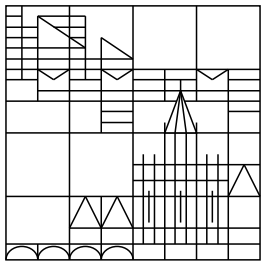

Universität Konstanz

Two Approaches for Time-Table Information: A Comparison of Models and Performance

Evangelia Pyrga
Frank Schulz
Dorothea Wagner
Christos Zaroliagis

Konstanzer Schriften in Mathematik und Informatik
Nr. 190, Juni 2003

ISSN 1430–3558

Universität Konstanz
Fach D 188, 78457 Konstanz, Germany
Email: preprints@informatik.uni–konstanz.de
WWW: http://www.informatik.uni–konstanz.de/Schriften/

Two Approaches for Time-Table Information: A Comparison of Models and Performance*

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 modelled as a node in the graph, while in the *time-dependent* approach the graph contains only one node per station. Train connections without intermediate stops correspond to edges. There is one edge for each single connection in the time-expanded model; in contrast, a couple of trains belong to the same edge in the time-dependent model. Both approaches have been recently considered for the earliest arrival problem. In this paper, we compare, on the one hand, the approaches with respect to more realistic modelling of real-world requirements. On the other hand, we evaluate their performance in an experimental study using real-world data. The time-expanded approach turns out to be more robust for modelling more complex scenarios, whereas the time-dependent approach shows a clearly better performance. As a conclusion the combination of both approaches seems promising.

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 modelling (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 modelling timetable information: the *time-expanded* [7, 11, 12, 13], and the *time-dependent* approach [1, 8, 9, 10]. 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 [12] 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

*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.

‡Department of Computer and Information Science, University of Konstanz, Box D188, 78457 Konstanz, Germany. Email: Frank.Schulz@uni-konstanz.de.

§University of Karlsruhe, Department of Computer Science, 76128 Karlsruhe, Germany.

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 changes problems. In the *earliest arrival* problem, a query consists of a departure and an arrival station, and a departure time (including the departure day). Connections are valid if they depart at least at the given departure time, and the goal is to find the valid connection that minimises the difference between the arrival time and the given departure time. There are two variants of the problem depending on whether train changes within a station are assumed to take negligible time (*simplified* version) or not. In the *minimum number of changes* problem, a query consists only of a departure station A and an arrival station B. Trains are assumed to operate daily (and there is no restriction on the number of days a timetable is valid). All connections from A to B are valid, and the goal is to find the valid connection that minimises the number of train changes when considering an itinerary from A to B. We consider also combinations of the above problems which can be seen as bicriteria or pareto-optimal problems.

Techniques for solving general multi-criteria problems have been discussed in [6, 7], where the discussion in [6] is focused on a distributed approach for timetable information problems. Space consumption aspects of modelling more complex real-world scenarios is considered in [5]. For the time-expanded model, the simplified version of the earliest arrival problem has been extensively studied [12, 13]. In [1] it is argued (theoretically) 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 compare the time-expanded and the time-dependent approaches with respect to modelling aspects and performance, and with respect to the specific, query-intensive scenario mentioned earlier. In order to cope with more realistic requirements we investigate train changes in combination with the earliest arrival problem and present new extensions of both approaches. In particular, the proposed extensions can handle cases not tackled by previous studies for the sake of simplification. These include cases where: (a) change of trains within a station does not take negligible time; (b) the minimum number of changes problem is considered; (c) traffic days are involved; and (d) a combination of the earliest arrival and minimum number of changes is considered. To compare the performance of the two models in practice, we conducted an extensive experimental study using real-world data for the simplified version of the earliest arrival problem, since both models have been actually developed for that problem (see [1, 12]).

In Section 2, after reviewing the basic modelling ideas of the time-expanded and time-dependent models, we present the new extensions of these approaches. Section 3 shortly discusses speed-up techniques. The experimental comparison of the two approaches for the simplified version of the earliest arrival problem based on real data from the German railways is presented in Section 4. We first consider how the plain versions of the two approaches compare, and subsequently the speed-up techniques. Section 5 summarises our insights on the advantages and disadvantages of the approaches under comparison.

2 Models and Problems

In this section, we provide definitions of the timetable problems that we will consider, as well as several models for their solution. All definitions and models refer to timetable information in a railway system, but the modelling and the algorithms can be applied to any other public transportation system, provided that its timetable satisfies the same characteristics.

A *timetable* consists of data concerning: *stations* (or bus stops, ports, etc.), *trains* (or busses, ferries, etc.) connecting stations, *departure* and *arrival times* of trains at stations, and *traffic days*. More formally, we are given a set of *trains* $\mathcal{T}$, a set of stations $\mathcal{S}$, and a set of *elementary connections* $\mathcal{C}$ whose elements c are 5-tuples of the form $c = (T, S_1, S_2, t_1, t_2)$. Such a tuple is interpreted as train T leaves station S_1 at time t_1, and the *immediately next* stop of train T is station S_2 at time t_2. If x denotes a tuple's field, then the notation $x(c)$ specifies the value of x in the elementary connection c. A consistency condition for trains must be fulfilled: all elementary connections belonging to a train T form a sequence, i.e., the arrival station of one elementary connection is the departure station of the following elementary connection.

The *arrival* and *departure times* t_1 and t_2 of the elementary connections within a day are integers in the interval $[0, 1439]$; the granularity of the timetable is one minute, and time values are minutes after midnight. Given two time values t_1 and t_2, the *cycle-difference* of t_1 and t_2 is the smallest nonnegative integer l such that $l \equiv t_2 - t_1 \pmod{1440}$. The *length* of an elementary connection is the cycle-difference of t_1 and t_2. A timetable is valid for N days, and every train is assigned a bit-field of N bits determining on which day the train is operated (for overnight trains the departure of the first elementary connection counts).

At a station $S \in \mathcal{S}$ it is possible to change from one train to another. Such a change 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, *minimum 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] $(1 \leq i \leq k)$ is called a *connection* from station $A = S_1(c_1)$ to station $B = S_2(c_k)$ if

$$
\begin{aligned}
c_i & \quad \text{is valid on day } \lfloor dep_i(P)/1440 \rfloor \\
S_2(c_i) &= S_1(c_{i+1}) \\
dep_i(P) &\equiv t_1(c_i) \pmod{1440} \\
arr_i(P) - dep_i(P) &= length(c_i) \\
dep_{i+1}(P) - arr_i(P) &\geq \begin{cases} 0 & \text{if } c_{i+1} \text{ follows } c_i \text{ in } T(c_i) \\ transfer(S_2(c_i)) & \text{otherwise} \end{cases}
\end{aligned}
$$

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 optimisation criterion on that set of connections. The problem is to find the optimal connection (or a set of optimal connections).

In this work, we are concerned with two main problems, namely the earliest arrival and the minimum number of changes problem.

Earliest arrival problem. A query consists of a departure and an arrival station, and a departure time (including the departure day). Connections are valid if they depart at least at the given

[1] We assume that the times $dep_i(P)$ and $arr_i(P)$ include data regarding also the departure/arrival day (by counting time in minutes from the first day of the timetable); i.e., 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 $\lfloor t/1440 \rfloor$.

departure time, and the optimisation criterion is to minimise 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 changes take negligible time; the input is restricted to $transfer(S) = 0$ for all stations S. (b) Train changes require arbitrary nonnegative minimum transfer times $transfer(S)$. A special case of the earliest arrival problem is the *latest departure* problem. Here, among all connections with earliest arrival time at the arrival station the actual departure time at the departure station has to be maximised.

Minimum number of changes problem. A query consists only of a departure station A and an arrival station B. Trains are assumed to operate daily (and there is no restriction on the number of days a timetable is valid). All connections from A to B are valid, and the optimisation criterion is to minimise the number of train changes. Let $P = (c_1, \ldots, c_k)$ be a connection from A to B. Then, $trans_i(P) = 1$, if c_{i+1} belongs to a different train than c_i, and $trans_i(P) = 0$, otherwise $(1 \leq i < k)$. Hence, the objective is to minimise, among all P, $\sum_{i=1}^{k-1} trans_i(P)$.

Combination of problems. We consider also combinations of the above problems (which can be seen as bicriteria or pareto-optimal problems). An interesting special case is to find the connection with minimum number of train changes among all connections that solve the earliest arrival problem.

2.1 Earliest Arrival Problem

We start presenting the models for the earliest arrival problem under the assumptions that changing trains within the same train station takes negligible time and that every train is operated daily; later we show how these assumptions can be waved.

2.1.1 Time-Expanded Model

This model [12] is based on the *time-expanded* digraph which is constructed as follows. For every elementary connection (T, S_1, S_2, t_1, t_2) in the timetable, there is one *train-edge* in the graph. These train-edges induce the set of nodes of the graph: each (directed) train-edge connects a *departure node* with an *arrival node*. We say that the departure node belongs to station S_1 and the arrival node to station S_2. Each of these nodes is assigned the time value t_1 and t_2, respectively. 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, v_i and v_{i+1} are connected by *stay-edges* $(1 \leq i \leq k-1)$, as well as v_k is connected to v_1. The edge length of an edge (u, v) is the cycle-difference of t_v and t_u, where t_u and t_v are the time values associated with u and v, respectively. Fig. 1 illustrates this definition.

The earliest arrival problem can be solved by simply 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. Using Dijkstra's algorithm, the main loop can be aborted when a node at the destination station is reached.

2.1.2 Time-dependent Model

The second model [1] is also based on a digraph, called *time-dependent* graph. In this graph there is only one node per station, in contrast to the previous model, where every event at a station corresponds to a node. There is an edge e from station A to station B if there is an elementary

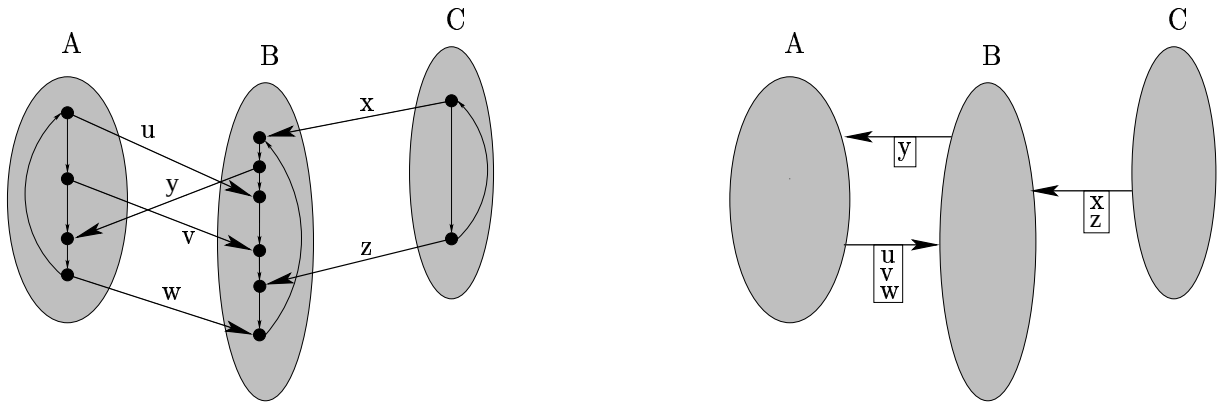

Figure 1: Consider a timetable with three stations A, B, and C. Three different trains are scheduled directly from A to B (that yields three elementary connections u, v, and w, say). Further there is one train from C via B (elementary connection x) to A (y), and one train from C to B (z). The time-expanded graph is drawn on the left (the time is expanded on the y-axis) and the time-dependent graph on the right.

connection from A to B. The set of elementary connections from A to B are denoted by $\mathcal{C}(e)$. The definition is illustrated in Fig. 1.

Let D denote the departure station and t_0 the earliest departure time. A modification of Dijkstra's algorithm can be used to solve the earliest arrival problem in this model [1]. The differences (w.r.t. Dijkstra's algorithm) are: not to set the distance of the start-node to 0, and to calculate the edge lengths on-the-fly. The algorithm is as follows. Initially, the distance label $d(D)$ of the departure station D is set to t_0. The edge lengths are calculated as follows. Since Dijkstra's algorithm is a label-setting shortest-path algorithm, whenever an edge $e = (A, B)$ is considered the distance label of the node A is the distance from the source node to A. Here, the distance label denotes the earliest arrival time at station A. In other words, we indeed know the earliest arrival time at station A whenever the edge $e = (A, B)$ is considered, and therefore we know at that stage of the algorithm which train has to be taken to reach station B via A as early as possible: the first train that departs later than or equal to the earliest arrival time at A[2]. Hence, we have to determine the connection $c \in \mathcal{C}(e)$ that minimises $\{\textit{cycle-difference}(d(A), t_1(c)) \mid c \in \mathcal{C}(e)\}$. This can be easily done using binary search if the elementary connections $\mathcal{C}(e)$ are maintained in a sorted array. The edge length of e is the *cycle-difference* of $d(A)$ and $t_1(c)$ plus the length of c.

2.1.3 Modelling Train Changes with Non-zero Transfer Time

We now discuss how the previous models can be extended to handle the case where changing of trains does not take negligible time, i.e., there are non-zero minimum transfer times at stations.

Time-expanded Model. In this case, we keep, for each station, a copy of all departure and arrival nodes in the station which we call *change-nodes*; see Fig. 2. The stay-edges are now introduced between the change-nodes. For every arrival node there are two additional outgoing edges: one edge to the departure of the same train, and a second edge to the change-node with time value

[2]We assume here that overtaking of trains on an edge is not allowed and that changing trains takes negligible time.

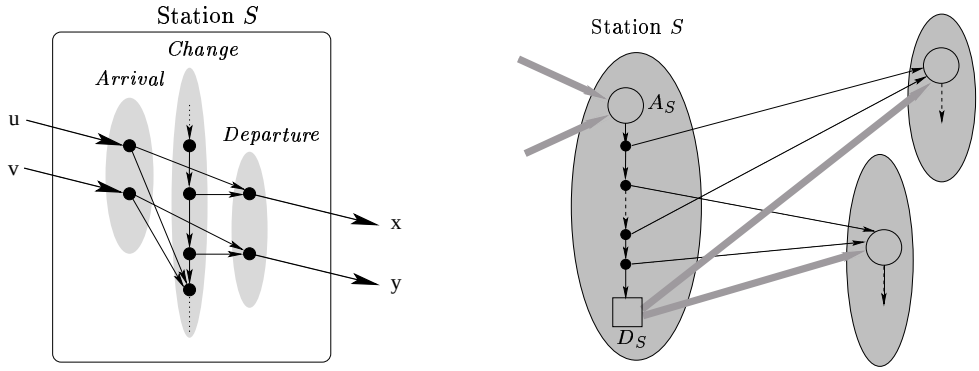

Figure 2: Modelling train changes in the time-expanded model (left) and the time-dependent model (right) for a sample station S. In the time-expanded case there are three types of nodes: arrival, change and departure nodes. In the time-dependent case the edges from the original graph remain (thick and grey); additionally, each station is partially time-expanded, i.e., replaced by a path from an arrival node A_S (big circle) to a departure node D_S (big rectangle).

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 basic definition of the model. Later, in Section 2.3, this model is also used to solve the minimum number of changes problem in combination with the earliest arrival problem. In [7] a similar method is proposed to solve this combination of problems, but in that paper the minimum transfer times are not taken into account.

Time-dependent Model. Train changes can be included in the time-dependent model by partial time-expansion; see Fig. 2. The graph is dynamically constructed during the algorithm. For each station S there is an arrival node A_S and a departure node D_S. Every time-dependent edge (S_1, S_2) in the previous time-dependent graph is transformed to a time-dependent edge (D_{S_1}, A_{S_2}). When an arrival node A_S is considered and labelled permanent in Dijkstra's main loop, A_S and D_S are dynamically connected by a directed A_S–D_S-path as follows. At that point of the algorithm, the earliest arrival time $arr(S)$ at S is known by the shortest path to A_S; we associate the arrival node with $arr(S)$ and the departure node D_S with $arr(S) + transfer(S)$. For each elementary connection c departing S between A_S and D_S there is an inner node v_c in the path, and the edges on that path are the *stay-edges* from the time-expanded model, with fixed edge lengths. For each inner node v_c there is a *train-edge* to the arrival node of the destination station of c, also with fixed edge length, namely $length(c)$.

The algorithm is modified as follows. For every elementary connection a flag is maintained that initially is set to false. If the flag is true we say that the elementary connection is *valid*. Initially, the departure node D_S of the departure station S is inserted in the priority queue, the earliest arrival time at S is set to the departure time of the query, and $transfer(S) = 0$. Further, the treatment of inner nodes is modified. Consider an elementary connection c, and let $(\ldots, c', c, c'', \ldots)$ be the sequence of elementary connections on the train to which c belongs. When an inner node v_c is permanently labelled, the following is done in the algorithm: (1) relax the stay-edge outgoing of v_c as usual; (2) check whether it is true that *either c is valid or c' departs later than the earliest arrival time plus transfer time at the previous station*; if that condition holds, then (2a) the train-edge outgoing of v_c is relaxed and (2b) the elementary connection c'' is set to be valid.

2.1.4 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[1] (not only modulo a day); thus, the day can be determined by dividing the calculated departure time by 1440. Then 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 simply be ignored.

2.2 Minimum Number of Changes Problem

Recall that in this problem, time is not involved at all. Hence, its modelling is independent of both the time-expanded and the time-dependent models.

Let A be the departure station. We show how to compute for every other station B the minimum number of changes needed to reach B. The algorithm consists of steps $0, \ldots, I$, where I is the highest minimum number of changes over all stations. In the i-th step all stations that have minimum number of changes i are found and marked with that label. For step 0, all trains outgoing from A are considered and each station that succeeds A in one of these trains is labelled with 0. Then, in step i ($1 \leq i \leq I$), all outgoing trains of stations labelled with $i-1$ are considered and succeeding stations within these trains are labelled with i only if they are not yet labelled.

This simple method can be optimised: for example, each set of completely parallel trains (i.e., the sequence of stations is the same) can be replaced by one representative train.

2.3 Combination of Problems

We consider bicriteria problems where the first criterion is to solve the earliest arrival problem. First, we consider the minimisation of train changes as second criterion, and show how to find among all connections solving the earliest arrival problem a connection minimising the number of train changes. Under certain circumstances, the approach can be generalised to find all pareto-optimal solutions.

For the time-expanded case, we assume that the time-expanded model is extended to cope with changing trains as described in Section 2.1.3. We maintain a second edge weight, the transfer value $trans(e)$ for an edge $e = (u, v)$, whose value is 1 if u is an arrival-node and v a change-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 $b < b')$. In Dijkstra's algorithm, maintain also distance labels as pairs of integers, and initialise 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.

Note that in the same way the latest-departure problem can be solved by minimising the difference between arrival time and actual departure time as second criterion. Further note that if waiting a whole day at a station never pays off (e.g., if every train is assumed to operate daily), then the following generalisation can be solved optimally and efficiently by the above method: given a departure station and time, find for all other stations the pareto-optimal connections minimising arrival time and the second criterion. For a station several pareto-optima can exist which makes the problem difficult in general. However, if we consider the problem in the time-expanded model, due to the fact that waiting for more than a day doesn't pay off every time-expanded node has only

one pareto-optimal solution, and the problem can be reduced to a single-criterion shortest-path problem in the described way.

In the pure time-dependent model, every station corresponds to one node, and without expanding the time-dependent graph the above idea cannot be applied.

3 Heuristics for Speeding-up Query Time

Previous experimental studies with the time-expanded model [12, 13, 14] have shown that heuristic improvements may considerably speed-up performance. These techniques allow for generation of additional data during preprocessing that can be used in the on-line phase to speed-up the algorithm. Although these techniques have been applied to the time-expanded model, they can be suitably modified to be applied to the time-dependent case as well.

In this section, we focus on the goal-directed search technique that can be applied to both models, and we describe two model-specific speed-up heuristics. All these techniques are then considered in the experimental study.

3.1 Goal-Directed search

The most natural heuristic to consider is the goal-directed search or the method of potentials (see e.g., [4]), in order to exploit the geometric information associated with the nodes (coordinates of stations). 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 $wt(u, v)$, its new length $wt'(u, v)$ becomes $wt'(u, v) = wt(u, v) - p[u] + p[v]$, 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 $wt'(u, v)$ is non-negative.

This heuristic was considered in [12] for the time-expanded model: if $d(u, t)$ denotes the Euclidean distance of a node u to the destination station t of the query and v_{max} is the maximum speed of the timetable[3], the potential function in the time-expanded model is defined as $p[u] = d(u, t)/v_{max}$. This scaling of Euclidean distances is necessary to guarantee valid (i.e., non-negative) potentials.

We applied the goal-directed search in the time-dependent model using Euclidean or Manhattan distances between stations. A slightly different scaling factor was used in this case to guarantee valid potentials (details are provided in the Appendix). Since the use of distances forces us to use floating point numbers in the priority queue associated with the shortest path algorithm and this may incur an additional time overhead, we also considered as potentials the integral parts of the floating-point potentials (see Appendix for the details). The latter turned out to be rather beneficial in certain cases.

3.2 Omit Arrival Nodes in the Time-Expanded Model

There is a simple way to optimise the model when only the earliest arrival problem should be solved. Except for the arrival station, we need only to consider nodes that correspond to departures of trains. All the other arrival nodes can be ignored, and ingoing edges to arrival nodes are redirected

[3]The speed of an elementary connection is the Euclidean distance of the two stations involved divided by the travel time. The maximum speed v_{max} of the timetable is the maximum over all elementary connections' speed.

to the next node of the same station that corresponds to a departure. This optimisation yields a graph of roughly half the original size.

3.3 Avoid Binary Search in the Time-Dependent Model

In the time-dependent model, we also considered the heuristic that avoids the binary search as described in [1]. The idea is as follows. Let k be the out-degree of a node v in the time-dependent digraph and let $(v, u_1), \ldots, (v, u_k)$ be its outgoing edges. Construct a table D_v by sorting all events of v's outgoing edges w.r.t. their departure time. Place the first k such events in the first k entries (primary segment) of D_v, leave the next k entries empty (secondary segment), place the next k events in the next k entries of D_v, and so on. Let t_0 be the last event in D_v before some secondary segment. For every (v, u_i), $1 \leq i \leq k$, find the first event with departure time $t \geq t_0$ and put it into the i-th entry of the next secondary segment. For an edge (w, v), let t_1 be the arrival time of a primary event $P^w \in D_w$ (event belonging to some primary segment of D_w). Create a pointer from P^w to the immediately next primary event $P^v \in D_v$ with timestamp $t_2 \geq t_1$. The above construction avoids binary search, because when node v is extracted from the priority queue, we simply follow the pointer of the event P^w that caused v's extraction from the priority queue, and which leads to a primary entry $P^v \in D_v$. Hence, to find the next outgoing event of node v it suffices to scan the rest of the primary segment containing P^v and its next secondary segment.

4 Experiments

For the comparison of the performance of the two approaches we consider the simplified version of the earliest arrival problem (i.e., changing trains takes negligible time and every train is assumed to be operated daily), since both models have been actually developed for that problem and we are interested in investigating their differences in exactly this setting. For both models we also investigate the heuristics described in Section 3.

4.1 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 (`ger`) timetables from the winter period 2000/01; one resembles the long-distance traffic in Germany (`longdist`), two contain local traffic in Berlin/Brandenburg (`local1`) and in the Rhein/Main region (`local2`), and the last is the union of all the three German timetables (`all`). Hafas [3], the timetable information system used by the German railway company Die 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, and Fig. 4 (in the Appendix) shows how the elementary connections are distributed on the edges in the time-dependent model.

Real-world queries were available only for the timetables `ger-longdist` and `ger-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.

4.2 Implementation Environment and Performance Parameters

For the time-expanded model the implementation is based on that used in [12]; the optimisation technique to ignore the arrival events described in Section 3.2 is included. For the time-dependent model, we have implemented both the plain version that uses binary search as well as the "avoid

	Time-Expanded		Time-Dependent			
Timetable	Nodes	Edges	Nodes	Edges	El. conn. per node	El. conn. per edge
`france`	166085	332170	4578	14791	36	11
`ger-longdist`	480173	960346	6817	18812	70	26
`ger-local1`	691541	1383082	13460	37315	51	19
`ger-local2`	1124824	2249648	13073	36621	86	31
`ger-all`	2295930	4591860	32253	92507	71	25

Table 1: Parameters of the considered graphs for each of the timetables. The two columns on the left show the size of the graph used in the time-expanded model with the optimisation described in Section 3.2. The number of nodes equals the total number of elementary connections in the timetable. The remaining columns show the parameters of the graph used in the time-dependent model.

binary search" technique. The code regarding the variant of the model that uses binary search is written using the LEDA parameterised graph type. On the other hand, the avoid binary search technique does not need to make use of an explicit graph type and thus is written without using LEDA graphs. We implemented a new priority queue based on heaps that bears a great resemblance to the lazy variant of pairing heaps [2]. For both models we also used the goal-directed search heuristic. Special care has also been taken to avoid the initialisation steps at the beginning of each query when the new priority queue was used. Thus, for the time-expanded model we have two different implementations, while for the time-dependent model we have several implementations depending on the use of: binary search or the "avoid binary search" version, the priority queue used (LEDA or our own), the goal-directed search heuristic with Euclidean or Manhattan distances and whether floating-point or integral potentials are used.

The code is written in C++ and compiled with the GNU C++ compiler version 2.95.3; 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 2.4.10).

For each possible combination of timetable and implementation variant we performed the corresponding set of random queries (for `ger-longdist` and `ger-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.

Given one time-expanded, one time-dependent implementation and a timetable, we define the *speed-up* with respect to one of the above parameters as the ratio of the value obtained by the time-expanded implementation and the one obtained by the time-dependent one.

4.3 Results and Discussion

Fig. 3 and Table 2, as well as Tables 3 and 4 in the Appendix, clearly show that the time-dependent model solves the simplified earliest arrival problem considerably faster than the time-expanded

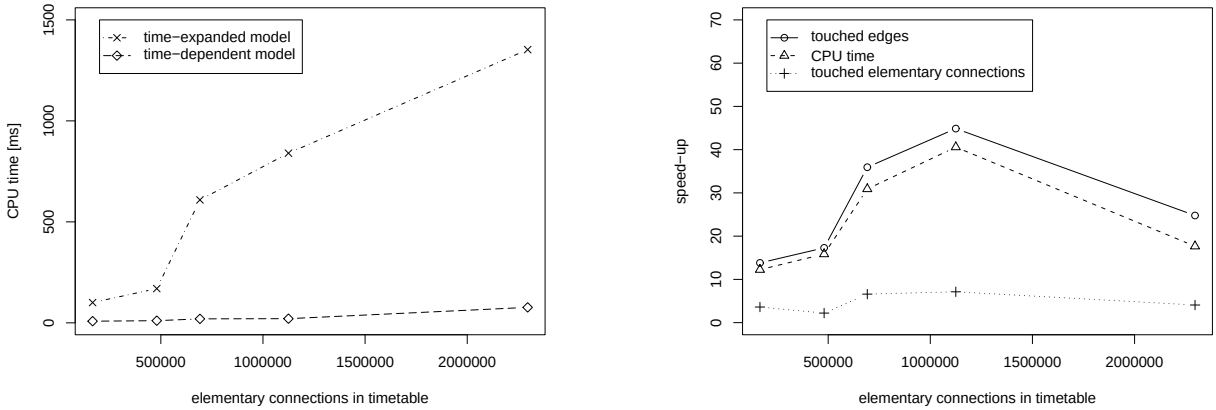

Figure 3: Performance of the basic implementations of the time-expanded and time-dependent models for the simplified earliest arrival problem (no goal-directed search, binary search in the time-dependent model) regarding the five timetables and the random queries. Each point represents the average over these queries of one measured performance parameter. On the abscissa the size of the timetable in number of elementary connections is shown. On the left, the average CPU-time for answering a query in the two models is presented. On the right, the speed-up with respect to the number of touched edges, CPU-time, and number of touched elementary connections is shown.

model, for every considered data set. Regarding CPU-time, the speed-up ranges between 12 (`france`) and 40 (`ger-local2`) when the basic implementations are used (see Fig. 3 and Table 2), and between 17 (`france`) and 57 (`ger-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 is 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 (`ger-local1`) and 1.86 (`ger-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. In most cases the CPU-time was increased due to the additional computations required to calculate the edge lengths, and this is the main reason why this technique appears slower in the results. Another reason is that in the timetables used in our experiments the maximum speed over all elementary connections is high. A high maximum speed yields small potential functions[4], and thus bad performance of the goal-directed search technique.

5 Comparison and Conclusions

We have discussed time-expanded and time-dependent models for several kinds of problems in timetable information. In the time-expanded case, extensions that model more realistic requirements (like modelling 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 optimisation problem could

[4]In both models the potentials are inversely proportional to the maximum speed; for the time-expanded case this is clear by definition, and for the time-dependent case see the definition of the factor λ_t in the Appendix.

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

Table 2: Average CPU-time 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 3.2), and in the time-dependent model binary search was used. Goal-directed search was not applied in both cases. The column *Real* indicates whether real-world or random queries have been used.

be provided by solving a shortest path problem in a static graph. In the time-dependent case, the central characteristic of having one node per station had to be violated when more complex optimisation problems (like the integration of minimum transfer times at stations) are considered. Modelling of other scenarios by expansion of the time-dependent graph is also discussed in [1], where additional data like information about platforms at stations is required.

The outcome of the experimental study clearly verified the arguments in [1] concerning the simplified version of the earliest arrival problem: the time-dependent model performs significantly better than the time-expanded model. Not only the CPU-time, but also the number of elementary connections that are considered during the algorithm are heavily reduced by using the time-dependent model.

In conclusion, there are methods – like the modelling of bicriteria problems as single-criterion shortest path problems – in the time-expanded model that cannot be applied in the time-dependent model. On the other hand, we showed that the simplified earliest arrival problem can be solved considerably faster using the time-dependent model instead of the time-expanded model. This fact along with the successful modelling of minimum transfer times in the time-dependent model using partial time-expansion of nodes, suggest that a combination of both approaches is rather promising.

References

[1] Gerth Stølting Brodal and Riko Jacob. Time-dependent networks as models to achieve fast exact time-table queries. Technical Report ALCOMFT-TR-01-176, ALCOM-FT, September 2001.

[2] Michael Fredman, Robert Sedgewick, Daniel Sleator, and Robert Tarjan. The pairing heap: A new form of self-adjusting heap. *Algorithmica*, 1:111–129, 1986.

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

[4] Thomas Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Wiley, 1990.

[5] Mathias Schnee Matthias Müller-Hannemann and Karsten Weihe. Getting train timetables into the main storage. In Dorothea Wagner, editor, *Electronic Notes in Theoretical Computer Science*, volume 66. Elsevier Science Publishers, 2002.

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

[7] Matthias Müller-Hannemann and Karsten Weihe. Pareto shortest paths is often feasible in practice. In *Proceedings 5th Workshop on Algorithm Engineering*, volume 2141 of *Springer LNCS*, pages 185–198, 2001.

[8] Karl Nachtigal. Time depending shortest-path problems with applications to railway networks. *European Journal of Operations Research*, 83:154–166, 1995.

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

[10] Ariel Orda and Raphael Rom. Minimum weight paths in time-dependent networks. *Networks*, 21, 1991.

[11] Stefano Pallottino and Maria Grazia Scutellà. *Equilibrium and Advanced Transportation Modelling*, chapter 11. Kluwer Academic Publishers, 1998.

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

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

[14] Dorothea Wagner and Thomas Willhalm. Geometric speed-up techniques for finding shortest paths in large sparse graphs. Technical Report 183, Preprints in Mathematics and Computer Science at University of Konstanz, 2003.

A Appendix

A.1 Euclidean and Manhattan Distances as Potentials in the Time-dependent Model

For an edge $(u, v) \in E$ with cost function $wt(u, v, \tau)$, where τ is the arrival time at u, the new cost function $wt'(u, v, \tau)$, is defined as $wt'(u, v, \tau) = wt(u, v, \tau) - p[u] + p[v]$, where $p[\cdot]$ is the potential function.

Let t be the destination node. Then, define

$$p[u] = d(u, t)\lambda_t, \quad u \in V, \quad \lambda_t \geq 0$$

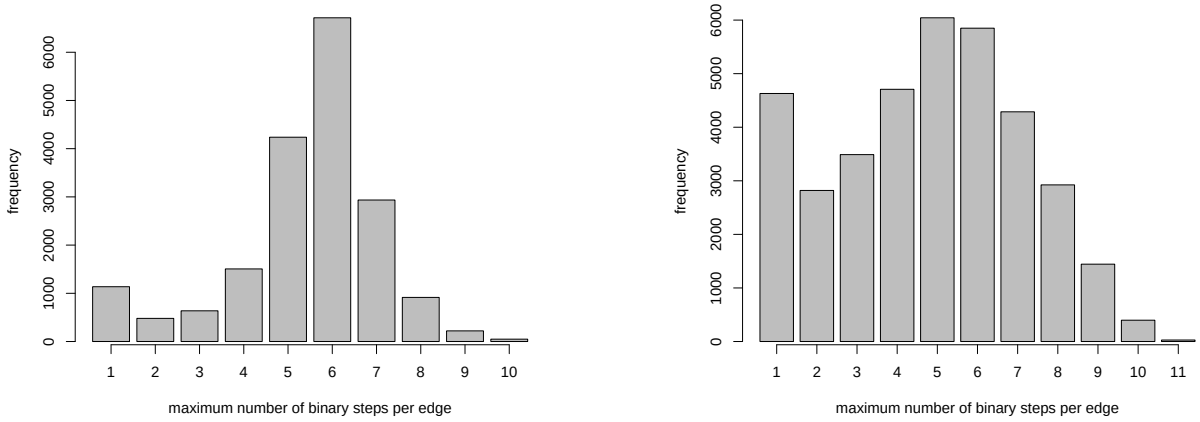

Figure 4: In the time-dependent model using binary search, for an edge e having c elementary connections at most $\lceil \log_2(c) \rceil + 1$ steps have to be carried out. The histograms show the distribution of that value over all edges (on the left for `ger-longdist`, and on the right for `ger-local2`).

where $d(u,t)$ is the Euclidean or Manhattan distance between nodes u and t, based on the coordinates of (the stations corresponding to) u and t. The parameter λ_t is called the *scaling factor* and is a non-negative number (which can be different for each destination node t) that is used to scale the distances so that $wt'(u,v,\tau)$ is non-negative. The scaling factor is defined as

$$\lambda_t = \min_{(u,v)\in E, d(u,t)-d(v,t)>0} \frac{\min_\tau wt(u,v,\tau)}{d(u,t)-d(v,t)}.$$

Note that λ_t corresponds to the inverse of the maximum speed considered in [12]; however, in [12] the maximum speed is the same for all destinations t, while λ_t depends on t.

Let $(\alpha,\beta) \in E$ be the edge with the above property and $wt(\alpha,\beta,\tau_0) = \min_\tau wt(\alpha,\beta,\tau)$. Note that there has to be an edge $(\alpha,\beta) \in E$ such that $d(u,t) - d(v,t) > 0$, since otherwise there would be no way reaching t – the distance to get there from every node $u \in V$ could never be reduced. Then for every edge $(u,v) \in E$ with $d(u,t) > d(v,t)$ we have

$$\frac{\min_\tau wt(u,v,\tau)}{d(u,t)-d(v,t)} \geq \frac{wt(\alpha,\beta,\tau_0)}{d(\alpha,t)-d(\beta,t)} \tag{1}$$

$$wt'(u,v,\tau') = wt(u,v,\tau') - p[u] + p[v], \quad \tau' \geq 0 \tag{2}$$

From Eq. (2) we get

$$wt'(u,v,\tau') = wt(u,v,\tau') - wt(\alpha,\beta,\tau_0)\frac{d(u,t)-d(v,t)}{d(\alpha,t)-d(\beta,t)}$$

hence

$$wt'(u,v,\tau') \geq min_\tau wt(u,v,\tau) - wt(\alpha,\beta,\tau_0)\frac{d(u,t)-d(v,t)}{d(\alpha,t)-d(\beta,t)}$$

and from Eq. (1) we obtain that

$$\min_\tau wt(u,v,\tau) \geq wt(\alpha,\beta,\tau_0)\frac{d(u,t)-d(v,t)}{d(\alpha,t)-d(\beta,t)}$$

which consequently implies that

$$wt'(u,v) \geq 0.$$

Now, if $d(u,t) - d(v,t) \leq 0$, then $-[d(u,t) - d(v,t)] \geq 0$ and $-\lambda_t[d(u,t) - d(v,t)] \geq 0$, which means that $wt'(u,v) = wt(u,v) - \lambda_t[d(u,t) - d(v,t)] \geq wt(u,v) \geq 0$.

Even if all edge weights are integers, the use of the Euclidean or Manhattan distances as potentials, forces us to use floating point numbers in the priority queue, and this may result in an additional time overhead. In order to avoid this, we can transform the floating-point potentials to integers without invalidating the potentials, as the following proposition shows.

Proposition 1 *If for the non-negative numbers $w \in \mathbb{N}$, $a, b \in \mathbb{R}^+$ it holds that $w - a + b \geq 0$, then it will also hold that $w - \lfloor a \rfloor + \lfloor b \rfloor \geq 0$.*

Proof. If $a = b$, the proposition holds trivially. Let $fr(a) = a - \lfloor a \rfloor$ and $fr(b) = b - \lfloor b \rfloor$. Consider first the case $a < b$. Then, $a - b < 0 \Rightarrow \lfloor a \rfloor - \lfloor b \rfloor < fr(b) - fr(a)$. Assume that $w - \lfloor a \rfloor + \lfloor b \rfloor < 0$. Since $w \geq 0$, we must have that $\lfloor a \rfloor - \lfloor b \rfloor > 0$, and hence $0 < \lfloor a \rfloor - \lfloor b \rfloor < fr(b) - fr(a)$. But, $fr(b) - fr(a) < 1$ and $\lfloor a \rfloor - \lfloor b \rfloor$ is an integer, a contradiction.

We turn now to the case $a > b$. Assume again that $w - \lfloor a \rfloor + \lfloor b \rfloor < 0$. Then, $w < \lfloor a \rfloor - \lfloor b \rfloor$. We also have $\lfloor a \rfloor - \lfloor b \rfloor \leq w + fr(b) - fr(a)$. Combining the last two inequalities and the fact that $fr(b) - fr(a) < 1$, we get that $w < \lfloor a \rfloor - \lfloor b \rfloor < w + 1$, which is again a contradiction. ∎

Time-Dependent Model, Binary Search

	Timetable	Real	CPU [ms]		El. conn.	Nodes	Edges
			LEDA	Own			
Goal Eucl. int	france		10.2	9.4	7072	1593	3415
	ger-longdist		14.4	13.5	16597	2737	4217
	ger-local1		30.9	28.6	26008	6257	9434
	ger-local2		33.1	30.4	31196	6398	9895
	ger-all		101.9	100.3	74525	14568	24030
	ger-longdist	✗	7.2	6.3	8349	1238	1991
	ger-all	✗	45.5	43.1	33676	5551	9420
Goal Eucl. float	france		10.3	9.4	7062	1590	3410
	ger-longdist		14.7	13.6	16560	2730	4208
	ger-local1		32.8	28.9	25975	6249	9422
	ger-local2		34.2	30.7	31152	6389	9882
	ger-all		103.9	103.6	74394	14538	23983
	ger-longdist	✗	7.3	6.4	8318	1233	1984
	ger-all	✗	46.1	44.5	33565	5532	9388
Goal Manh. int	france		8.3	7.9	7225	1647	3511
	ger-longdist		11.8	11.2	16975	2807	4316
	ger-local1		25.1	23.2	26086	6284	9473
	ger-local2		26.6	24.7	31235	6407	9908
	ger-all		86.0	86.4	74822	14639	24138
	ger-longdist	✗	6.0	5.3	8555	1272	2041
	ger-all	✗	39.4	38.0	33994	5615	9524
Goal Manh. float	france		8.7	7.7	7214	1644	3505
	ger-longdist		12.4	11.1	16938	2800	4306
	ger-local1		26.5	23.1	26053	6276	9461
	ger-local2		28.2	24.6	31189	6398	9894
	ger-all		89.7	88.9	74689	14608	24091
	ger-longdist	✗	6.2	5.2	8524	1267	2034
	ger-all	✗	40.9	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; CPU-times are given for the LEDA priority queue implementation and our own priority queue.

Time-Expanded Model

	Timetable	Real	CPU [ms]	El. Conn.	Nodes	Edges
Goal Eucl. int	france		84.0	22259	24179	44517
	ger-longdist		175.0	34259	37453	68517
	ger-local1		684.3	170369	176243	340741
	ger-local2		953.0	219992	226386	439986
	ger-all		1392.6	285440	300788	570885
	ger-longdist	✗	54.3	13384	14931	26768
	ger-all	✗	341.9	74069	80229	148140

Time-Dependent Model, Avoid Binary Search

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