Experimental Comparison of Shortest Path Approaches for Timetable Information*

Evangelia Pyrga† Frank Schulz‡ Dorothea Wagner‡ Christos Zaroliagis†

Abstract

We consider two approaches that model timetable information in public transportation systems as shortest-path problems in weighted graphs. In the *time-expanded* approach every event at a station, e.g., the departure of a train, is modeled as a node in the graph, while in the *time-dependent* approach the graph contains only one node per station. Both approaches have been recently considered for the earliest arrival problem, but little is known about their relative performance. So far, there are only theoretical arguments in favor of the time-dependent approach. In this paper, we provide an extensive experimental comparison of the two approaches. Using several real-world data-sets we evaluate the performance of the basic models and of several extensions towards realistic modeling. Furthermore, new insights on solving bicriteria problems in both models are presented. The time-expanded approach turns out to be more robust for modeling more complex scenarios, whereas the time-dependent approach shows a clearly better performance.

1 Introduction

An important problem in public transportation systems is to model timetable information so that subsequent queries asking for optimal itineraries can be efficiently answered. The main target that underlies the modeling (and which applies not only to public transportation systems, but also to other systems as well like route planning for car traffic, database queries, web searching, etc) is to process a vast number of on-line queries as fast as possible. In this paper, we are concerned with a specific, query-intensive scenario arising in public railway transport, where a central server is directly accessible to any customer either through terminals in train stations or through a web interface, and has to answer a potentially infinite number of queries. The main goal in such an application is to reduce the average response time for a query.

Two main approaches have been proposed for modeling timetable information: the *time-expanded* [5, 9, 11, 12], and the *time-dependent* approach [1, 6, 7, 8]. The common characteristic of both approaches is that a query is answered by applying some shortest path algorithm to a suitably constructed digraph. The time-expanded approach [11] constructs the time-expanded digraph 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 problem that we want to solve (see below), the construction assigns specific fixed weights to the edges. This naturally results in the construction of a very large (but usually sparse) graph. The time-dependent approach [1] constructs the time-dependent digraph 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 weights on the edges are assigned "on-the-fly", i.e., the weight of an edge depends on the time in which the particular edge will be used by the shortest path algorithm to answer the query.

The two most frequently encountered timetable problems are the earliest arrival and the minimum number of transfers problems. In the *earliest arrival* problem, the goal is to find a train connection from a departure station A to an arrival station B that departs at A later than a given departure time and arrives at B as early as possible. There are two variants of the problem depending on whether train transfers within a station are assumed to take negligible time (*simplified* version) or not. In the *minimum number of transfers* problem, the goal is to find a connection that minimizes the number of train transfers when

*This work was partially supported by the IST Programme of EU under contract no. IST-1999-14186 (ALCOM-FT), by the Human Potential Programme of EU under contract no. HPRN-CT-1999-00104 (AMORE), and by the DFG under grant WA 654/1-12.

†Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece, and Department of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece. Emails: {pirga,zaro}@ceid.upatras.gr.

‡University of Karlsruhe, Department of Computer Science, P.O. Box 6980, 76128 Karlsruhe, Germany. Emails: {fschulz,dwagner}@ira.uka.de.

considering an itinerary from A to B. We consider also combinations of the above problems as bicriteria problems.

Techniques for solving general multi-criteria problems have been discussed in [4, 5], where the discussion in [4] is focused on a distributed approach for timetable information problems. Space consumption aspects of modeling more complex real-world scenarios is considered in [3]. For the time-expanded model, the simplified version of the earliest arrival problem has been extensively studied [11, 12], and an extension of the model able to solve the minimum number of transfers problem, but without transfer times, is discussed in [5]. For the time-dependent model, several extensions to that model are proposed in [10] including transfer times and the minimum number of transfers problem. Comparing the time-expanded and time-dependent approach, it is argued theoretically in [1] that the time-dependent approach is better than the time-expanded one when the simplified version of the earliest arrival problem is considered.

In this paper, we provide the first experimental comparison of the time-expanded and the time-dependent approaches with respect to their performance in the specific, query-intensive scenario mentioned earlier. For the simplified earliest arrival problem we show that the time-dependent approach is clearly superior to the time-expanded approach. In order to cope with more realistic requirements, we investigate, besides the extensions to train transfers in combination with the earliest arrival problem proposed in [5, 10], additional new extensions of both approaches. In particular, the proposed extensions can handle cases not tackled by most previous studies for the sake of simplification. These new cases are: (a) the waiving of the assumption that transfer of trains within a station takes negligible time; (b) the consideration of the minimum number of transfers problem; (c) the involvement of traffic days; and (d) the consideration of bicriteria problems combining the earliest arrival and the minimum number of transfers problems.

We also conducted extensive experiments comparing the extended approaches. That comparison is important, since the described extensions are mandatory for real-world applications, and (to the best of our knowledge) nothing is known about the relative behavior of realistic versions of the two approaches.

In Section 2 the variants of itinerary problems that are considered in this paper are defined. The modeling of the earliest-arrival problem is considered in Section 3, where first the basic ideas of the time-expanded and time-dependent models are briefly reviewed and then the realistic extensions of these approaches are presented. Sections 4 and 5 discuss how the minimum number of transfers problem and the bicriteria problems, resp., can be solved in either of the extended models. The experimental comparison of the two approaches based on real data from the German railways is presented in Section 6. We first consider how the plain versions of the two approaches compare, and subsequently investigate the extensions and bicriteria problems. Section 7 summarizes our insights on the advantages and disadvantages of the approaches under comparison.

2 Itinerary Problems

In this section, we provide definitions of the timetable problems that we will consider. 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*. We define an *elementary connection* c to be a 5-tuple of the form $c = (Z, S_1, S_2, t_d, t_a)$ and interpret it as train Z leaves station S_1 at time t_d, and the *immediately next* stop of train Z is station S_2 at time t_a. The time values t_a and t_d are integers in the interval $[0, 1439]$ representing the time in minutes past midnight. The *length* of elementary connection c, denoted by $length(c)$, is $t_a - t_d \pmod{1440}$. We generally assume that trains are operated daily, unless stated otherwise, as e.g., in Section 3.3. There, we discuss the integration of *traffic days*: for each elementary connection we are given additionally one bit per day indicating whether that particular connection is operated on that day. If x denotes a tuple's field, then the notation $x(c)$ specifies the value of x in the elementary connection c. The timetable induces a set $\mathcal{C}$ of elementary connections. At a station S it is possible to *transfer* from one train to another. Such a transfer is only possible if the time between the arrival and the departure at that station S is larger than or equal to a given, station-specific, *transfer time*, denoted by $transfer(S)$.

A sequence of elementary connections $P = (c_1, \ldots, c_k)$ together with departure times $dep_i(P)$ and arrival times $arr_i(P)$, $1 \leq i \leq k$, is called a *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; 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. We also assume that the times $dep_i(P)$ and $arr_i(P)$ include data regarding the departure/arrival day by counting time in minutes from the first day of the timetable. Such a time 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 $\lfloor t/1440 \rfloor$.

For the timetable information problem we are additionally given a large, on-line sequence of *queries*. A query 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. In this work, we are concerned with two of the most important criteria, namely the earliest arrival (EA) and the minimum number of transfers (MNT), and consequently investigate two single-criterion and a few bicriteria optimization problems which are defined next.

Earliest Arrival Problem (EAP). A query (A, B, t_0) consists of a departure station A, an arrival station B, and a departure time t_0 (including the departure day). Connections are valid if they depart at least at the given departure time t_0, and the optimization criterion is to minimize the difference between the arrival time and the given departure time. We distinguish between two different variants of the problem: (a) The *simplified* version, where train transfers take negligible time and hence the input is restricted to $\mathit{transfer}(S) = 0$ for all stations S. (b) The *realistic* version where train transfers require arbitrary nonnegative minimum transfer times $\mathit{transfer}(S)$. We will discuss efficient solutions to these problems in Section 3.

Minimum Number of Transfers Problem (MNTP). A query consists only of a departure station A and an arrival station B. Trains are assumed to be operated daily, and there is no restriction on the number of days a timetable is valid[1]. All connections from A to B are valid, and the optimization criterion is to minimize the number of train transfers. We will discuss this problem in Section 4.

Bicriteria Problems. We consider also bicriteria or Pareto-optimal problems with the earliest arrival (EA) and the minimum number of transfers (MNT) as the two criteria. We are interested in two problem variants: (i) finding the so-called *Pareto-curve* which is the set of undominated Pareto-optimal paths (the set of feasible solutions where the attribute-vector of one solution is not dominated by the attribute-vector of another solution), and (ii) finding the lexicographically first Pareto-optimal solution (e.g., find among all connections that minimize EA the one with minimum number of transfers). We will discuss these problems in detail in Section 5.

[1]This assumption can be safely made since time is not minimized in the MNTP, and thus in a MNTP-optimal connection one can wait arbitrarily long at a station for some connection that is valid only on certain days.

3 Earliest Arrival Problem

In this section we consider the modeling of the EAP, in both the time-expanded and the time-dependent approach. In either approach we first briefly describe how to model the simplified version of the problem, where transfers between trains at a station take negligible time, and subsequently consider the realistic version of EAP, where transfer time between trains at a station is non-zero.

3.1 Time-Expanded Model

3.1.1 Simplified Version The time-expanded model [11] is based on the *time-expanded* digraph 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 edge length of an edge (u, v) is $t_v - t_u \pmod{1440}$, where t_u and t_v are the time values associated with u and v, respectively.

It is easy to see that the simplified version of EAP can be solved by computing a shortest path from the first departure node at the departure station with departure time later than or equal to the given start time. Since edge lengths are non-negative, one can use Dijkstra's algorithm and abort the main loop when a node at the destination station is reached.

3.1.2 Realistic Version In this case, we keep, for each station, an additional copy of all departure nodes in the station which we call *transfer nodes*; see Fig. 1. 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, called *transfer edge*, to the transfer node with time value greater than or equal to the time of the arrival node plus the minimum time needed to change trains at the given station. The edge lengths are defined as in the definition of the original model (see Section 3.1.1).

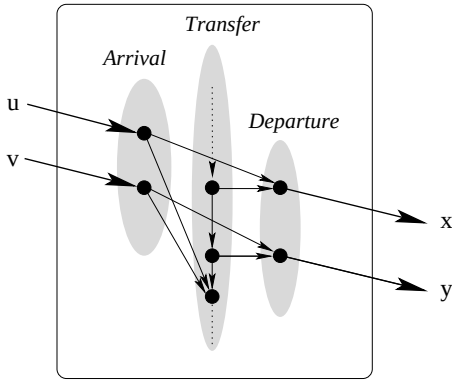

Figure 1: Modeling train transfers in the time-expanded approach.

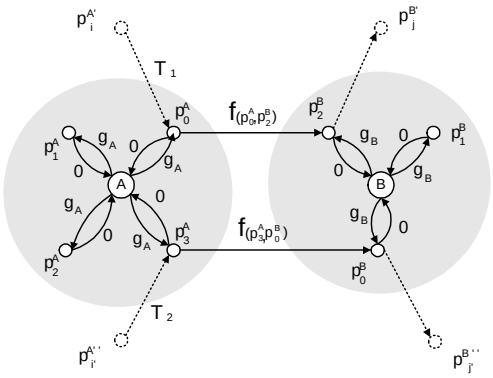

Figure 2: Modeling train transfers in the time-dependent approach.

3.2 Time-Dependent Model

3.2.1 Simplified Version The time-dependent model [1] 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 cost of an edge $e = (v, w)$ depends on the time at which this particular edge will be used by an algorithm which solves EAP. In other words, if T is a set denoting time, then the cost 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.

A modification of Dijkstra's algorithm can be used to solve the earliest arrival problem in the time-dependent model. 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 by evaluating the functions f_e on-the-fly. Assume that the edge $e = (A, B)$ is considered, and let the earliest arrival time at station A be t. We compute $f_e(t)$ by determining the earliest connection $c^* \in \mathcal{C}(e)$ departing from A later than t. Then, the earliest arrival at B via A is the arrival time of c^*. In other words, the length of e is the waiting time at A for c^* plus *length*(c^*). The particular connection c^* can be easily found by binary search if the elementary connections $\mathcal{C}(e)$ are maintained in a sorted array. See [1] for more details on the algorithm and its correctness.

3.2.2 Realistic Version To model non-zero train transfers in the time-dependent model, we use information on the routes that trains may follow as proposed in [10]. In the following, we describe the construction of a digraph $G = (V, E)$ which will be our main model and will be referred to as the *train-route digraph*; see Fig. 2.

We say that stations $A_0, A_1, ..., A_{k-1}$, $k > 0$, form a *train route* if there is some train starting its journey from A_0 and visiting consecutively $A_1, ..., A_{k-1}$ in turn. If there are several trains following the same schedule (with respect to the order in which they visit the above stations), then we say that they all belong to the same train route P.

In the train-route digraph there are several nodes per station A: one node A representing the station itself, and for each train route visiting A an additional node p_i^A. There are three kinds of edges: (i) *get-in edges* from A to p_i^A with constant length $g_A = transfer(A)$; (ii) *get-off edges* from p_i^A to A with zero edge length; and (iii) *route edges* from p_i^A to p_j^B (B is the next station in the train route) with time-dependent edge length. A route edge (p_i^A, p_j^B) contains those elementary connections from A to B that belong only to the considered train route. Get-in edges belonging to the departure station have zero edge length.

3.3 Incorporating Traffic Days Integrating traffic days in any of the so far described models and algorithms can be done as follows. Whenever an elementary connection is considered, the real departure time is known not only modulo a day, and the day can be determined by dividing the calculated departure time by 1440 (see Section 2 on page 2). A look-up in the traffic-day table of the corresponding train shows whether that day the elementary connection is valid or not. Elementary connections that are not valid can be ignored.

4 The Minimum Number of Transfers Problem

The graphs defined for the realistic version of the earliest arrival problem in both the time-expanded (Section 3.1.2) and the time-dependent (Section 3.2.2) approach can be used to solve the MNT problem with a similar method. Edges that model transfers are assigned a weight of one, and all the other edges are assigned weight zero. In both approaches a shortest path in the resulting (always static) weighted digraph yields a connection with minimum number of transfers. In the time-expanded case all transfer edges are the edges with weight one, and a shortest path from an arbitrary transfer node of the source station to an arrival node of the destination station yields a solution of the MNTP. In the time-dependent case the get-in edges except the get-in edges belonging to the departure station are assigned the weight one, all other edges have weight zero. Here, the MNTP is solved by a shortest path from the node representing the departure station to the one representing the arrival station.

5 Bicriteria Problems

We consider bicriteria problems with the earliest arrival (EA) and the minimum number of transfers (MNT) as the two criteria. We investigate two problem variants: on the one hand we want to find all Pareto-optimal solutions, and on the other hand we want to find the lexicographically first Pareto-optimal solution (e.g., find among all connections that minimize EA the one with minimum number of transfers). In the following, we shall refer to the bicriterion problem we consider as (X,Y) with X (resp. Y) as the first (resp. second) criterion we want to optimize and X,Y $\in$ {EA,MNT}. Again, the graphs defined for the realistic EAP described in Sections 3.1.2 and 3.2.2 are used.

5.1 Time-Expanded Model

5.1.1 Lexicographically First Pareto-optima We first consider the (EA,MNT) case. We maintain a second edge weight, the transfer value $trans(e)$ for an edge $e = (u, v)$, whose value is 1 if e is a transfer edge (i.e., u is an arrival node and v a transfer node), and 0 otherwise. Consider now the edge weights as pairs of travel time and $trans(e)$, and define the canonical addition on these pairs: $(a, b) + (a', b') = (a + a', b + b')$. The smaller relation is the lexicographical extension to pairs: $(a, b) < (a', b') \Leftrightarrow (a < a')$ or $(a = a'$ and $b < b')$. To find the lexicographically first Pareto-optimal solution, it then suffices to run Dijkstra's algorithm by maintaining distance labels as pairs of integers and by initializing the distance label of the start-node s to $d(s) = (0, 0)$. The optimal solution is found when a node at the destination station is considered for the first time during the execution of the algorithm. The (MNT,EA) is symmetric to the above and can be solved similarly. Note that in the same way the latest-departure problem can be solved by minimizing the difference between arrival time and actual departure time as second criterion.

5.1.2 All Pareto-optima Finding all Pareto-optimal solutions is generally a hard problem, since there can be an exponential number of them. However, if we make the (apparently reasonable) assumption that connections that arrive more than one day later than the earliest arriving connection are not of interest, then every node in the time-expanded graph can have only one Pareto-optimum. Hence, the above described method for producing the lexicographically first Pareto-optimum can provide all Pareto-optima of a station, if one simply lets the algorithm run until all nodes of the destination station have been considered (either settled or disregarded as dominated solutions).

5.2 Time-Dependent Model

5.2.1 Lexicographically First Pareto-optima The lexicographically first Pareto-optimum in the (MNT,EA) case can be solved also in the time-dependent model like in the time-expanded model (see Section 5.1.1), by defining edge weights as pairs of transfers and travel time (see also [10]). The (EA,MNT) case cannot be solved by that method, which can be easily shown by the construction of a counterexample (see Appendix B).

5.2.2 All Pareto-optima For generating all Pareto-optimal solutions in the time-dependent model we use as a sub-procedure the computation of an earliest arriving connection with bounded number of transfers (see Appendix). Then, the following approach can generate all Pareto-optimal solutions.

Solve EAP and count the number of transfers found, say M. Then, run the algorithm described in the Appendix for all values $M - 1, M - 2, \ldots, 0$. The algorithm given in the Appendix actually does this and in fact can speed up this process, since instead of stopping when the optimal solution with at most M transfers at the destination is found, we can just continue with the execution of the algorithm to produce the next EA solution with at most $M - 1$ transfers, and so on, until no new path can be found.

6 Experiments

The main goal of the experimental study is to compare the performance of the time-expanded and the time-

	Timetable	Nodes	Edges	C./ Node	C./ Edge
Expanded	France	166085	332170	1	0.5
	G-long	480173	960346	1	0.5
	G-local1	691541	1383082	1	0.5
	G-local2	1124824	2249648	1	0.5
	G-all	2295930	4591860	1	0.5
Dependent	France	4578	14791	36	11
	G-long	6817	18812	70	26
	G-local1	13460	37315	51	19
	G-local2	13073	36621	86	31
	G-all	32253	92507	71	25

Table 1: Parameters of the graphs considered in the comparison of the original models. The last two columns show the number of elementary connections per node and per edge.

dependent approach. Thus, given two different implementations and a timetable, we define the *relative performance* or *speed-up* with respect to a measured performance parameter as the ratio of the value obtained by the first implementation and the value obtained by the second one. When one time-expanded and one time-dependent implementation is compared, we always divide the time-expanded value by the time-dependent value, i.e., we consider the speed-up achieved when the time-dependent approach is used instead of the time-expanded approach.

All code is written in C++ and compiled with the GNU C++ compiler version 3.2; the experiments were run on a PC with AMD Athlon XP 1500+ processor at 1.3 GHz and 512MB of memory running Linux (kernel version 2.4.19). The implementation of the time-dependent model for the simplified earliest arrival problem uses the parameterized graph data structure of LEDA version 4.4.

6.1 Comparison of Original Models First, we consider the simplified version of the earliest arrival problem, since both approaches have been actually developed for that problem and we are interested in investigating their differences in exactly this setting.

6.1.1 Data The following five railway timetables were used. The first timetable contains French long-distance traffic (`France`) from the winter period 1996/97. The remaining four are German timetables from the winter period 2000/01; one resembles the long-distance traffic in Germany (`G-long`), two contain local traffic in Berlin/Brandenburg (`G-local1`) and in the Rhein/Main region (`G-local2`), and the last is the union of all the three German timetables (`G-all`). Hafas [2], the commercial timetable information system used by the German railway company Deutsche Bahn, is based on data in the same format. Table 1 shows the characteristics of the graphs used in these models for the above mentioned timetables.

Real-world queries were available only for the timetables `G-long` and `G-all`, so we additionally generated random queries for every timetable. Each set of queries consists of 50,000 queries of the form departure station, destination station and earliest departure time. In the tables, real queries are specially marked (✗).

6.1.2 Heuristics On top of the models described in Section 3, we considered heuristics to reduce the running time while optimal solutions are guaranteed. For both approaches we considered the goal-directed search heuristic (see, e.g., [11]). In this heuristic the length of every edge is modified in a way that if the edge points towards the destination its length gets smaller, while if the edge points away from the destination node, then its length gets larger. More precisely, for an edge (u, v) with length $l(u, v)$, its new length $l'(u, v)$ becomes $l'(u, v) = l(u, v) - p[u] + p[v]$, where $p[\cdot]$ is a potential function associated with the nodes of the graph. The crucial fact is that $p[\cdot]$ must be chosen in such a way so that $l'(u, v)$ is non-negative. For example, a valid potential of a node can be defined by dividing the euclidean distance to the destination by the highest speed of a train in the timetable (i.e., the time that the fastest train would need on the direct line towards the destination).

Concerning the time-expanded model we reduced the node set: All arrival nodes which have outdegree one can be removed, by redirecting incoming edges to the target node of the outgoing edge. Thus, in the time-expanded graphs, the number of nodes equals the number of elementary connections, and the number of edges is twice the number of nodes. In the time-dependent model the binary search technique to determine the edge length (see Section 3.2.1) can be replaced by the method described in [1] which avoids the binary searches.

6.1.3 Implementation Environment and Performance Parameters For the time-expanded model the implementation is based on that used in [11]; the optimization technique to ignore the arrival events described in Section 6.1.2 is included. For the time-dependent model, we have implemented both the plain version that uses binary search as well as the "avoid binary search" technique. For both models we also used the goal-directed search heuristic. Thus, for the time-expanded model we have two different implemen-

	Timetable	Time	El.C.	Nodes	Edges
Expanded	France	100.4	30824	33391	61649
	G-long	169.6	44334	48094	88668
	G-local1	608.7	176720	182717	353443
	G-local2	840.1	226027	232511	452056
	G-all	1352.8	326186	342917	652378
	G-long ✗	66.7	18891	20853	37783
	G-all ✗	392.1	96943	104369	193888
Dependent	France	8.2	8539	2269	4463
	G-long	10.7	20066	3396	5129
	G-local1	19.7	26792	6535	9835
	G-local2	20.7	31698	6524	10075
	G-all	76.6	79981	16145	26333
	G-long ✗	5.5	11173	1711	2682
	G-all ✗	37.3	40808	6926	11647

Table 2: Average CPU-time in ms and operation counts for solving a single query for the time-expanded (upper part) and the time-dependent model (lower part). The arrival nodes are omitted in the time-expanded model (see Section 6.1.2), and in the time-dependent model binary search was used. Goal-directed search was not applied in both cases. The marker (✗) indicates whether real-world or random queries have been used.

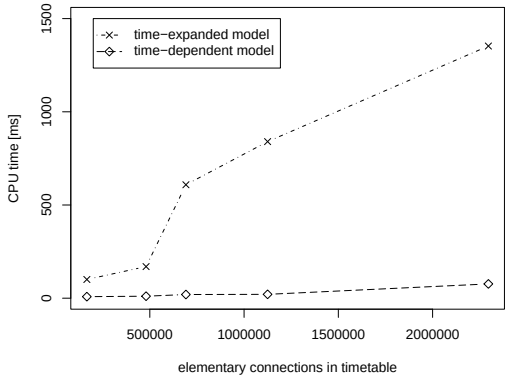

Figure 3: Comparison of the performance (CPU-time in ms) of the basic time-expanded and time-dependent implementations (see Table 2, only random queries are shown.) The abscissa shows the size of the timetable in number of elementary connections.

tations (goal-directed search with Euclidean distances or not), while for the time-dependent model we have several implementations depending on the use of: binary search or the "avoid binary search" version the goal-directed search heuristic with Euclidean or Manhattan distances and whether floating-point or integral potentials are used.

For each possible combination of timetable and implementation variant we performed the corresponding set of random queries (for `G-long` and `G-all` we additionally performed the corresponding real-world queries) and measured the following performance parameters as mean values over the set of performed queries: CPU-time in milliseconds, number of nodes, number of edges, and number of elementary connections touched by the algorithm. For the time-expanded model, the number of elementary connections touched is the number of train-edges touched by the algorithm, while for the time-dependent model it is the total number of elementary connections that have been used for calculating the edge lengths. More precisely, when binary search is used in the time-dependent model, for a single edge the number of steps needed by the binary search is the number of touched elementary connections.

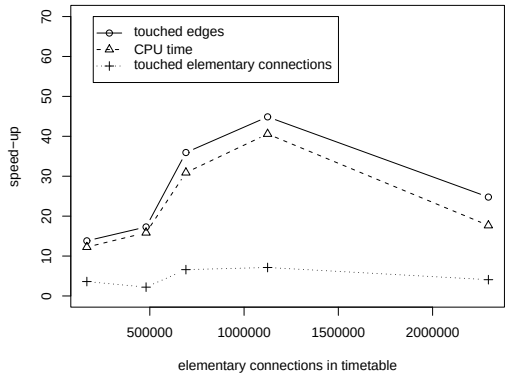

Figure 4: Like Figure 3, with the difference that here the ordinate shows not the runtime but the speed-up with respect to the number of touched edges, CPU-time, and number of touched elementary connections.

6.1.4 Results and Discussion Figures 3 and 4 as well as Tables 2, 3 and 4, clearly show that the time-dependent model solves the simplified earliest arrival

Time-Dependent Model, Binary Search

	Data	Time	El. conn.	Nodes	Edges
Goal Eucl. int	France	9.4	7072	1593	3415
	G-long	13.5	16597	2737	4217
	G-local1	28.6	26008	6257	9434
	G-local2	30.4	31196	6398	9895
	G-all	100.3	74525	14568	24030
	G-long ✗	6.3	8349	1238	1991
	G-all ✗	43.1	33676	5551	9420
Goal Eucl. float	France	9.4	7062	1590	3410
	G-long	13.6	16560	2730	4208
	G-local1	28.9	25975	6249	9422
	G-local2	30.7	31152	6389	9882
	G-all	103.6	74394	14538	23983
	G-long ✗	6.4	8318	1233	1984
	G-all ✗	44.5	33565	5532	9388
Goal Manh. int	France	7.9	7225	1647	3511
	G-long	11.2	16975	2807	4316
	G-local1	23.2	26086	6284	9473
	G-local2	24.7	31235	6407	9908
	G-all	86.4	74822	14639	24138
	G-long ✗	5.3	8555	1272	2041
	G-all ✗	38.0	33994	5615	9524
Goal Manh. float	France	7.7	7214	1644	3505
	G-long	11.1	16938	2800	4306
	G-local1	23.1	26053	6276	9461
	G-local2	24.6	31189	6398	9894
	G-all	88.9	74689	14608	24091
	G-long ✗	5.2	8524	1267	2034
	G-all ✗	39.0	33880	5594	9491

Table 3: Comparison of the time-dependent implementations that use binary search and four different versions of goal-directed search: Euclidean distance with (a) integer and (b) float potentials, and Manhattan distance with (c) integer and (d) float potentials. Columns are as in Table 2.

Time-Expanded Model

	Data	Time	El. conn.	Nodes	Edges
Goal Eucl. int	France	84.0	22259	24179	44517
	G-long	175.0	34259	37453	68517
	G-local1	684.3	170369	176243	340741
	G-local2	953.0	219992	226386	439986
	G-all	1392.6	285440	300788	570885
	G-long ✗	54.3	13384	14931	26768
	G-all ✗	341.9	74069	80229	148140

Time-Dependent Model, Avoid Binary Search

	Data	Time	El. conn.	Nodes	Edges
Plain Dijkstra	France	5.9	8942	2262	4386
	G-long	7.5	9216	3396	5129
	G-local1	14.2	18312	6541	9814
	G-local2	14.6	18435	6524	10075
	G-all	47.4	48520	16146	26333
	G-long ✗	3.8	4773	1711	2682
	G-all ✗	20.1	20993	6927	11648
Goal Eucl. int	France	6.6	6711	1614	3406
	G-long	9.2	7553	2737	4217
	G-local1	20.5	17656	6301	9499
	G-local2	21.5	18088	6398	9895
	G-all	63.0	44010	14568	24030
	G-long ✗	4.2	3527	1238	1991
	G-all ✗	23.7	16898	5552	9421
Goal Manh. int	France	5.1	6926	1669	3505
	G-long	7.1	7733	2807	4316
	G-local1	15.3	17621	6289	9480
	G-local2	16.1	18113	6407	9908
	G-all	50.5	44214	14639	24138
	G-long ✗	3.2	3618	1272	2041
	G-all ✗	19.1	17088	5615	9524

Table 4: Comparison of goal-directed search in the time-expanded case (upper part) and the technique to avoid binary searches in the time-dependent case (lower part). In the time-dependent case two different distance measures for the goal-directed search are reported, the Euclidean and the Manhattan distances with integral potentials, which were the fastest. Columns are as in Table 2.

Expanded	Nodes	Edges
Simplified	289432	578864
Realistic	578864	1131164

Dependent	Station Nodes	Route Nodes	Timetable Edges	Transfer Edges
Simplified	6685	–	17577	–
Realistic	6685	79784	72779	159568

Table 5: Graph parameters for the realistic models, applied to the same input timetable `G-long-1`: The number of nodes and edges of the graphs in the time-expanded (upper part) and time-dependent (lower part) approach, compared to the simplified, original models.

Problem	Time	Nodes	Edges
EA-simple ✗	70	20760	41519
EA ✗	78	40624	73104
MNT ✗	125	101731	138417
(EA,MNT) ✗	82	40628	73123
(MNT,EA) ✗	161	99061	137075
Pareto ✗	287	123943	236887
EA-simple	106	34469	61955
EA	122	61159	111301
MNT	212	169299	239841
(EA,MNT)	129	61195	111386
(MNT,EA)	259	163438	234297
Pareto	405	170946	330150

Table 6: Results for the realistic problems using the time-expanded implementations. For comparison, the columns referred by EA-simple show the results in the simple model using the `G-long-1` data set.

problem considerably faster than the time-expanded model, for every considered data set. Thus, the much smaller graph in the time-dependent approach pays off, and the edge lengths can be computed efficiently enough when real data is considered.

Regarding CPU-time, the speed-up ranges between 12 (`France`) and 40 (`G-local2`) when the basic implementations are used (see Fig. 4 and Table 2), and between 17 (`France`) and 57 (`G-local2`) when comparison concerns the best implementations (including heuristics) in both models (see Tables 3 and 4).

Concerning the time-dependent model, we observe that it is better to use the "avoid binary search" technique (see Tables 3 and 4). Compared to the binary search implementation the speed-up was between 1.39 (`G-local1`) and 1.86 (`G-all` with real-world queries). The goal-directed search technique always reduces the search space of Dijkstra's algorithm, i.e., the number of touched nodes and edges. However, this reduction payed off only in a few cases in the sense that it could not also decrease the CPU-time.

6.2 Comparison of Realistic Models We now turn to the comparison of the two approaches when the more realistic problems and models are considered. As input data we use a variant of the `G-long` timetable, which we get by using only elementary connections that are valid on the first day of the timetable period. We refer to that timetable as `G-long-1`. We applied the real-world and random queries described in Section 6.1.1. Table 5 shows the parameters of the graphs used in the realistic models compared to the original models.

6.2.1 Implementation Environment For both approaches we implemented the described solutions for the realistic earliest arrival problem (Section 3), the minimum number of transfers problem (Section 4), and the all-Pareto-optima problem involving both of the former problems (Section 5). Additionally, we have more efficient implementations in the time-expanded case for the lexicographically first pareto-optimum when the arrival time is the first criterion (EA,MNT), and in the time-dependent case when the number of transfers is the first criterion (MNT,EA). In the time-expanded implementations we reduced the node set using a similar method as in the simplified case (see 6.1.2), and omitted the departure nodes in the time-expanded graph. Also for the realistic time-dependent implementations we applied heuristics similar to the method that avoids the binary search; for details see [10].

6.2.2 Results and Discussion The average values of the number of touched nodes, edges, and the average running time for solving the real-world queries are displayed in Tables 6 and 7. These results show that, concerning CPU time, the time-dependent approach still performs better than the time-expanded approach in all the cases considered. However, the gap is not as big as for the simplified earliest arrival problem. In fact, for the realistic earliest arrival problem with realistic queries the speed-up is only 1.6 (compared to a speed-up of 12 for the simplified EAP and the data set `G-long`, see Table 2). In the time-expanded case, the graph used in the realistic EAP has less than twice as many nodes and edges as the graph used in the simplified EAP, and is of very similar structure. Thus, it needs only slightly more time to solve the realistic EAP than to solve the simplified EAP. The lexicographically first (EA,MNT) problem is solved in a very similar way as the realistic EAP, and the CPU-time and operation

Problem	Time [ms]	Nodes	TT-Edges	Trans. Edges
EA-simple ✗	10	2967	4365	–
EA ✗	50	44731	38168	45494
MNT ✗	38	26680	21558	61615
(MNT,EA) ✗	83	28272	22901	60462
Pareto ✗	181	78412	65753	79691
EA-simple	11	3315	4811	–
EA	54	48200	41011	48942
MNT	47	33455	27235	69411
(MNT,EA)	106	35262	28779	69054
Pareto	219	92378	77610	94904

Table 7: As Table 6, but for the time-dependent implementations.

counts are almost identical. In contrast, for the MNT and the lexicographically first (MNT,EA) problems as well as for finding all Pareto-optimal solutions a much bigger part of the graph has to be searched, and thus more CPU-time is needed.

In the time-dependent case, because of the additional nodes and edges in the train-route graph, which are much more than the nodes in the simplified time-dependent graph, the realistic earliest arrival problem is solved 5 times slower than the simplified EAP. The MNT is solved faster than the realistic EAP, since all edge lengths in the train-route digraph are static. The solution of the (MNT,EA) problem again involves time-dependent edge lengths, and thus is slower than computing the realistic EAP and MNTP. The implementation for all Pareto-optimal solutions uses the compution of the earliest arrival problem with bounded number of transfers as a sub-procedure, and needs only roughly twice the time as the solution to the lexicographically first (MNT,EA) problem.

7 Conclusion

We have discussed time-expanded and time-dependent models for several kinds of single and bicriteria problems in timetable information. In the time-expanded case, extensions that model more realistic requirements (like modeling train changes) could be integrated in a more-or-less straightforward way and the central characteristic of the approach is that a solution to a given optimization problem could be provided by solving a shortest path problem in a static graph, even for finding all Pareto-optimal solutions in the considered bicriterion problem. In the time-dependent case, the central characteristic of having one node per station had to be violated when more complex optimization problems (like the integration of minimum transfer times at stations) are considered, and more sophisticated techniques in the bicriterion case had to be used. Finding the lexicographically first connection when the earliest arrival is the main criterion could not be done directly in the time-dependent model. Nevertheless, all other problems under consideration could be modeled also in the time-dependent approach.

The experimental study showed that the time-dependent approach is clearly superior with respect to performance of the original models, as speed-up factors in the range from 10 to 40 were observed. Considering the extensions towards realistic models, however, the time-dependent approach still performs better, but the difference is much smaller. The time-expanded approach benefits from the straight-forward modeling that allows more direct extensions and simpler implementations.

References

[1] G.S. Brodal and R. Jacob. Time-dependent networks as models to achieve fast exact time-table queries. In *Proc. 3rd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2003)*, *Electronic Notes in Theoretical Computer Science*, volume 92, issue 1, Elsevier, 2003.

[2] `http://bahn.hafas.de`. Hafas is a trademark of Hacon Ingenieurgesellschaft mbH, Hannover, Germany.

[3] M. Schnee, M. Müller-Hannemann, and K. Weihe. Getting train timetables into the main storage. In *Proc. 3rd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2002)*, *Electronic Notes in Theoretical Computer Science*, volume 66, issue 6, Elsevier, 2002.

[4] R. Möhring. *Angewandte Mathematik - insbesondere Informatik*, pages 192–220. Vieweg, 1999.

[5] M. Müller-Hannemann and K. Weihe. Pareto shortest paths is often feasible in practice. In *Proc. 5th Workshop on Algorithm Engineering (WAE 2001)*, *Springer LNCS*, volume 2141, pages 185–198, Springer, 2001.

[6] K. Nachtigal. Time depending shortest-path problems with applications to railway networks. *European Journal of Operations Research*, volume 83, pages 154–166, 1995.

[7] A. Orda and R. Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM*, volume 37(3), 1990.

[8] A. Orda and R. Rom. Minimum weight paths in time-dependent networks. *Networks*, volume 21, 1991.

[9] S. Pallottino and M. Grazia Scutellà. *Equilibrium and advanced transportation modelling*, chapter 11. Kluwer Academic Publishers, 1998.

[10] E. Pyrga, F. Schulz, D. Wagner, and C. Zaroliagis. Towards realistic modeling of time-table information through the time-dependent approach. In *Proc. 3rd Workshop on Algorithmic Methods and Models for*

Optimization of Railways (ATMOS 2003), *Electronic Notes in Theoretical Computer Science*, volume 92, issue 1, Elsevier, 2003.

[11] F. Schulz, D. Wagner, and K .Weihe. Dijkstra's algorithm on-line: An empirical case study from public railroad transport. *ACM Journal of Experimental Algorithmics*, volume 5(12), 2000.

[12] F. Schulz, D. Wagner, and C. Zaroliagis. Using multi-level graphs for timetable information in railway systems. In *Proceedings 4th Workshop on Algorithm Engineering and Experiments (ALENEX 2002)*, *Springer LNCS*, volume 2409, pages 43–59, Springer, 2001.

[13] D. Wagner and T. Willhalm. Geometric speed-up techniques for finding shortest paths in large sparse graphs. In *Proc. 11th European Symposium on Algorithms (ESA 2003)*, *Springer LNCS*, volume 2832, pages 776–787, Springer, 2003.

[14] M. Ziegelmann. Constrained shortest paths and related problems. PhD Thesis, Naturwissenschaftlich-Technischen Fakultät der Universität des Saarlandes, 2001.

Appendix

A Earliest Arrival with Bounded Number of Transfers in the Time-Dependent Model

Given two stations a and b, and a positive integer k, the Earliest Arrival problem with Bounded number of Transfers (EABT) is defined to be the problem of finding a valid connection from a to b such that the arrival time at b is the earliest possible, and subject to the additional constraint that the total number of transfers performed in the path is not greater than k. Since EAP reduces to a shortest path problem, EABT is clearly a resource constrained shortest path problem.

We consider two algorithms for solving the EABT problem. The first one is an adaptation of the method proposed in [1] to our extended time-dependent model (train-route digraph). The second one is an adaptation of the labeling approach (see e.g., [14]) for solving resource constraint shortest paths to our extended time-dependent model.

The idea of [1] casted to the extended time-dependent model is as follows. Let A denote the get-off edges, D the get-in edges, and R the route edges (cf. Section 3.2.2). We construct a new digraph $G' = (V', E')$ consisting of $k+1$ levels. Each level contains a copy of the train-route digraph $G = (V, A \cup D \cup R)$. For node $u \in V$, we denote its i-th copy, placed at the i-th level, by u_i, $0 \leq i \leq k$. For each edge $(u, v) \in A \cup R$, we place in E' the edges (u_i, v_i), $\forall 0 \leq i \leq k$. For each edge $(u, v) \in D$, we place in E' the edges (u_i, v_{i+1}), $\forall 0 \leq i \leq k$. These edges, which connect consecutive levels, indicate transfers. With the above construction, it is easy to see that a path from some node s_0 (at the 0-th level) to a node t_l (at the l-th level) represents a path from $station(s)$ to $station(t)$ with l transfers. In other words, the EABT problem can be solved by performing a shortest path computation in G' aiming to find a shortest path from a node p_0^s at level 0, where $a = station(s)$, to the first possible u_i at level i, $0 \leq i \leq k$, where u is the node of the train-route graph such that $b = station(u)$.

The adaptation of the labeling approach to our train-route digraph is as follows. We use the modified Dijkstra's algorithm (cf. Section 3.2.2), where now we maintain $k+1$ (instead of one) labels, and which requires some additional operations to take place as nodes are extracted from the priority queue. Each label is of the form $(t_i, l_i)_u$, $0 \leq i \leq k$, representing the currently best time t_i to reach node u by performing exactly l_i transfers.

Let s be the node for which $a = station(s)$. The algorithm works as follows. Initially, we insert to the priority queue the label $(t, 0)_s$. The priority queue is ordered according to time, aiming at computing the earliest arrival path. When we extract a label $(t_l, l)_u$, we relax the outgoing edges of u considering that u is reached on time t_l and with l transfers. In addition, if $(t_{l'}, l')$ was the last label of u that has been extracted, then we delete from the priority queue all labels of the form $(t_m, m)_u$ for $l < m < l'$, setting $l' = k$ in the case where $(t_l, l)_u$ was the first of the labels of u to have been extracted. In this way, we discard the *dominated* – by $(t_l, l)_u$ – labels from the priority queue, since for all such $(t_m, m)_u$ it holds that $t_l \leq t_m$ (as $(t_l, l)_u$ was extracted before $(t_m, m)_u$) and $l < m$. Clearly, such labels are no longer useful as $(t_l, l)_u$ corresponds to an s-u path at least as fast as the one suggested by $(t_m, m)_u$, and with less transfers than the latter. Exactly for the same reasons, when we relax an edge $(u, v) \in E$ having found a new label $(t_{l_1}, l_1)_v$ for v, we will actually update the label of v only if there has been so far no label of v extracted from the priority queue, or if the last label of v that was extracted had a number of transfers greater than l_1.

Concerning now the complexity of the labeling algorithm, we need to see that for each node the total number of labels that is scanned in order to find those that are in the priority queue and can safely be deleted is $O(k)$, while the total number of deletions is $O(nk)$, where $n = |V|$. This is due to the fact that we only check the labels from the last known (by a *delete-min* operation) number of transfers, until the previous one. In this way, each label is checked at most once throughout the execution of the algorithm. Since each edge will be relaxed at most $k+1$ times, the total number of relaxations will be $O(mk)$, where $m = |E|$.

We can also see that the total number of labels that are in the priority queue is at most $O(nk)$. Because of this, the time for a *delete-min* or a *delete* operation is $O(\log(nk))$. This means that the total time needed for the algorithm is $O(mk + nk \cdot \log(nk) + nk \cdot \log(nk)) = O(nk \cdot \log(nk))$, which is the same as for the algorithm in [1].

B The (EA,MNT) Problem in the Time-Dependent Approach

The following describes an example showing that the (EA,MNT) problem cannot be solved directly in the time-dependent approach by using pairs as edge costs in the train-route digraph. Consider the train-route digraph shown in Figure 5, and a query to find an A-D-connection. Let $\mathcal{C}_1$ be an A-B-C-D-connection with one transfer at C, and $\mathcal{C}_2$ be an A-C-D-connection with no transfer. Both connections arrive at the same (optimal) time at D, and connection $\mathcal{C}_1$ arrives earlier at C than the connection $\mathcal{C}_2$. Then, the algorithm for finding the lexicographically first solution using pairs (EA,MNT) as edge costs outputs connection $\mathcal{C}_1$ as optimal, while there is connection $\mathcal{C}_2$ with the same arrival time, but less transfers. The reason is that the time-dependent function for edge e is decreasing.

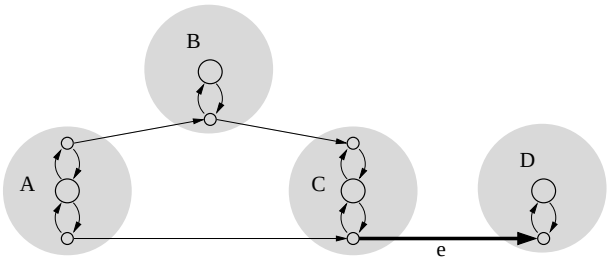

Figure 5: Lexicographically first (EA,MNT) connections cannot be found by simply using pairs as edge costs in the train-route digraph.