of $(1 + \varepsilon)$-Pareto set [27], and the concept of relaxed Pareto dominance, usually called ε-efficiency [17,42].

Approximate Pareto Sets

Despite so much research in multiobjective optimization [9,10], only recently a systematic study of the complexity issues regarding the construction of approximate Pareto sets has been initiated [27]. Informally, an $(1 + \varepsilon)$-*Pareto set* $\mathcal{P}_\varepsilon$ is a subset of feasible solutions such that for any Pareto optimal solution and any $\varepsilon > 0$, there exists a solution in $\mathcal{P}_\varepsilon$ that is no more than $(1 + \varepsilon)$ away in all objectives. Although this concept is not new (it has been previously used in

the context of bicriteria and multiobjective shortest paths [14,41]), Papadimitriou and Yannakakis in a seminal work [27] show that for *any* multiobjective optimization problem there exists a $(1+\varepsilon)$-Pareto set $\mathcal{P}_\varepsilon$ of (polynomial) size $|\mathcal{P}_\varepsilon| = O((4B/\varepsilon)^{d-1})$, where B is the number of bits required to represent the values in the objective functions (bounded by some polynomial in the size of the input). They also provide a necessary and sufficient condition for its efficient (polynomial in the size of the input and $1/\varepsilon$) construction. In particular, $\mathcal{P}_\varepsilon$ can be constructed by $O((4B/\varepsilon)^d)$ calls to a GAP routine that solves (in time polynomial in the size of the input and $1/\varepsilon$) the following problem: given a vector of values $\mathbf{a}$, either compute a solution that dominates $\mathbf{a}$, or report that there is no solution better than $\mathbf{a}$ by at least a factor of $1+\varepsilon$ in all objectives. Extensions to that method to produce a constant approximation to the smallest possible $(1+\varepsilon)$-Pareto set for the cases of 2 and 3 objectives are presented in [38], while for $d > 3$ objectives inapproximability results are shown for such a constant approximation.

Apart from the above general results, there has been very recent work on improved approximation algorithms (FPTAS) for multiobjective shortest paths by Tsaggouris and Zaroliagis in [36]. In that paper, a new and remarkably simple algorithm is given that constructs $(1+\varepsilon)$-Pareto sets for the single-source multiobjective shortest path problem, which improves considerably upon previous approaches. The algorithm can be viewed as a generalization of the Bellman-Ford algorithm. It proceeds in rounds. In each round i and for each node v, the algorithm maintains a d-dimensional label representing an approximate Pareto set to all Pareto optimal s-v paths with no more than i edges (s is the source node). When an edge (u, v) is considered during round i, the algorithm performs (instead of a relaxation) an extend-&-merge operation. This operation extends the node label of u in round $i-1$ with the edge (u, v) and merges the resulting set with the label associated with v by keeping the solution that approximately dominates all other solutions. This keeps the size of all labels polynomially bounded, contrary to previous label correcting or setting approaches which used to keep all undominated solutions and thus resulting in exponentially large sets of labels.

Relaxed Pareto Dominance

Müller-Hannemann and Schnee [20] recently generalized the concept of *relaxed Pareto dominance* (also known as ε-efficiency [17,42]) and applied it to traffic information. In relaxed Pareto dominance, a solution p dominates (in the relaxed-Pareto sense) another solution q iff $f_i(p) + h_i(p, q) \leq f_i(q)$, for all $1 \leq i \leq d$, and $f_i(p) + h_i(p, q) < f_i(q)$, for at least one j, $1 \leq j \leq d$, where $h_i(p, q)$ is an appropriately chosen relaxation function. The idea is to make more pairs of connections mutually incomparable by redefining the dominance relation for certain criteria. For example, if we do not want to suppress a connection with a slightly longer travel time, say of less than 5 minutes, then we would define that connection A will dominate connection B with respect to travel time only if the travel time of A plus these 5 minutes are less or equal to the travel time of B. For more examples how to apply relaxed dominance, see [20].

5.3 Normalization Approaches

In this approach, a utility function is introduced that translates (in a linear or non-linear way) the different criteria into a common cost (utility) measure. For instance, when traveling in a traffic network one typically wishes to minimize travel distance and time; both criteria can be translated into a common cost measure (e.g., money), where the former is linearly translated, while the latter non-linearly (small amounts of time have relatively low value, while large amounts of time are very valuable). Under the normalization approach, we seek for a single optimum in the Pareto set (a feasible solution that optimizes the utility function). We distinguish between the case where all criteria are linearly translated to the common cost measure and to the case where some (or all) of the criteria are non-linearly translated.

The Linear Case – Weighted Sum of Criteria

The straightforward (and simplest) approach could be to express the relative importance of optimization criteria by weights and then to optimize a weighted sum of the criteria. This approach reduces the multi-criteria problem to a single-criterion optimization which can be solved by the standard Dijkstra algorithm provided we use a graph model where each criterion is non-negative and additive on the edges. Setting all but one weight to zero, we get the single-criterion optimization as a special case.

Such an approach has two serious drawbacks. First, it will inevitably miss many attractive connections as it will find just one single solution (and not all Pareto optima). The second drawback of such an approach lies in the choice of suitable weight parameters. Each potential customer has its own preference system, but typically this preference system is not given explicitly in terms of weight parameters. The user (customer or salesperson of a train company) of a traffic information system and/or the system itself might set the parameters incorrectly as neither of them will typically know the customer's preference system to its full extent.

The Non-linear Case

The case of non-linear utility function is the most interesting one, since it reflects realistic scenaria in traffic optimization. Experience shows that users of traffic networks value certain attributes (e.g., time) non-linearly [15]: small amounts have relatively low value, while large amounts are very valuable. Also, the vast majority of transit systems have a non-additive (non-linear) fare structure [12]. Consequently, the most interesting theoretical models for traffic equilibria involve minimizing a monotonic non-linear utility function. In this case, the problem of computing optimal connections reduces to the so-called *non-additive shortest path* (NASP) problem: given a digraph whose edges are associated with d-dimensional cost vectors, the task is to find a path that minimizes a certain d-attribute non-linear cost function.

Very recently, Tsaggouris and Zaroliagis [36] presented the first FPTAS for NASP. In particular, they showed how the FPTAS for multiobjective shortest paths can provide a FPTAS for NASP for any number of objectives and for a rather general form of a utility function that includes all polynomials of bounded degree with non-negative coefficients. For the bicriteria case, a FPTAS for NASP was independently presented in [1].

5.4 Lexicographical Ordering

One other possibility is to settle for only one specific Pareto-optimal solution: the *lexicographically first* one. Dijkstra's algorithm works not only for non-negative real edge weights, but in general for semi-rings [31]. In our case, edge weights and node labels are d-tuples, with lexicographical ordering and element-wise addition.

With the simplified version of the time-expanded approach the lexicographically first solution can be computed for any d-tuples as edge weights. For example, if $d = 2$, the first element being the travel time and the second one the number of transfers, among all fastest connections the one with the minimum number of transfers is computed. With the realistic version of the time-expanded approach only tuples can be used where the first criterion is travel time. This restriction is due to the 24-hour cycles induced by the stay-edges belonging to each station. A special case are pairs as edge weights with travel time as first criterion. In this case all Pareto-optimal solutions can be computed by Dijkstra's algorithm (cf. the last paragraph of Section 5.1).

In the time-dependent approach the edge weights are required to be non-decreasing (cf. Section 3.2). This is not necessarily true for arbitrary d-tuples as edge weights, but it can be shown that for the case $d = 2$, where the first element is the number of transfers and the second one is the travel time, the time-dependent version of Dijkstra's algorithm can be extended to find the lexicographically first solution. See [28,29,32] for further details.

6 Performance

As mentioned in the introduction, the performance of the core algorithms is crucial for a timetable information system. The average performance is particularly important in a scenario of a central server that has to answer several hundreds of (on-line) queries which are issued, for example, through the Internet or through terminals at train stations. We review the results of experimental studies involving the approaches introduced in the previous sections.

6.1 Simplified Earliest Arrival Problem

The approaches introduced in Section 3 for solving the simplified earliest arrival problem have been extensively studied, both in the time-expanded and the time-dependent approach.

Time-Expanded Approach

Schulz, Wagner, and Weihe [33] conducted an experimental study based on the time-expanded graph (cf. Section 3.1) with realistic timetable data of the German railway company Deutsche Bahn and a sample of half a million of real-world customer queries. Using a single 336 MHz Ultra-SPARC-II processor, the average running time per query of Dijkstra's algorithm was 0.103 seconds. The main contribution of the study is that it demonstrates that the average running time of Dijkstra's algorithm can be drastically improved by applying distance-preserving speed-up techniques (which still guarantee optimal solutions): a speed-up of a factor of 34 could be observed, yielding an average running time of 0.003 seconds. More precisely, they used a geometric speed-up technique based on angular sectors limiting the reachable nodes through an edge, and a graph decomposition technique based on a small "backbone graph" for finding the shortest path. Both techniques reduce the search space of the algorithm and rely on a preprocessing step in which the additional information is pre-computed. Wagner and Willhalm [40] have shown that other geometric containers are better suited and yield even higher speed-up factors, in particular bounding boxes around the reachable nodes through an edge show good results.

The second technique has been generalized by Schulz, Wagner, and Zaroliagis in [34]. They demonstrated, also conducting experiments with the same time-expanded graph as in [33], that several hierarchical levels (3 or 4 levels for the data used) of backbone graphs yield better running times than only one additional level (by a factor of roughly 3). See also [39] for a survey on speed-up techniques for shortest path algorithms.

Time-Dependent Approach

Brodal and Jacob proved in [4] by a detailed theoretical analysis of operation counts in both approaches that the time-dependent approach is more efficient than the time-expanded approach. This was also the starting point of an experimental comparison of the two approaches conducted by Pyrga, Schulz, Wagner, and Zaroliagis [30,28]. They revealed that indeed the time-dependent approach is faster than the time-expanded approach by factors in the range 12 to 40 depending on the data set (timetables consisting of French and German long-distance traffic as well as two timetables consisting of local traffic have been used).

Basically, the preprocessing speed-up techniques mentioned above for the time-expanded case can also be applied in the time-dependent approach; however, we are not aware of experimental studies dealing with this issue.

6.2 Realistic Single-Criterion Problems

In the experiments mentioned above [28], also the realistic specifications have been considered. Solving the minimum number of transfers problem is clearly faster (by a factor of roughly 4) in the train-route graph than in the realistic time-expanded graph. This is due to the fact that in this case the train-route graph is also static and smaller than the time-expanded graph.

Concerning the realistic earliest arrival problem, the picture looks different: The average running times of the time-expanded and the time-dependent approach are almost equal, only a speed-up factor of 1.5 was observed. Comparing the average running time for solving the simplified earliest arrival problem to the realistic earliest arrival problem, the time-expanded implementations solved the simplified version only slightly faster (by a factor of less than 2), while the simplified time-dependent implementation was faster by a factor of 5.

6.3 Multi-criteria Optimization

Finding all attractive connections with respect to travel time, fare, and number of interchanges is of course more expensive than just searching for a fastest connection. In the implementation of [20], such a search needs about 10 times as long as the search with a single criterion. For the multi-criteria case, we still need more effective speed-up techniques.

7 Conclusion

We have discussed time-expanded and time-dependent models for several kinds of single- and multi-criteria optimization problems for timetable information systems that provide optimal solutions via shortest paths. Extensions that model realistic requirements (like train transfers) can be integrated in both approaches.

The time-dependent approach is clearly superior with respect to performance when the simplified earliest arrival problem is considered, and speed-up factors in the range from 10 to 40 were observed. When considering extensions of the models for the solution of realistic versions of optimization problems in the single-criterion case, the performance of the two approaches is almost equal. Speed-up techniques yield running times indicating that these approaches are applicable in practice. The main open question is how these speed-up techniques—relying on additional information computed beforehand—can be extended to deal with dynamic changes of the timetable; such a change of the timetable invalidates the preprocessed information. Possibly, the additional information can be adapted by small updates to cope with both "off-line" changes like the treatment of different traffic days and "on-line" changes caused for example by accidents.

For other optimization criteria, it is more likely that the integration can be modeled directly by edge lengths in the time-expanded model than in the time-dependent model: In case the criterion can be expressed as additive costs for elementary connections, these costs induce edge lengths in the time-expanded graph. In contrast, in the time-dependent approach it is not clear if the costs can be mapped to feasible edge lengths, since only the first elementary connection per edge is considered. Because of this most studies concerning multi-criteria optimization have focused on the time-expanded approach. In that, (relaxed) Pareto-optimal solutions are desirable, and it turns out that in practice the size of the Pareto frontier is quite small, such that labeling approaches are feasible. For practical application, the multi-criteria optimization techniques provide the

most satisfactory solutions. However, further speed-up techniques are required (the techniques for the single-criterion problems cannot be directly applied for the general multi-criteria algorithms) in order to yield a performance that is acceptable for a real-world timetable information system.

References

1. Ackermann, H., Newman, A., Röglin, H., Vöcking, B.: Decision making based on approximate and smoothed pareto curves. In: Deng, X., Du, D.-Z. (eds.) ISAAC 2005. LNCS, vol. 3827, pp. 675–684. Springer, Heidelberg (2005)
2. Baumann, N., Schmidt, R.: Buxtehude–Garmisch in 6 Sekunden. Die elektronische Fahrplanauskunft (EFA) der Deutschen Bundesbahn. Die Bundesbahn. Zeitschrift für aktuelle Verkehrsfragen, 10, 929–931 (1988)
3. Brodal, G.S., Jacob, R.: Time-dependent networks as models to achieve fast exact time-table queries. Technical Report ALCOMFT-TR-01-176, BRICS, University of Aarhus, Denmark (2001), `http://www.brics.dk/ALCOM-FT/TR/ALCOMFT-TR-01-176.html`
4. Brodal, G.S., Jacob, R.: Time-dependent networks as models to achieve fast exact time-table queries. In: Proceedings of the 3rd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2003). Electronic Notes in Theoretical Computer Science, vol. 92, Elsevier, Amsterdam (2004) (A previous version appeared as [3])
5. Cooke, K.L., Halsey, E.: The shortest route through a network with time-dependent internodal transit times. Journal of Mathematical Analysis and Applications 14, 493–498 (1966)
6. DELFI. Durchgängige elektronische Fahrplaninformation, `http://www.delfi.de/`
7. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)
8. EFA. A timetable information system by Mentz Datenverarbeitung GmbH, München, Germany, `http://www.mentzdv.de/`
9. Ehrgott, M.: Multicriteria Optimization. Springer, Heidelberg (2000)
10. Ehrgott, M., Gandibleux, X.: Multiobjective combinatorial optimization. In: Multiple Criteria Optimization — State of the Art Annotated Bibliographic Surveys, pp. 369–444. Kluwer Academic Publishers, Boston, MA (2002)
11. EUSpirit. European travel information system, `http://www.eu-spirit.com/`
12. Gabriel, S., Bernstein, D.: The traffic equilibrium problem with nonadditive path costs. Transportation Science 31(4), 337–348 (1997)
13. HAFAS. A timetable information system by HaCon Ingenieurgesellschaft mbH, Hannover, Germany, `http://www.hacon.de/hafas/`
14. Hansen, P.: Bicriteria path problems. In: Fandel, G., Gal, T. (eds.) Multiple Criteria Decision Making Theory and Applications. Lecture Notes in Economics and Mathematical Systems, vol. 177, pp. 109–127. Springer, Berlin (1979)
15. Hensen, D., Truong, T.: Valuation of travel times savings. Journal of Transport Economics and Policy, 237–260 (1985)
16. Kostreva, M.M., Wiecek, M.M.: Time dependency in multiple objective dynamic programming. Journal of Mathematical Analysis and Applications 173, 289–307 (1993)
17. Loridan, P.: ε-solutions in vector minimization problems. Journal of Optimization Theory and Applications 43, 265–276 (1984)

18. Martins, E.Q.V.: On a multicriteria shortest path problem. European Journal of Operations Research 16, 236–245 (1984)
19. Möhring, R.: Verteilte Verbindungssuche im öffentlichen Personenverkehr: Graphentheoretische Modelle und Algorithmen. In: Angewandte Mathematik – insbesondere Informatik, Vieweg, pp. 192–220 (1999)
20. Müller-Hannemann, M., Schnee, M.: Finding all attractive train connections by multi-criteria Pareto search. In: Proceedings of the 4th Workshop in Algorithmic Methods and Models for Optimization of Railways (ATMOS 2004), To appear in the same volume (2004)
21. Müller-Hannemann, M., Schnee, M., Weihe, K.: Getting train timetables into the main storage. In: Proceedings of the 2nd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2002). Electronic Notes in Theoretical Computer Science, vol. 66, Elsevier, Amsterdam (2002)
22. Müller-Hannemann, M., Weihe, K.: Pareto shortest paths is often feasible in practice. In: Brodal, G.S., Frigioni, D., Marchetti-Spaccamela, A. (eds.) WAE 2001. LNCS, vol. 2141, pp. 185–198. Springer, Heidelberg (2001)
23. Nachtigal, K.: Time depending shortest-path problems with applications to railway networks. European Journal of Operations Research 83, 154–166 (1995)
24. Orda, A., Rom, R.: Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. Journal of the ACM, 37(3) (1990)
25. Orda, A., Rom, R.: Minimum weight paths in time-dependent networks. Networks, 21 (1991)
26. Pallottino, S., Scutellà, M.G.: Shortest path algorithms in transportation models: Classical and innovative aspects. In: Equilibrium and Advanced Transportation Modelling, ch. 11, Kluwer Academic Publishers, Dordrecht (1998)
27. Papadimitriou, C., Yannakakis, M.: On the approximability of trade-offs and optimal access of web sources. In: Proc. 41st IEEE Symp. on Foundations of Computer Science – FOCS 2000, pp. 86–92 (2000)
28. Pyrga, E., Schulz, F., Wagner, D., Zaroliagis, C.: Experimental comparison of shortest path approaches for timetable information. In: Proceedings of the Sixth Workshop on Algorithm Engineering and Experiments, SIAM, pp. 88–99 (2004)
29. Pyrga, E., Schulz, F., Wagner, D., Zaroliagis, C.: Towards realistic modeling of time-table information through the time-dependent approach. In: Proceedings of the 3rd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2003). Electronic Notes in Theoretical Computer Science, vol. 92, pp. 85–103. Elsevier, Amsterdam (2004)
30. Pyrga, E., Schulz, F., Wagner, D., Zaroliagis, C.: Efficient Models for Timetable Information in Public Transportation Systems. ACM Journal of Experimental Algorithmics, 12(2.4) (2007)
31. Rote, G.: Path problems in graphs. In: Tinhofer, G., Mayr, E., Noltemeier, H., Syslo, M. (eds.) Computational Graph Theory, pp. 155–190. Springer, Heidelberg (1990)
32. Schulz, F.: Timetable Information and Shortest Paths. PhD thesis, Universität Karlsruhe (TH), Fakultät Informatik (2005)
33. Schulz, F., Wagner, D., Weihe, K.: Dijkstra's algorithm on-line: An empirical case study from public railroad transport. Journal of Experimental Algorithmics, 5(12) (2000)
34. Schulz, F., Wagner, D., Zaroliagis, C.: Using multi-level graphs for timetable information in railway systems. In: Mount, D.M., Stein, C. (eds.) ALENEX 2002. LNCS, vol. 2409, pp. 43–59. Springer, Heidelberg (2002)

35. Theune, D.: Robuste und effiziente Methoden zur Lösung von Wegproblemen. Teubner Verlag, Stuttgart (1995)
36. Tsaggouris, G., Zaroliagis, C.: Multiobjective optimization: Improved FPTAS for shortest paths and non-linear objectives with applications. Theory of Computing Systems (to appear, 2007)
37. Tulp, E., Siklóssy, L.: TRAINS, an active time-table searcher. In: Eighth European Conf. on AI, pp. 170–175 (1988)
38. Vassilvitskii, S., Yannakakis, M.: Efficiently computing succinct trade-off curves. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 1201–1213. Springer, Heidelberg (2004)
39. Wagner, D., Willhalm, T.: Speed-up techniques for shortest-path computations. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 23–36. Springer, Heidelberg (2007)
40. Wagner, D., Willhalm, T., Zaroliagis, C.: Geometric containers for efficient shortest-path computation. ACM Journal of Experimental Algorithmics, 10 (2005)
41. Warburton, A.: Approximation of pareto optima in multiple-objective shortest path problems. Operations Research 35, 70–79 (1987)
42. White, D.J.: Epsilon efficiency. Jorunal of Optimization Theory and Applications 49, 319–337 (1986)