Finding All Attractive Train Connections by Multi-criteria Pareto Search

Matthias Müller-Hannemann and Mathias Schnee

Darmstadt University of Technology, Department of Computer Science,
Hochschulstraße 10, 64289 Darmstadt, Germany
{muellerh,schnee}@algo.informatik.tu-darmstadt.de
http://www.algo.informatik.tu-darmstadt.de

Abstract. We consider efficient algorithms for timetable information in public transportation systems under multiple objectives like, for example, travel time, ticket costs, and number of interchanges between different means of transport. In this paper we focus on a fully realistic scenario in public railroad transport as it appears in practice while most previous work studied only simplified models.

Algorithmically this leads to multi-criteria shortest path problems in very large graphs. With several objectives the challenge is to find *all* connections which are potentially attractive for customers. To meet this informal goal we introduce the notion of *relaxed Pareto dominance*. Another difficulty arises from the fact that due to the complicated fare regulations even the single-criteria optimization problem of finding cheapest connections is intractable. Therefore, we have to work with fare estimations during the search for good connections.

In a cooperation with Deutsche Bahn Systems we realized this scenario in a prototypal implementation called PARETO based on a time-expanded graph model. Computational experiments with our PARETO server demonstrate that the current central server of Deutsche Bahn AG often fails to give optimal recommendations for different user groups. In contrast, an important feature of the PARETO server is its ability to provide many attractive alternatives.

1 Introduction

We consider efficient algorithms for timetable information in public transportation systems (with emphasis on public railroad systems) under multiple objectives. We concentrate on three optimization criteria: travel time, fare and number of train changes. However, it is easy to add further criteria like the possibility of seat reservation or safety margins for train changes in case of delays.

Previous work. In recent years several papers studied models for timetable information. These models are based on suitably constructed digraphs on which one can apply shortest path algorithms to answer queries. Two main approaches have been proposed: the *time-expanded* [13,17,8,7,18] and the *time-dependent*

F. Geraets et al. (Eds.): Railway Optimization 2004, LNCS 4359, pp. 246–263, 2007.

approach [11,12,10,1]. In the *time-expanded* digraph we have a node for each event (departure or arrival of a train) at a station. Basically, there are two kinds of edges: train edges connecting the departure of a train with its arrival at the next station and waiting edges within a station. Fixed weights are assigned to the edges like e.g. travel time. This construction usually creates very large but sparse graphs.

In the *time-dependent* approach [1] every node represents a station and two nodes a and b are connected by an edge if there is a train that departs at a and arrives at b without stopping in between. The key idea in a *time-dependent* digraph is that the time-delay of an edge depends on the point in time the edge is used. So the weights on the edges are computed "on-the-fly". This models the phenomenon, that the delay an edge induces depends on the path that is used to reach this edge.

Most of the cited work on public railroad information systems solely considers single-criteria optimization. Next we briefly sketch the previous work on multi-criteria shortest path problems. For a more complete overview, we refer to the section on shortest paths in the recent annotated bibliography on multi-objective combinatorial optimization [2,3]. 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. Instead of one scalar distance label, each node $v \in V$ is assigned a number of k-dimensional vectors, which are the lengths of all Pareto optimal paths from s to v (clearly, for $k = 1$ the Pareto optima are exactly the distance labels). For the bicriteria case, generalizations of the standard label setting (Dijkstra's algorithm) [4] and label correcting [16] methods have been developed. In the monograph of Theune [19] algorithms for the multi-criteria case are described in detail in the general setting of cost structures over semi-rings. A *two-phase method* has been proposed by Mote et al. [5]. They use a simplex-type algorithm to find a subset of all Pareto optimal paths in the first place, and a label-correcting method to find all remaining Pareto optimal paths in the second phase.

The crucial parameter for the run time and the space consumption is the total number of Pareto optima over all visited nodes. The insight that this number is exponential in $|V|$ in the worst case has motivated the design of approximation algorithms. Hansen [4] and Warburton [20] both present a fully polynomial approximation scheme (FPAS) for finding a set of paths which are approximately Pareto optima for the bicriteria shortest-path problem. The *(resource)-constrained* or *weight-restricted shortest-path problem* [9] is a simplifying (yet still $\mathcal{NP}$-hard) variation of the bicriteria case. Here only one Pareto optimal path is to be computed, namely the one that optimizes the first criterion subject to the condition that the second criterion does not exceed a given threshold value. A theoretical study on the size of the Pareto set in practical applications appeared in [8].

Our contribution. As mentioned in the beginning, the challenge is to find all reasonable train connections meeting a query. The traditional concept of Pareto

optimality bears two problems: First, many attractive alternatives will be sorted out if classical Pareto dominance is applied. Second, some Pareto-optimal solutions may be practically useless. The latter problem can easily be overcome: we just eliminate practically useless connections in a postprocessing step. To cope with the first problem, we introduce in Section 2 the concept of *relaxed Pareto dominance* which allows us to identify also "near-optimal" solutions.

This work arose in cooperation with Deutsche Bahn Systems. Within this project we built an information server called PARETO. As described above, there has been a number of papers on time-table queries in recent years, but most of them considered only very simplified scenarios. In contrast, our aim was to incorporate all necessary details and side-constraints such that the output of our server could in principle be used even for purposes like ticketing.

In [7] we already argued that the time-expanded model might be more appropriate to model such complex scenarios than the time-dependent approach. The recent studies of Pyrga et al. [14,15] supported this claim and also indicated that the better computational efficiency of the time-dependent model diminishes greatly if the model becomes more realistic. Therefore, we use a time-expanded graph model to capture all practical requirements. Unfortunately, due to complicated fare regulations (in particular, fares are not additive on the edges in the underlying graphs) even the single-criteria optimization problem of finding cheapest connections is intractable. Therefore, we have to work with fare estimations during the search for good connections.

In principle, our system uses a generalized version of Dijkstra's algorithm for finding the Pareto-optimal set of solutions. This system has been adapted to our relaxed notion of Pareto-optimality and to the approximation of fares during the search. As speed-up techniques we use goal-directed search and improved dominance tests with the terminal station of a query. The goal-directed search is made effective by using lower bounds based on an auxiliary *station graph* (which will be defined in Section 5.1).

Finally, we provide results from a computational study. It turns out that the current version of the Deutsche Bahn server often fails to find optimal solutions for several different user groups. Moreover, our PARETO server has the advantage to offer many more attractive alternatives which the Deutsche Bahn server does not find. This is an important feature for marketing reasons to improve customer satisfaction. Namely, with the larger pool of attractive connections that PARETO delivers, the railway company may use the timetable information system to balance the usage of trains more evenly and to improve the availability of seat reservations. If there are enough good alternatives then connections with trains working to capacity may simply be suppressed in the interface of the information system.

Overview. The rest of the paper is organized as follows. First, in Section 2, we provide further background information about railway timetable information systems. In particular, we explain in detail the specification of queries and side constraints for feasible answers. We discuss different possibilities to evaluate and

to compare train connections with respect to several objectives simultaneously, and introduce our concept of relaxed Pareto dominance. In Section 3, we briefly sketch our time-expanded graph model used for timetable information and discuss it in comparison with a time-dependent graph model. In Section 4, we describe our information server PARETO. Specific speed-up and space-saving techniques are discussed in Section 5. Then, in Section 6, we present our computational results and compare them with results of the Deutsche Bahn server. Finally, we conclude with a summary and possible future extensions.

2 Railway Timetable Information Systems

2.1 Specification of Queries

A *query* to a timetable information system usually includes:

The (start or) *source station* of the connection, the *target station* and an *interval* in time in which either the departure or the arrival of the connection has to be, depending on the *search direction*, the user's choice whether to provide the interval for departure ("forward search") or arrival ("backward search").[1] Additional query options include:

Vias and duration of stay. A query may contain one (or more) so called *vias*, stations the connection has to visit and where at least the specified amount of time can be spent, e.g. from Cologne to Munich via Frankfurt with a stay of at least two hours for shopping in Frankfurt.

Train class restrictions. Each train has a specific *train class* assigned to it. These classes are high-speed trains such as the German ICE and French TGV; ICs and ECs; Interregios and the like; local trains, "S-Bahn" and subway; busses and trams. The *query* may be restricted to a subset of all *train classes*. By excluding high speed trains one might be able to find cheaper connections.

Attribute requirements. Trains have *attributes* describing additional services they provide. Such attributes are for example: "bike transportation possible", "sleeping car", "board restaurant available". A user can specify attributes a connection has to satisfy or is not allowed to have. We allow Boolean operators for specifying *attribute requirements* like: (a restaurant OR a bistro) AND bike transportation.

2.2 Connections Matching a Query

A connection needs to be feasible and has to satisfy all requirements of the query specification to match the query. Some additional feasibility requirements are:

Meta Stations and Source-/Target-Equivalents. For a passenger it might be unimportant to start at a specific stations, as long as these stations are relatively close

[1] Note that the specification of an interval is crucial for typical pre-trip queries although previous work often assumes single point intervals.

together. Virtual *meta stations* group such stations together (like the railway station and bus stops that can be found right next to each other at the central station of any city). *Source/target-equivalents* group stations together in a similar fashion, but not as a new virtual station: Every *source/target-equivalent* consists of a station and its possible replacements.

A meta-station or source/target-equivalent may replace the source and target station as well as any via in a query.

Special attributes: NotIn / NotOut. There are some train and station related attributes that do have a special meaning for the stops of a train. Although a train stops at a station, boarding or leaving the train or both may not be allowed. Especially for night and high-speed trains there are some stations near the origin of the train where one is only allowed to enter the train and some stations near the end where one is only allowed to leave it. In a night train passengers should not be disturbed by too much "traffic" inside the train. In both cases the trains should not be used only for a short transfer. Passengers are encouraged to rather use local transportation instead.

Traffic days. Most trains do not operate on a daily basis. There is a lot of change during the year. Some trains only operate on workdays, others only on Sundays. National and local holidays affect the days of operation as well as school holidays.

Interchanges. The time table data provides rules for a lower bound on the time between the arrival of a train and the departure of its connection if a change of trains occurs. Arranged from most general to most specific these are:

- *Interchange rules at stations.* Every station has two interchange times: one for interchanges between two faster (or higher valued) trains like the German ICE or French TGV and one for all other interchanges.
- *Transfers between transfer classes.* Each train is associated with a *transfer class.* The time needed for the train change depends on the transfer classes of the arriving train at arrival and the leaving train at departure. There are rules with and without dependence on the station of the train change.
- *Line to line transfers.* Similar to the *transfer classes* each train may be associated with a *line* it serves and specific rules for line changes.
- *Service to service transfers.* The most specific interchange rule gives interchange times between individual trains.

2.3 Measuring the Quality of Connections

Most timetable information systems only regard one criterion, namely *travel time.* As mentioned before we want to focus on the three criteria travel time, ticket costs, and number of interchanges. Simply minimizing any of these three independently (or all three separately) is obviously not the method of choice. In the *weighted multi–criteria* case an evaluation function c may look like:

$$c = \varphi \cdot \text{travel time} + \xi \cdot \text{number of interchanges} + \vartheta \cdot \text{ticket cost}.$$

Table 1. Example connections

Connection	*Departure*	*Travel time* (minutes)	*Number of interchanges*	*Price* (Euro)	*Pareto optimal*
c_1	7 : 30	110	2	75	
c_2	8 : 00	100	2	75	$\surd$
c_3	8 : 00	160	0	60	$\surd$
c_4	8 : 00	200	3	35	$\surd$
c_5	8 : 00	300	3	34	$\surd$
c_6	8 : 15	110	1	45	$\surd$

Different choices for the set of parameters $\{\varphi, \xi, \vartheta\}$ express the difference in importance of the three criteria. Users may never see some interesting alternatives (for them) if either they or a system/operator sets the wrong parameters.

To overcome this problem the concept of *Pareto-optimality* treats all criteria as equally important. For two given k-dimensional vectors $x = (x_1, \ldots, x_k)$ and $y = (y_1, \ldots, y_k)$, x *dominates* y if $x_i \leq y_i$ for $1 \leq i \leq k$ and $x_i < y_i$ for at least one $i \in \{1, \ldots, k\}$. Vector x is *Pareto optimal* in set X if there is no $y \in X$ that dominates x. Here, we assume for simplicity that all cost criteria shall be minimized. In our scenario we compare 3-dimensional vectors (travel time, ticket costs, number of interchanges) for our connections. Note that this approach is easily extendable to cover further criteria.

Consider the connections of Table 1: Connections c_2 to c_6 are Pareto optimal. The single-criterion and weighted-criteria approaches (for some parameters) both do not find c_6 which is probably the most promising of all connections for most people. Unfortunately, the classical Pareto approach has its drawbacks as well: Suppose connection c_6 does not exists in the list. Although connection c_1 is dominated by c_2 it still arrives earlier at its destination. A passenger using a timetable information system at the departure station might prefer c_1 as it leaves more time to get to his final destination from the target station instead of waiting 30 minutes at the departure station. In spite of being Pareto optimal connection c_5 is of no practical use at all. The almost as cheap alternative c_4 is much more attractive.

Relaxed Dominance

To tackle the drawbacks of the simple Pareto dominance approach we *relax* the dominance rule in the *relaxed Pareto dominance* case. This means that more pairs of connections become mutually incomparable. In addition to the cost criteria travel time, travel fare and number of train changes, further aspects are taken into account to define the smaller relation between connections.

Formally, we now consider n-dimensional (integral or real-valued) vectors $x = (x_1, \ldots, x_k, x_{k+1}, \ldots, x_n) \in S$ where the first k components are cost criteria and the remaining $n - k$ components encode additional data (like departure and arrival time, highest used train class). Furthermore, for each cost criterion we have a non-negative *relaxation function* $f_i : S \times S \mapsto \mathbb{R}_0^+ \cup \{+\infty\}$. For

any two $x, y \in S$ we now define that x *dominates* y *(in the relaxed sense)* if $x_i + f_i(x, y) \leq y_i$ for all $1 \leq i \leq k$ and $x_i + f_i(x, y) < y_i$ for at least one $i \in \{1, \ldots, k\}$. Note, that in order to be able to apply relaxed Pareto dominance during search and/or for final filtering of the connections, it is essential that dominance is a transitive relation. This restricts the set of reasonable relaxation functions. Next we give examples how to specify suitable relaxation functions f_i.

- The travel time spent for getting less expensive connections has to yield a fair *hourly wage*, say of Δ Euros per hour. (In the examples of Table 1 an hourly wage of less than one Euro is not enough to make connection c_5 worth considering.) This can be modeled as follows. Suppose we want to compare connections A and B with associated costs c_A, c_B in Euros and travel times t_A, t_B in minutes, respectively. Then connection A dominates B with respect to the cost criterion only if

$$c_A + \frac{\max\{t_A - t_B, 0\}}{60} \cdot \Delta < c_B.$$

- The larger the time difference between the departure and arrival times of two connections is, the less these connections should influence each other.
  Suppose we want to compare connections A and B which have departure times d_A, d_B, arrival times a_A, a_B and travel times t_A, t_B (all data given in minutes), respectively. Then connection A dominates B with respect to the criterion travel time if

$$t_A + \alpha(t_A) \cdot \Delta(A, B) + \beta(t_A) < t_B,$$

  where, e.g., we may choose $\alpha(t_A) := t_A/360$ and $\beta(t_A) := 5 + \sqrt{t_A}/4$, and define

$$\Delta(A, B) = \begin{cases} 0 & \text{if } d_A > d_B \text{ and } a_A < a_B, \\ \min\{|d_A - d_B|, |a_A - a_B|\}, & \text{otherwise .} \end{cases}$$

- Different kinds of connections shall not dominate each other (e.g., connections using an event train (e.g. a special train to a sports event) or night trains). Using night trains one does not want to arrive as fast (and/or cheap) as possible but has the chance to arrive relaxed and even save a night's stay at a hotel. Both these alternatives should not be dominated by connections using other kinds of transportation. This can be modeled by defining a relaxation function to be $+\infty$ if the encoding of the train class attributes forbids a mutual domination.

Incomparable connections do not dominate each other, thus attractive alternatives are not suppressed. It is easy to check that all the proposed relaxation functions preserve the desired transitivity of our Pareto dominance relation. In Section 4.2 it will turn out that this concept can also be used to handle special tariffs of pricing systems.

Our overall goal is to determine the complete set of connections not dominated by relaxed dominance. However, some other aspects are still not covered,

like the stability of a connection, i.e. how high is the possibility of getting all interchanges, the aim to use a sleeping cart as long as possible during the night, the maximization of a stay at "nicer" locations, lovely panorama etc.

3 Modeling Timetable Information in Graphs

3.1 A Time-Expanded Graph Model for a Realistic Scenario

Schulz et al. [17] introduced the time-expanded model to compute earliest arrival paths which assumed zero interchange times. Modifications to allow the counting of train interchanges with $\{0, 1\}$-weights in a Dijkstra-search have been proposed in [8,7,15]. The extension of Pyrga et al. [15] models also some basic interchange rules with the concept of change nodes: In their time-expanded digraph there is one node for each departure and arrival event of every train. The departure of a train at one station and its arrival at the next station are connected with a *train edge*. The so-called *change nodes* are copies of all arrival and departure events. A *waiting edge* is introduced between each change node and the next change node in time at the same station (introducing a waiting arc over midnight between the last and first change node at that station as time is taken modulo a single day). Between the arrival of a train and its departure at the same station there is a *stay in train edge* in the graph.

We further extended the model to cover all interchange rules and the special attributes NotIn/NotOut (cf. Section 2.2): If boarding is permitted we have an *entering edge* from the change node copy to the original departure node. If leaving a train is possible, we have one *leaving edge* connecting the arrival with the change node at the point in time from which on all other events are reachable, i.e. the time difference of this node to the arrival is the maximum over the interchange times required by all change rules concerning this train at this station. For all trains reachable before this point in time we have *special interchange edges* from the arrival to the departure nodes of the corresponding trains. The first two types of edges can also be found in the model by Pyrga et al. but with different semantics.

It is easy to see that we have indeed covered all interchange rules and the attributes *NotIn/NotOut*. Take a look at Fig. 1 (left) for an example. Arriving with train t we here can either stay in train t (use stay in train edge e) or change to t^* what is possible due to some interchange rule e.g. service to service transfer (special interchange edge f). However, we can not take t' (for example, if the minimum interchange time at the station does not allow this). Therefore, we needed the special interchange edge to reach t^* and not to reach t' from t although entering t' is allowed from the change level. Every event from time b on is again reachable (leaving edge g to change node at time b), e.g., we can take train t'' (via entering edge h). Train t' stops at the station only for boarding (no leaving edges for the arrival at time a).

Traffic days, possible attribute requirements and train class restrictions with respect to a given query can be handled quite easily. We simply mark train edges as *invisible* for the search if they do not meet all requirements of the given query.

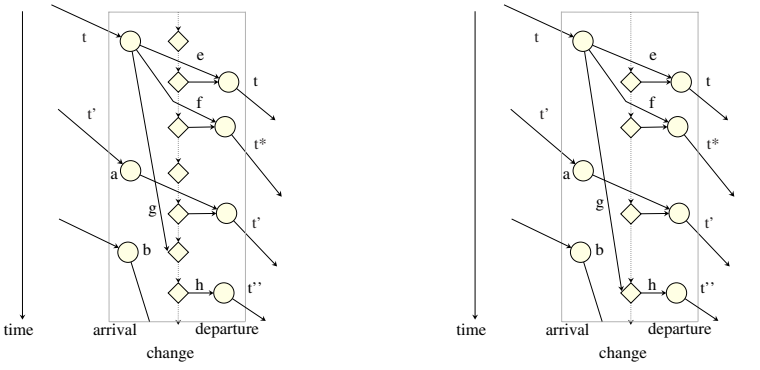

Fig. 1. Time expanded model without special interchange edges (left) and an extension to skip arrival change nodes in forward search (right) cf. Section 5.4

With respect to this visibility of edges, there is a one-to-one correspondence between feasible connections and paths in the graph.

We associate component-wise non-negative cost vectors to the edges. Here we describe only the choice for forward search, the necessary modifications for backward search should be obvious. For the cost criterion travel time, the cost for edge $e = (v, w)$ is the difference between the timestamps of the nodes w and v. For the cost criterion number of train changes, all entering edges and all special interchange edges get a cost value of 1, and all other edges a value of 0. (More precisely, such an assignment counts the number of used trains.) Ticket costs are more difficult to handle. We come back to this issue in Section 4.2.

3.2 Discussion: Time-Expanded vs. Time-Dependent Models

For single–criteria shortest path search, the time-dependent model seems to be more attractive due to its smaller sized graph that drastically speeds up the search. This advantage does not hold for more realistic scenarios. In recent experiments for computing all Pareto optima for two criteria Pyrga et al. [14] did not show a big advantage for time-dependent models. Not only did the size of the graph significantly grow due to their modeling of constant transfer times (which are still far from reality), the computational time required for solving the problem on the time-expanded graph was only 58% higher than for the time-dependent graph. Their constructions in [14] show how difficult the extension to model more realistic scenarios is. The general interchange rules and the special attributes disallowing boarding and deboarding even violate assumptions made for the most realistic model of interchanges known for time-dependent graphs (i.e. taking a later connection might allow for taking an earlier connection at some subsequent station). The time-dependent approach is not as easily extendable to cover traffic days, attribute requirements and train class restrictions as the time-expanded approach as already observed in [7].

Considering time intervals instead of points in time for possible departures requires either lists of labels for each node or computing a solution for each node in the interval. The latter is surely not reasonable for larger intervals. In summary, the time-expanded model is much more flexible and extensible (for possible further extensions see Section 7). Therefore, we used the time-expanded model for our algorithm.

4 The Information Server PARETO

4.1 The Search Algorithm in PARETO

Our algorithm is a "Pareto version" of Dijkstra's algorithm using multi-dimensional labels. See Möhring [6] or Theune [19] for a general description and correctness proofs of this approach.

We keep the travel time, number of interchanges, ticket costs (cf. Section 4.2) and some additional information in the labels. For every node in the graph we maintain a list of labels that is not dominated by any other label at this node. Every time a node is extracted from the priority queue, its outgoing edges are scanned and (if they are not infeasible due to traffic days, attributes and train class restrictions etc.) labels for their head nodes are created. Such a new label is compared to all labels in the list at the head node and only inserted into that list and into the priority queue if it is not dominated by any other label in the list. On the other hand, labels dominated by the new label are removed. For changes to this basic algorithm see the rest of this Section (for modeling reasons) and Section 5 (for space and time consumption reasons).

4.2 Modeling Ticket Costs

Pricing systems of railway companies are very complex. Unfortunately, ticket costs are typically not proportional to the distance traveled. The cost of the distance traveled in one train depends not only on the number of kilometers but also on its train class and other train classes used in the connection. Currently there are different supplementary fares for the different higher speed train classes. Furthermore, the system undergoes rapid change. In building timetable information systems it should not be the task to rebuild pricing components.

To be resistant to the changes in the pricing system (to some degree) we have a black-box pricing component (BPC) that can be used to calculate the ticket cost for some connection. Unfortunately, one call to this black-box routine is very costly: The path information stored in labels has to be converted into structures for the BPC. Much additional information like attributes on the train edges has to be set to get a correct price. Therefore, it is not at all possible to calculate the correct price for every label and achieve a bearable running time.

As a consequence we use price estimates in the labels that are updated during the search. The distance between the two stations of a train edge is taken as the straight line distance obtained from the coordinates of the stations. For

every train edge the price estimate is increased by the distance times a factor depending on the train class used. The supplementary fare is paid once and only for the highest train class involved.

This simplified model provides helpful estimates for the search. In order not to loose low cost connections due to this approximation we need a safety margin which is incorporated into the corresponding relaxation function for the relaxed Pareto dominance. Here another benefit of the Pareto relaxation (compare Section 2.3) comes into play, enabling us to model some exceptions: For example, there might be a special offer (like "Schönes Wochenende-Ticket" in Germany) for traveling on weekends for a fixed price independent of the distance but valid only on non-high speed trains. The relaxation allows us not to compare connections using no high speed trains to connections with high speed trains on weekends. After a search is completed, all connections are correctly priced by the BPC and relaxed Pareto dominance can be applied to true fares.

5 Speed-Up and Space-Saving Techniques

5.1 The Station Graph for Lower Bounds

For the strategies goal direction (Section 5.3) and to discard labels dominated by labels at the target station (Section 5.2) lower bounds for the distance from any node n to the target are required.

Let us consider lower bounds for the criterion travel time: Regarding space efficiency, it is not reasonable to store a precomputed lower bound for every pair of stations. Thus, these values must be computable "on the fly" during the search. One easy approach for calculating a lower bound on the remaining travel-time is to calculate the straight-line distance from the station $S(n)$ of node n to the target station Ω and divide this value by the fastest travel speed of all trains in the data, as used for example by Schulz, Wagner and Weihe [17]. Empirical testing revealed that this method leads only to a small speed-up.

Our idea, giving tighter lower bounds, uses the *station graph*. This graph consists of one node per station and we insert an arc from station A to B if there is a direct connection and take as the travel time the minimum among all such connections (not considering traffic days). If we simply reverse all edges in the *station graph* we may use one Dijkstra-search on the *station graph* starting at the target station Ω at the beginning of each search and get a lower bound on the travel time to Ω for *every* station or the information that no connection to Ω exists.

5.2 Domination by Labels at the Terminal

To reduce the number of labels investigated during the search, a simple heuristic improvement can be utilized that relies upon the simple fact, that if P is a Pareto optimal path, then any subpath P' of P must also be Pareto optimal.

We use the following lower bounds on the cost of a path from node v to station Ω for the three criteria:

- The value from the station graph as lower bounds on travel time.
- The trivial lower bound zero for the number of interchanges.
- The straight line distance from the station $S(v)$ of v to Ω multiplied by the cost for traveling in the lowest train class for the ticket cost.

We maintain a list of strict Pareto optimal labels at the terminal station Ω. Not all relaxed Pareto optimal labels are stored in that list to keep it short. Every new label is checked against each label in this list. If the values of the label plus the lower bounds are dominated by any label in the list there is no need to further regard the new one and it is not inserted into the priority queue.

5.3 Goal-Direction in PARETO

We use travel time as the criterion for goal direction and only fall back on the number of interchanges for breaking ties. The smaller relation for the priority queue orders the labels according to the sum of the time traveled so far, the lower bounds for the travel time to the target station (computed via the station graph (see Section 5.1)) plus γ times the number of interchanges for some constant $\gamma > 0$. Although the search cannot be terminated once the first label at the target station Ω is extracted from the priority queue using goal direction, labels at Ω are generated fairly early in the search process, thus improving the efficiency of the strategy from the previous section.

5.4 Skipping Arrival Events

Having a arrival and d departure nodes on the change level results in $2 \cdot (a + b)$ waiting edges ($a + d$ for forward and backward search each). In forward search there is no need to consider arrival events except the ones that are extracted from the priority queue (inserted as the target of a feasible train edge). Thus we may arrange the change nodes in two cycles, one by linking the departure change nodes with waiting edges according to increasing time values (for forward search) and the other by linking arrival change nodes ordered by decreasing time value (see Figure 1 right). Applying this construction we only need d waiting edges for forward search and a for backward search. Thus we save half of the edges and operations on the change level.

5.5 The Impact of Speed-Up Techniques

We ran our algorithm on over 5000 queries stemming from an internet server of Deutsche Bahn AG (original customer queries). They include forward and backward searches with and without train class restrictions and attribute requirements but no vias. The time table data was prepared for one week and all queries were shifted to the corresponding weekday in that week. The resulting graph has about 1 million original arrival and departure nodes and half a million of train edges and about 1.8 million additional edges. Up to now the sole purpose of our implementation was to countercheck the results of the server

Table 2. The impact of various speed-up techniques. The percentages (right columns) give the increase compared to the run with all speed-up techniques activated.

activated Speed-Ups	all	noDom		noGoal		noSkip		noSkip + noGoal	
given in	k	k	%	k	%	k	%	k	%
PQMinOps	309	2302	526	459	49	526	70	776	144
LabelsUsed	599	4544	659	884	48	802	34	1145	91
LabMaxAct	223	949	326	316	42	376	69	523	135
DomTarget	81	0	-100	77	-5	80	-2	60	-26
StationsHit	1.7	5.5	215	2.1	21	1.7	0	2.1	19
PQ Before	17	17	0	169	920	26	62	266	1507

currently used by Deutsche Bahn AG. Speed considerations were only secondary goals. The algorithm solves queries with an average computational time of less than 5 seconds on an Athlon XP 2100+ PC with 1 Gigabyte of RAM both under Unix using the GNU Compiler version 2.95.3 and under Microsoft Windows using MS Visual Studio 6.0.

As we did not fine tune the computational efficiency of our prototype, we consider in the following only operation counts to study the impact of the proposed heuristics. Computational results are shown in Table 2. The column headings describe the activated speed-up techniques for the corresponding column:

all. All speed-up techniques activated.
noDom. Domination by a label at the terminal station (see Section 5.2) is deactivated.
noGoal. Goal-direction (see Section 5.3) is deactivated.
noSkip. The modification of the graph in the skipping arrival events heuristic (see Section 5.4) is not done.

The rows in Table 2 give the average numbers (left) or the increase (in percent on the right) compared to the run with all speed-up techniques turned on. We used the following key values as operation counts:

- The number of extractMin()-operations on the priority queue for the whole search (*PQMinOps*) and before creation of the first label at the target station (*PQBefore*).
- The number of labels used for the search (*LabelsUsed*).
- The maximum number of labels active at the same time (*LabMaxAct*). This is the minimum number of labels that must fit into the main storage.
- The number of labels dominated by other labels at the target station (*DomTarget*).
- The number of stations that were hit by the search (*StationsHit*).

We now take a look at the three main speed-up techniques.

Dominated by labels at terminal. This technique is powerful in bounding the number of operations and labels needed for the search. When combined with

goal direction it makes the search much faster and significantly reduces the memory consumption. Deactivating this strategy slows the search down by a factor of over 5. It leads to the exploration of nearly 3 times as many stations and requires about 4 times as many labels in memory at the same time.

Goal direction. This technique produces labels at the target station fairly early in the search and is instrumental to significantly improving the performance of strategy "dominated by labels at terminal". Deactivating goal direction results in an increase of nearly 50% of the first four key values. After deactivation about 10 times as many operations as with goal direction are needed to reach the terminal station.

Skip arrival events. This strategy reduces the number of extractMin()-operations on the priority queue. Unnecessary labels for waiting are neither created nor inserted into nor extracted from the priority queue.

6 Comparison to the Deutsche Bahn Server

In this section we want to present a comparison between the quality of results computed by PARETO and the server currently used by Deutsche Bahn AG. The comparison was performed by Deutsche Bahn Systems personnel due to licensing reasons concerning the Deutsche Bahn server. We will present a measurement of quality based on three costumer groups as utilized internally by Deutsche Bahn AG. PARETO was delivered to DB systems without knowing the utility functions beforehand (i.e. we did *not* fine tune PARETO to look good with respect to these functions). Various quality distributions with respect to these groups are shown.

6.1 Utility Functions for Customer Groups

A weighted multi–criteria function is used to evaluate the quality of a connection c that looks like this (compare Section 2.3):

$$N(c) = \varphi \cdot \text{travel time} + \xi \cdot \text{number of interchanges} + \vartheta \cdot \text{ticket cost}.$$

There are three groups of costumers, each with another set of parameters $\mathcal{P} = (\varphi, \xi, \vartheta)$. These parameters were chosen by Deutsche Bahn Systems.

- The first group of costumers is mainly interested in travel time. A representative is a *businessman.* The parameters are $\mathcal{P} = (100, 1000, 1)$.
- The second group of costumers is mainly interested in convenience, i.e. the number of interchanges. Travel time is not unimportant, too. This group may contain a family with children, elderly people or a *handicapped person.* The parameters are $\mathcal{P}' = (12.5, 1500, 1)$ for this group.
- The third group of costumers is mainly interested in ticket cost. No other group is willing to spend so much time (hourly wage of 4.80 Euro for traveling longer) or to accept additional interchanges to save so little money. A representative is a *student.* The parameters are $\mathcal{P}^* = (8, 5, 1)$.

6.2 Quality Distribution for Costumer Groups

All connections are evaluated for all three costumer groups independently. The best connection for a group c_{opt} found by one of the two servers is used to normalize the value. The *quality points* for connection c are positive integers not greater than 100: $Q(c) = \lfloor N(c_{opt})/N(c) \cdot 100 \rfloor$.

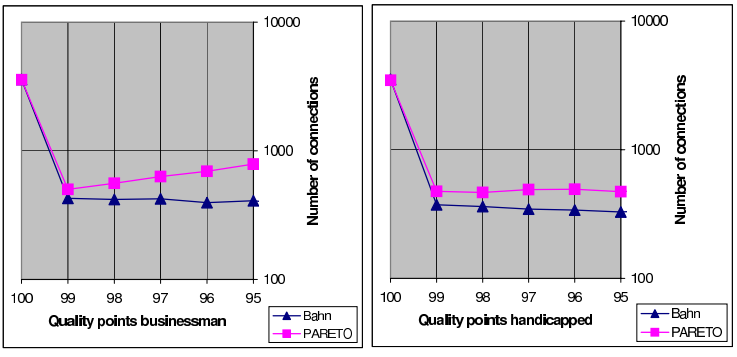

Fig. 2. Quality points for groups businessmen and handicapped. The number of connections is given in logarithmic scale.

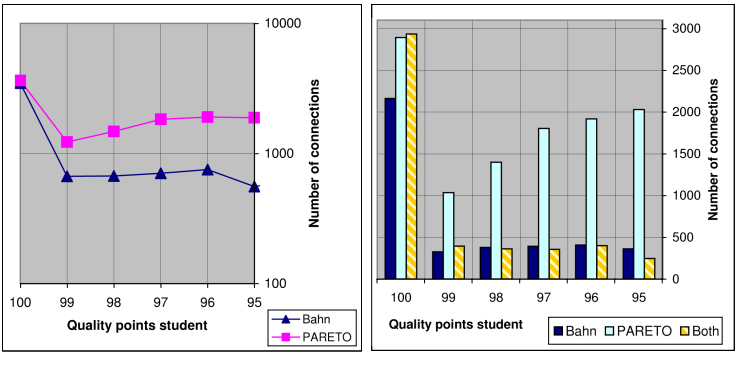

Fig. 3. Quality points for group student. On the right, we show a partition of the number of found connections with respect to the two servers.

For each user group, we evaluated the achieved quality points independently, see Figs. 2 and 3. Note that the number of connections is given in logarithmic scale. As we are only interested in "very good" connections we will not show connections with a value below 95. In this region of highest quality between 95 and 100 quality points, PARETO manages to outperform the Deutsche Bahn server significantly for every user group. (The gap becomes even larger for ranges below 95.) A deeper look into the results further revealed that the connections

found by the two servers differ from each other surprisingly often. In Fig. 3 (right part), we display for the students group the partition of all connections into those found solely by the Deutsche Bahn server or by PARETO, respectively, and those found by both of them. Again, PARETO clearly delivers many more alternatives of very high quality above 94 points. The figures for the other user groups look rather similar.

7 Conclusion and Outlook

In this paper we have presented the concept of our timetable information server PARETO. Its major goal is to search for all "reasonably attractive" train connections for given customer queries and different groups of users. To formalize this goal and to avoid some weaknesses of ordinary Pareto dominance in multi-criteria optimization, we introduced the concept of *relaxed Pareto dominance*. Based on a fully-realistic time-expanded graph model, PARETO outperforms with respect to quality the current state of the art timetable information server of Deutsche Bahn AG. We briefly sketched some implementation issues especially concerning space and time consumption and experimental results.

There are many more interesting aspects in the field of timetable information. In the future, we plan to consider some of these and add extensions to the functionality of our tool. Such extensions are for example:

Secure interchanges. All connections where the time for a train change is not less than the minimum time needed for the change with respect to the interchange rules are valid. Connections providing a buffer time of k minutes for some $k > 0$ have a lower possibility of missing a train due to a delay of the arriving train. Such secure alternatives are especially noteworthy if a connection is the last connection of a day, i.e. a missed train would result in a night's stay to take the first train in the morning.

Guaranteed interchanges. The time buffer for an interchange is calculated as the difference between the arrival and departure of the trains involved minus the time needed for the interchange. Some connections have *guaranteed interchange times*. For example, ICE A waits for ICE B at station S at least 20 minutes. Although the interchange would be considered insecure for a calculated buffer of less than 2 minutes, the real buffer now is 20 minutes and that is pretty safe. Similar rules exist for the last busses in evenings that wait for the last trains to arrive.

Avoiding highly frequented trains. Trains often working to capacity currently have a static special attribute in the timetable data indicating high usage. If trains with known potential of working to capacity appear among the best connections, a timetable information system should be able to produce alternatives with less frequently used trains.

Seat reservability. Currently connections using identical trains in the same order are equivalent. If information about the reservation possibilities is available

the location of train changes becomes important. The possibility of reservation throughout the whole connection might be possible for some orders of train changes and not for others.

Search for special offers. Every tariff that enables costumers to buy a ticket for a price below the actual standard ticket cost is considered a special offer (e.g. the already mentioned "Schönes Wochenende" ticket for non high speed connections). Basically there are two categories of special offers, one with and the other without quotas limiting the availability of the offer. Offers subject to quotas require very short updating periods for the data concerning the current quotas. Active search for such special offers (with or without quotas) is an important extension to finding cheaper connections.

Search for event trains. Special event trains are introduced for the purpose to bring participants jointly to events like soccer matches, the "love parade" in Berlin or the "Oktoberfest" in Munich. Such trains should be offered to the target group. Hence, it is necessary to be able to inform participants of such events explicitly about these trains even if they are not as good as others by standard criteria.

Acknowledgments

This work was done in cooperation with Deutsche Bahn Systems, Frankfurt am Main, Germany, and datagon GmbH, Waldems, Germany. In particular, we wish to thank Wolfgang Sprick for fruitful discussions and close collaboration in the development of PARETO.

References

1. Brodal, G.S., Jacob, R.: Time-dependent networks as models to achieve fast exact time-table queries. In: Proceedings 3rd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2003). Electronic Notes in Theoretical Computer Science, vol. 92, pp. 3–15. Elsevier, Amsterdam (2003)
2. Ehrgott, M., Gandibleux, X.: An annotated biliography of multiobjective combinatorial optimization. OR Spektrum 22, 425–460 (2000)
3. Ehrgott, M., Gandibleux, X. (eds.): Multiple Criteria Optimization: State of the Art Annotated Bibliographic Survey. Kluwer Academic Publishers, Boston (2002)
4. 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)
5. Mote, J., Murthy, I., Olson, D.L.: A parametric approach to solving bicriterion shortest path problems. European Journal of Operations Research 53, 81–92 (1991)
6. Möhring, R.H.: Verteilte Verbindungssuche im öffentlichen Personenverkehr: Graphentheoretische Modelle und Algorithmen. In: Angewandte Mathematik - insbesondere Informatik, Vieweg, pp. 192–220 (1999)

7. Müller-Hannemann, M., Schnee, M., Weihe, K.: Getting train timetables into the main storage. In: Proceedings 3rd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2002). Electronic Notes in Theoretical Computer Science, vol. 66, Elsevier, Amsterdam (2002)
8. 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)
9. Mehlhorn, K., Ziegelmann, M.: Resource constrained shortest paths. In: Paterson, M.S. (ed.) ESA 2000. LNCS, vol. 1879, pp. 326–337. Springer, Heidelberg (2000)
10. Nachtigal, K.: Time depending shortest-path problems with applications to railway networks. European Journal of Operations Research 83, 154–166 (1995)
11. Orda, A., Rom, R.: Shortest paths and minimum-delay algorithms in networks with time-dependent edge-length. Journal of the ACM 37(3), 607–625 (1990)
12. Orda, A., Rom, R.: Minimum weight paths in time dependent networks. Networks 21, 295–319 (1991)
13. Pallottino, S., Scutellà, M.G.: Equilibrium and advanced transportation modelling, ch. 11. Kluwer Academic Publishers, Dordrecht (1998)
14. Pyrga, E., Schulz, F., Wagner, D., Zaroliagis, C.: Towards realistic modeling of time-table information through the time-dependent approach. In: Proceedings 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 (2003)
15. Pyrga, E., Schulz, F., Wagner, D., Zaroliagis, C.: Experimental comparison of shortest path approaches for timetable information. In: Proceedings 6th Workshop on Algorithm Engineering and Experiments and the 1st Workshop on Analytic Algorithmics and Combinatorics, SIAM, pp. 88–99 (2004)
16. Skriver, A.J.V., Andersen, K.A.: A label correcting approach for solving bicriterion shortest path problems. Computers and Operations Research 27, 507–524 (2000)
17. Schulz, F., Wagner, D., Weihe, K.: Dijkstra's algorithm on-line: An empirical case study from public railroad transport. ACM Journal of Experimental Algorithmics, 5(12) (2000)
18. Schulz, F., Wagner, D., Zaroliagis, C.: Using multilevel 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)
19. Theune, D.: Robuste und effiziente Methoden zur Lösung von Wegproblemen. Teubner Verlag, Stuttgart (1995)
20. Warburton, A.: Approximation of pareto optima in multiple-objective shortest path problems. Operations Research 35, 70–79 (1987)