

# Efficient Models for Timetable Information in Public Transportation Systems

EVANGELIA PYRGA

Max-Planck-Institut für Informatik

FRANK SCHULZ and DOROTHEA WAGNER

University of Karlsruhe

and

CHRISTOS ZAROLIAGIS

CTI and University of Patras

2.4

---

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 (a simplified version of) the earliest arrival problem, but little is known about their relative performance. Thus far, there are only theoretical arguments in favor of the time-dependent approach. In this paper, we provide the first extensive experimental comparison of the two approaches. Using several real-world data sets, we evaluate the performance of the basic models and of several new extensions towards realistic modeling. Furthermore, new insights on solving bicriteria optimization 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.

---

This work was partially supported by the Human Potential Programme of EC under contract no. HPRN-CT-1999-00104 (AMORE), by the Future and Emerging Technologies Unit of EC (IST priority – 6th FP), under contracts no. IST-2002-001907 (DELIS) and no. FP6-021235-2 (ARRIVAL), and by the DFG under grant WA 654/1-12. Part of this work has been done while the first author was with the University of Patras, the second and the third authors were with the University of Konstanz, and while the last author was visiting the University of Karlsruhe.

Authors' addresses: Evangelia Pyrga, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany; email: pyrga@mpi-sb.mpg.de. Frank Schulz, Department of Computer Science, University of Karlsruhe, 76128 Karlsruhe, Germany; email: fschulz@ira.uka.de. Dorothea Wagner, Department of Computer Science, University of Karlsruhe, 76128 Karlsruhe, Germany; email: wagner@ira.uka.de. Christos Zaroliagis, R.A. Computer Technology Institute, N. Kazantzaki Str, Patras University Campus, 26500 Patras, Greece, and Department of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece; email: zaro@ceid.upatras.gr.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org. © 2007 ACM 1084-6654/2007/ART2.4 \$5.00 DOI 10.1145/1227161.1227166 http://doi.acm.org/10.1145/1227161.1227166

Categories and Subject Descriptors: G.2.1 [**Combinatorics**]: Combinatorial algorithms; G.2.2 [**Graph Theory**]: Graph algorithms, Network problems, Path and circuit problems; G.2.3 [**Applications**]: Traffic information systems; G.4 [**Mathematical Software**]: Algorithm design and analysis; D.2.8 [**Metrics**]: Performance Measures; E.1 [**Data Structures**]: Graphs and Networks

General Terms: Algorithms, Design, Experimentation, Measurement, Performance

Additional Key Words and Phrases: Shortest path, timetable information, public transportation system, itinerary query

**ACM Reference Format:**

Pyrga, E., Schulz, F., Wagner, D., and Zaroliagis, C. 2007. Efficient models for timetable information in public transportation systems. *ACM J. Exp. Algor.* 12, Article 2.4 (2007), 39 pages DOI 10.1145/1227161.1227166 <http://doi.acm.org> 10.1145/1227161.1227166

## 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.<sup>1</sup> 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* [Müller-Hannemann and Weihe 2001; Pallottino and Scutellà 1998; Schulz et al. 2000, 2002], and the *time-dependent* approach [Brodal and Jacob 2004; Nachtigal 1995; Orda and Rom 1990, 1991]. 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 [Schulz et al. 2000] 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 costs to the edges. This naturally results in the construction of a very large (but usually sparse) graph. The time-dependent approach [Brodal and Jacob 2004] 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 costs on the edges are assigned “on-the-fly,” i.e., the cost of an edge depends on the time in which the particular edge will be used by the shortest-path algorithm to answer the query.

<sup>1</sup>For example, the server of the German railways receives about 100 queries per second.

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

Techniques for solving general multicriteria optimization problems have been discussed in Möhring [1999] and Müller-Hannemann and Weihe [2001], where the discussion in the former is focused on a distributed approach for timetable information problems. Space consumption aspects of modeling more complex real-world scenarios is considered in Müller-Hannemann et al. [2002]. For the time-expanded model, the simplified version of the earliest-arrival problem has been extensively studied [Schulz et al. 2000, 2002], and an extension of the model able to solve the minimum number of transfers problem, but without transfer times, is discussed in Müller-Hannemann and Weihe [2001]. Comparing the time-expanded and time-dependent approach, it is argued theoretically in Brodal and Jacob [2004] 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 present 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 optimization problems combining the earliest-arrival and the minimum number of transfers problems.

We also conducted extensive experiments comparing the extended approaches. This 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.

The rest of this paper is organized as follows. In Section 2, the variants of itinerary problems that are considered in this paper are defined. The modeling of the simplified version of the earliest-arrival problem is considered in Section 3, where the basic ideas of the original time-expanded and time-dependent models are briefly reviewed. In Section 4 the new extensions to these approaches are presented in order to cope with the realistic version of the earliest-arrival

problem. Section 5 discusses the incorporation of different traffic days in the original or in the extended models. Sections 6 and 7 discuss how the minimum number of transfers problem and the bicriteria optimization problems, respectively, can be solved in either of the extended models. Heuristics that speedup the proposed algorithmic solutions are discussed in Section 8. The experimental comparison of the two approaches based on real-world data from the German and French railways is presented in Section 9. We first consider how the original versions of the two approaches compare and, subsequently, investigate the comparison of the extended models on the realistic version of the earliest-arrival problem, the minimum number of transfers problem, and the bicriteria optimization problems. Section 10 summarizes our insights on the advantages and disadvantages of the approaches under comparison. Preliminary parts of this work appeared in Pyrga et al. [2004a, 2004b].

## 2. ITINERARY PROBLEMS

In this section, we provide definitions of the timetable problems that we will consider. Problem definitions, as well as models and algorithms that will be presented throughout the paper for their efficient solution, refer to timetable information in a railway system, but the modeling 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*  $\mathcal{Z}$ , a set of stations  $\mathcal{B}$ , and a set of *elementary connections*  $\mathcal{C}$ , whose elements  $c$  are 5-tuples of the form  $c = (Z, S_1, S_2, t_d, t_a)$ . Such a tuple (elementary connection) is interpreted as train  $Z$  leaves station  $S_1$  at time  $t_d$  and the *immediately next* stop of train  $Z$  is station  $S_2$  at time  $t_a$ . If  $x$  denotes a tuple's field, then the notation  $x(c)$  specifies the value of  $x$  in the elementary connection  $c$ .

The *departure* and *arrival times*  $t_d(c)$  and  $t_a(c)$  of an elementary connection  $c \in \mathcal{C}$  within a day are integers in the interval  $[0, 1439]$  representing time in minutes after midnight. Given two time values  $t$  and  $t'$ ,  $t \leq t'$ , the *cycle difference*  $(t, t')$  is the smallest nonnegative integer  $l$  such that  $l \equiv t' - t \pmod{1440}$ . The *length* of an elementary connection  $c$ , denoted by  $length(c)$ , is  $cycle\ difference(t_d(c), t_a(c))$ . A timetable is valid for a number of  $N$  *traffic days*, and every train is assigned a bit-field of  $N$  bits determining on which traffic days the train operates (for overnight trains the departure of the first elementary connection counts). We will generally assume that trains operate daily; in Section 5 we will discuss how different traffic days can be incorporated in the models.

At a station  $S \in \mathcal{B}$  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, *minimum transfer time*, denoted by  $transfer(S)$ .

Let  $P = (c_1, \dots, c_k)$  be a sequence of elementary connections together with departure times  $dep_i(P)$  and arrival times  $arr_i(P)$  for each elementary

connection  $c_i$ ,  $1 \leq i \leq k$ . We assume that the times  $dep_i(P)$  and  $arr_i(P)$  include data regarding also the departure/arrival day by counting time in minutes from the first day of the timetable. Such a time  $t$  is of the form  $t = a \cdot 1440 + b$ , where  $a \in [0, N - 1]$  and  $b \in [0, 1439]$ . Hence, the actual time within a day is  $t \pmod{1440}$  and the actual day is  $\lfloor t/1440 \rfloor$ . Such a sequence  $P$  is called a *consistent connection* from station  $A = S_1(c_1)$  to station  $B = S_2(c_k)$  if it fulfills some consistency conditions: (a) the departure station of  $c_{i+1}$  is the arrival station of  $c_i$ ; (b) the time values  $dep_i(P)$  and  $arr_i(P)$  correspond to the time values  $t_d$  and  $t_a$ , resp., of the elementary connections (modulo 1440) and respect the transfer times at stations. More formally,  $P$  is a *consistent connection* if the following conditions are satisfied:

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

For the timetable-information problems defined below, we are, in addition, 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) with respect to 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.

## 2.1 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  $transfer(S) = 0$  for all stations  $S$ . (b) The *realistic version*, where train transfers require arbitrary nonnegative minimum transfer times  $transfer(S)$ . We discuss efficient solutions to these problems in Sections 3 and 4. A special case of the earliest-arrival problem is the *latest-departure* problem: among all connections with earliest-arrival time at the arrival station, maximize the actual departure time at the departure station. Efficient solutions to this problem are discussed in Section 10.

## 2.2 Minimum Number of Transfers Problem (MNTP)

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.<sup>2</sup> All connections from  $A$  to  $B$  are valid and the optimization criterion is to minimize the number of train transfers. In particular, let  $P = (c_1, \dots, c_k)$  be a connection from  $A$  to  $B$  and let  $trans_i(P) \in \{0, 1\}$  be a variable denoting whether a transfer is needed from elementary connection  $c_i$  to  $c_{i+1}$ ,  $1 \leq i < k$ . Then,  $trans_i(P) = 1$ , if  $Z(c_{i+1}) \neq Z(c_i)$ , and  $trans_i(P) = 0$ , otherwise. Consequently, the objective of MNTP is to minimize, among all  $P$ , the quantity  $\sum_{i=1}^{k-1} trans_i(P)$ . We will discuss this problem in Section 6.

## 2.3 Bicriteria Optimization 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 three problem variants: (1) finding the so-called *Pareto-curve*, which is the set of all 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); (2) finding the solution that minimizes one criterion while retaining the second below a given threshold; (3) finding the lexicographically first Pareto-optimal solution (e.g., find among all connections that minimize EA the one with the minimum number of transfers). We will discuss these problems in Section 7.

## 3. EARLIEST-ARRIVAL PROBLEM—SIMPLIFIED VERSION

In this section, we review the modeling of the simplified version of EAP, in both the time-expanded and the time-dependent approach. Recall that, in this case, transfer time between trains at a station is negligible.

### 3.1 Time-Expanded Model

The time-expanded model [Schulz et al. 2000] 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  $t_d$ , to 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  $v_1, \dots, v_k$  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 cost of an edge  $(p, q)$  is *cycle difference* $(t_p, t_q)$ , where  $t_p$  and  $t_q$  are the time values associated with nodes  $p$  and  $q$ , respectively. Figure 1

<sup>2</sup>This assumption can be safely made since time is not minimized in MNTP and, thus, in a MNT-optimal connection, one can wait arbitrarily long at a station for some connection that is valid only on certain days.

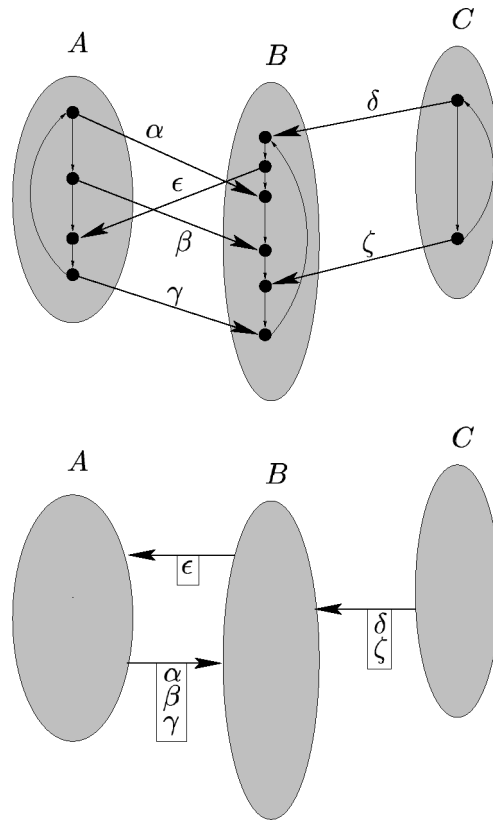


Fig. 1. The time-expanded (*left*) and the time-dependent digraph (*right*) of a timetable with three stations A, B, C. There are three trains that connect A with B (elementary connections  $\alpha$ ,  $\beta$ ,  $\gamma$ ), one train from C via B to A ( $\delta$ ,  $\epsilon$ ), and one train from C to B ( $\zeta$ ).

illustrates this definition. The following theorem states that an appropriate shortest-path computation on the above graph solves the simplified version of EAP.

**THEOREM 1.** *A shortest path in the time-expanded digraph from the first departure node  $s$  at the departure station A with departure time later than or equal to the given start time  $t_0$  to one of the arrival nodes of the destination station B constitutes a solution to the simplified version of EAP in the time-expanded model. The actual path can be found by Dijkstra's algorithm in time  $O(m_{e_o} + n_{e_o} \log n_{e_o})$ , where  $n_{e_o}$  (resp.  $m_{e_o}$ ) is the number of nodes (resp. edges) of the time-expanded digraph.*

**PROOF.** An optimal connection never stays more than a day at a station, because otherwise there would be a 1 day faster connection, since every train is assumed to operate daily.<sup>3</sup> Let  $\Pi$  be a path in the time-expanded digraph from

<sup>3</sup>When traffic days are modeled, this is not necessarily true anymore; see Section 5.

node  $s$  of station  $A$  to some arrival node of station  $B$ . To prove correctness, it suffices to show that there is a one-to-one correspondence between path  $\Pi$  and a valid and consistent  $A$ – $B$  connection  $P$ , such that the cost of path  $\Pi$  equals the duration of  $P$ . Clearly, such a one-to-one correspondence applies between the shortest among those paths  $\Pi$  and the minimum in duration among all those valid and consistent connections  $P$ .

We will first show that path  $\Pi$  corresponds to a valid and consistent  $A$ – $B$  connection  $P$  in the EAP setting. Define  $P$  to be the sequence of all elementary connections corresponding to the train edges that occur in the path  $\Pi$ . Further, define the time value  $dep_i(P)$  to be the cost of the subpath from  $s$  to the tail of the  $i$ th train edge plus the time associated with  $s$ . Similarly, set  $arr_i(P)$  to be the cost of the subpath from  $s$  to the head of the  $i$ th train edge plus the time associated with  $s$ . It can be easily verified that  $P$  is a valid and consistent  $A$ – $B$  connection, whose duration equals the cost of path  $\Pi$ .

Conversely, let  $P'$  be a valid and consistent  $A$ – $B$  connection that never stays more than a day at a station, departing from  $A$  at  $t_0$  or at the earliest possible time after  $t_0$ , and let the corresponding departure node in the time-expanded digraph be  $s$ . We will show that  $P'$  corresponds to a path  $\Pi'$  starting from node  $s$  of station  $A$  and ending at some arrival node of  $B$  in the time-expanded digraph. Path  $\Pi'$  is constructed as follows. It contains all train edges corresponding to the elementary connections in  $P'$ . Path  $\Pi'$  starts at  $s$ , which is connected by a simple path of stay edges with the tail of the first train edge. The head of the first train edge is connected by a path of stay edges to the tail of the second train edge, and so on, until the tail of the last train edge is reached. Clearly, the head of the last train edge is an arrival node of  $B$ . It can be easily verified that the cost of path  $\Pi'$  is equal to the difference of the arrival and the departure time of  $P'$ .

Since edge costs are nonnegative, the actual path can be found by using Dijkstra's algorithm [Dijkstra 1959] and abort the main loop when a node at the destination station is reached.  $\square$

### 3.2 Time-Dependent Model

The time-dependent model [Brodal and Jacob 2004] 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  $A$ – $B$  if there is an elementary connection from  $A$  to  $B$ . The set of elementary connections from  $A$  to  $B$  is denoted by  $\mathcal{C}(e)$ . The definition is illustrated in Figure 1. The cost of an edge  $e = (A, B)$  depends on the time at which this particular edge will be used by an algorithm that solves EAP. In other words, if  $T$  is a set denoting time, then the cost of an edge  $(A, B)$  is given by  $f_{(A,B)}(t) - t$ , where  $t$  is the departure time at  $v$ ,  $f_{(A,B)} : T \rightarrow T$  is a function such that  $f_{(A,B)}(t) = t'$ , and  $t' \geq t$  is the earliest possible arrival time at  $B$ . The time-dependent model is based on the assumption that overtaking of trains on an edge is not allowed.

**ASSUMPTION 1.** *For any two given stations  $A$  and  $B$ , there are no two trains leaving  $A$  and arriving to  $B$  such that the train that leaves  $A$  second arrives first at  $B$ .*

A modification of Dijkstra’s algorithm can be used to solve the earliest arrival problem in the time-dependent model [Brodal and Jacob 2004]. Let  $D$  denote the departure station,  $t_0$  the earliest departure time, and  $\delta : V \rightarrow \mathbb{R}$  the distance labels maintained by Dijkstra’s algorithm. 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 costs on-the-fly. The edge costs (and, implicitly, the time-dependent function  $f$ ) are calculated as follows. Since Dijkstra’s algorithm is a label-setting shortest-path algorithm, whenever an edge  $e = (A, B)$  is considered, the distance label  $\delta(A)$  of node  $A$  is optimal. In the time-dependent model,  $\delta(A)$  denotes the earliest-arrival time at station  $A$ . In other words, we, indeed, know the earliest-arrival time at station  $A$  whenever the edge  $e = (A, B)$  is considered, and, therefore, we know at that stage of the algorithm (by Assumption 1) 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  $A$ . Let  $t = \delta(A)$  and let  $c^* \in \mathcal{C}(e)$  be the connection minimizing  $\{cycle\ difference(t, t_d(c)) \mid c \in \mathcal{C}(e)\}$ . The particular connection  $c^*$  can be easily found by binary search if the elementary connections  $\mathcal{C}(e)$  are maintained in a sorted array. The edge cost of  $e$ ,  $\ell_e(t)$ , is then defined as  $\ell_e(t) = cycle\ difference(t, t_d(c^*)) + length(c^*)$ . Consequently,  $f_e(t) = t + \ell_e(t)$ .

The following theorem is proved in Brodal and Jacob [2004]. Its correctness is based on the fact that  $f$  has *nonnegative delay* ( $\forall t, f(t) \geq t$ ), and is *non-decreasing* ( $t \leq t' \Rightarrow f(t) \leq f(t')$ ), which follows directly from Assumption 1. Because of the nature of the investigated application, we can safely assume that all functions defined throughout the paper have nonnegative delay.

**THEOREM 2.** *The above modified Dijkstra’s algorithm solves the simplified version of EAP in the time-dependent model, provided that Assumption 1 holds, in time  $O(m_{d_o} \log W_o + n_{d_o} \log n_{d_o})$ , where  $n_{d_o}$  (resp.  $m_{d_o}$ ) is the number of nodes (resp. edges) of the time-dependent digraph, and  $W_o$  is the maximum number of elementary connections associated with an edge.*

#### 4. EARLIEST-ARRIVAL PROBLEM—REALISTIC VERSION

In this section, we describe how both models of Section 3 can be extended to solve the realistic version of EAP, where transfer time between trains at a station is non-zero.

##### 4.1 Time-Expanded Model

To solve the realistic version of EAP, in this case, we extend the time-expanded model by constructing the *realistic time-expanded* digraph as follows. Based on the time-expanded digraph, we keep, for each station, a copy of all departure and arrival nodes in the station, which we call *transfer nodes*; see Figure 2. The stay edges are now introduced between the transfer nodes. For every arrival node there is now one edge to the first 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 corresponding station, and a second edge from the arrival node to the departure node of the same train, if it departs from the corresponding station (otherwise there is no second edge). The edge costs are defined as in

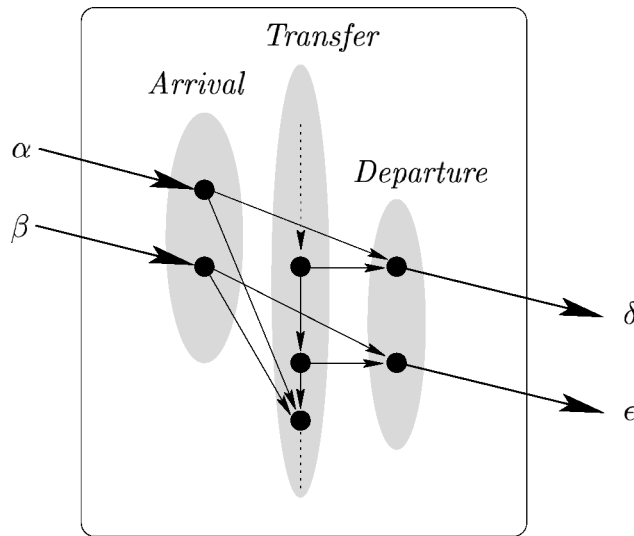


Fig. 2. Modeling train transfers in the time-expanded model for a sample station. The realistic time-expanded digraph has three types of nodes: arrival, transfer and departure nodes.

the definition of the original model (see Section 3.1). The correctness of the modeling is established by the following theorem.

**THEOREM 3.** *A shortest path in the realistic time-expanded digraph from the first departure node at the departure station  $A$  with departure time later than or equal to the given start time  $t_0$  to one of the arrival nodes of the destination station  $B$ , constitutes a solution to the realistic version of EAP in the extended time-expanded model. The actual path can be found by Dijkstra’s algorithm in time  $O(m_e + n_e \log n_e)$ , where  $n_e$  (resp.  $m_e$ ) is the number of nodes (resp. edges) of the realistic time-expanded digraph.*

**PROOF.** The proof of Theorem 1 that allows construction of paths from connections (and vice versa) can be analogously adapted to establish an one-to-one correspondence between paths and valid and consistent connections in the realistic version of EAP.  $\square$

#### 4.2 Time-Dependent Model

To model non-zero train transfers, we extend the original time-dependent model using information on the routes that trains may follow. We examine both the cases of constant and variable transfer times (a somehow similar idea for the constant transfer time case was very briefly mentioned in Brodal and Jacob [2004], but without providing the details). In the following, we describe the construction of a digraph  $G = (V, E)$  that models these two cases. We shall refer to  $G$  as the *train-route digraph*.

Since now there is no one-to-one correspondence between nodes and stations, we denote nodes of the train-route digraph with small latin letters  $u, v, \dots$ . Let  $S$  be a set of nodes representing stations (i.e., for each station in the input there

is a node in  $S$  that corresponds to that station, and each  $u \in S$  corresponds to a different station). For  $u \in S$ , we denote by  $station(u)$  the actual station, which  $u$  represents ( $station(\cdot)$  is a function mapping a node of  $S$  to its corresponding station).

We divide the set of trains  $\mathcal{Z}$  into so-called *train routes*: A train route  $r$  consists of a maximal subset of the set of trains  $\mathcal{Z}$ , such that each train of  $r$  follows the exact same route, i.e., passes through the exact same stations, at the exact same order, possibly at different times throughout the day. In other words, two trains are equivalent concerning their route if the sequence of stations the trains pass is the same. The train routes are the equivalence classes of this relation.

For each different route  $r$ , let  $k_r > 0$  be the number of stations, not all of them necessarily distinct, that  $r$  stops at (including the first station of the route). Denote by  $S_1, S_2, \dots, S_{k_r}$  the stations that  $r$  visits, in this order. (It can be  $S_i = S_j$  for  $i \neq j$ , in the general case that  $r$ , at some point, forms a loop.) Let  $v_1, \dots, v_{k_r} \in S$  be the nodes for which  $station(v_i) = S_i$ ,  $1 \leq i \leq k_r$ , and define sets  $\mathcal{P}_{v_i}$  as follows: for each  $i$ ,  $1 \leq i \leq k_r$ , insert a new route node to  $\mathcal{P}_{v_i}$  modeling the fact that  $r$  passes through  $station(v_i)$ . Thus, for each  $u \in S$ , a node set  $\mathcal{P}_u$  is constructed by repeating this procedure for all train routes.

Also, let  $P_u = |\mathcal{P}_u|$ , and  $\mathcal{P} = \bigcup_{u \in S} \mathcal{P}_u$ . Then, the node set  $V$  of  $G$  is defined as  $V = S \cup \mathcal{P}$ . For  $u \in S$ , we denote by  $p_i^u$ ,  $0 \leq i < P_u$ , the  $P_u$  nodes contained in  $\mathcal{P}_u$ , in some arbitrary order.

We distinguish four types of edges for the train-route digraph: edges  $I$  from a route node to a station node, edges  $D$  from a station node to a route node, edges  $\overline{D}$  between route nodes of the same station, and route edges  $R$  between route nodes of the same route. More formally, the edge set  $E = I \cup D \cup \overline{D} \cup R$  of  $G$  is defined as follows.

- $I = \bigcup_{u \in S} I_u$ , where  $I_u = \bigcup_{0 \leq i < P_u} \{(p_i^u, u)\}$ .
- $D = \bigcup_{u \in S} D_u$ , where  $D_u = \bigcup_{0 \leq i < P_u} \{(u, p_i^u)\}$ .
- $\overline{D} = \bigcup_{u \in S} \overline{D}_u$ , where  $\overline{D}_u = \emptyset$ , if the time needed for a transfer is the same for all trains that stop to  $station(u)$ ;  $\overline{D}_u = \bigcup_{0 \leq i, j < P_u, i \neq j} \{(p_i^u, p_j^u)\}$ , otherwise.
- $R = \bigcup_{u, v \in S, 0 \leq i < P_u, 0 \leq j < P_v} \{(p_i^u, p_j^v) : station(u) \text{ and } station(v) \text{ are visited successively by the same train route and } p_i^u, p_j^v \text{ are the corresponding route nodes that belong to } \mathcal{P}_u \text{ and } \mathcal{P}_v, \text{ respectively}\}$ .

An edge  $e$  is called a *route* or *timetable edge* if  $e \in R$ , and it is called a *transfer edge* if  $e \in I \cup D \cup \overline{D}$ . The modeling with train routes is based on two additional assumptions.

**ASSUMPTION 2.** *Let  $u, v$  be any two nodes in  $S$  and  $p_i^u \in \mathcal{P}_u$ ,  $p_j^v \in \mathcal{P}_v$  such that  $(p_i^u, p_j^v) \in R$ . If  $d_1, d_2$  are departure times from  $p_i^u$  and  $a_1, a_2$  are the respective arrival times to  $p_j^v$ , then  $d_1 \leq d_2 \Rightarrow a_1 \leq a_2$ .*

This assumption states that there cannot be two trains that belong to the same train route, such that the first of them leaving a station is a slow train, while the following one is a fast train and it arrives to the next station before the first. When this assumption is violated, we can enforce it by separating

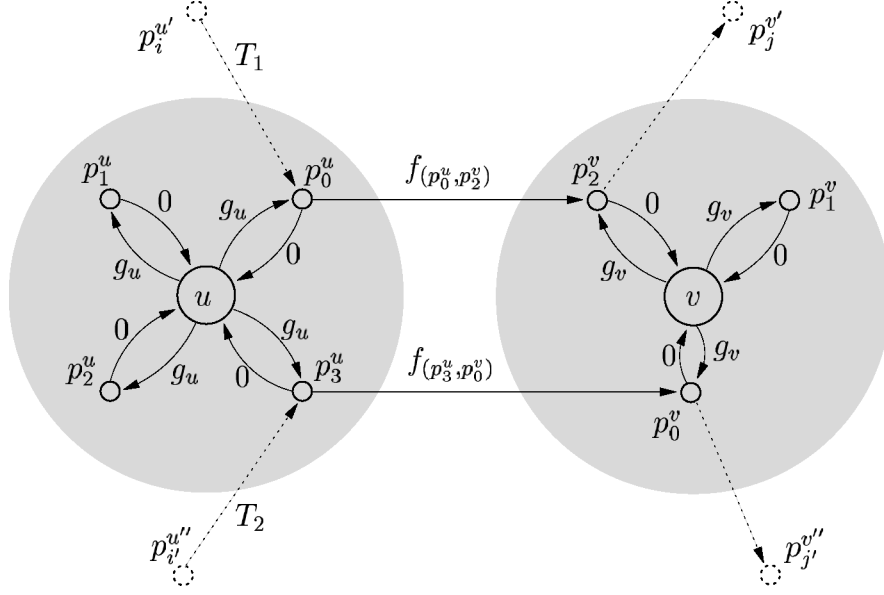


Fig. 3. An example of modeling through train routes with constant transfer time per station.

the trains that belong to the same train route into different speed classes and, hence, introduce new train routes, one for each different speed class, where all follow the same schedule as before.

**ASSUMPTION 3.** For any  $u \in S$  and  $p_i^u \in \mathcal{P}_u$  such that  $(x, p_i^u) \in R$ , for some  $x \in V$ , let  $\delta_x^{u_i}$  be the smallest interval between two successive arrivals to  $p_i^u$  from  $(x, p_i^u) \in R$  and  $\tau_u$  be the maximum time needed for a transfer at station  $(u)$ . Then, it must hold that  $\delta_x^{u_i} \geq \tau_u$ .

Assumption 3 serves the purpose of ensuring that waiting at stations to take the next train of the same train route cannot be beneficial. In other words, given that Assumption 2 holds, taking the first possible train from a station  $A$  to some station  $B$  will not result in missing some connection from  $B$  that could be used if we had followed some train (of the same train route) that departed later than the one we followed. It will be seen later that Assumption 3 will only be needed in the case where we consider variable transfer times within the same station.

In the following, we shall assume that  $u, v \in S$ ,  $0 \leq i, i' < P_u$ ,  $0 \leq j < P_v$ , and that  $T$  is a set representing time.

**4.2.1 Constant Transfer Time.** In this case, the edge costs of the train-route digraph are defined as follows (see Figure 3).

- An edge  $(p_i^u, u) \in I$  has zero cost.
- An edge  $(u, p_i^u) \in D_u$  has cost determined by a function  $g_u : T \rightarrow T$ , such that  $g_u(t) = t + \tau_{change}^u$ , where  $\tau_{change}^u$  is the time needed for transferring from one train to another at station  $(u)$ .

- An edge  $(p_i^u, p_j^v) \in R$  has cost determined by a function  $f_{(p_i^u, p_j^v)} : T \rightarrow T$  such that  $f_{(p_i^u, p_j^v)}(t)$  is the time at which  $p_j^v$  will be reached using the edge  $(p_i^u, p_j^v)$ , given that  $p_i^u$  was reached at time  $t$ .

To solve the realistic version of EAP for a given query  $(A, B, t_0)$ , we need to find a shortest path from station  $A$  to  $B$  using the modified Dijkstra's algorithm (cf. Section 3.2), where for each edge we use its associated cost function as described above. We actually need to find the shortest path in the graph  $G = (V, E_{A,B})$  from  $s_A \in S$  to  $s_B \in S$  starting at time  $t_0$ , where  $station(s_A) = A$ ,  $station(s_B) = B$ , and  $E_{A,B} \equiv I \cup D \cup R$  (see Figure 3). Note that all edges  $e \in D_{s_A}$  must have zero cost.

**THEOREM 4.** *The above algorithm solves the realistic version of EAP with constant transfer times at stations in the extended time-dependent model, provided that Assumption 2 holds, in time  $O(m_d \log W + n_d \log n_d)$ , where  $n_d$  (resp.  $m_d$ ) is the number of nodes (resp. edges) of the train-route digraph, and  $W$  is the maximum number of elementary connections associated with an edge.*

**PROOF.** In view of the discussion in Section 3.2, regarding the correctness of Theorem 2, the correctness of the above algorithm for solving the realistic version of EAP, when the transfer time in each station is constant, follows from the nonnegative delay assumptions of functions  $f$  and  $g$ , and by the fact that both functions are nondecreasing ( $f$  by Assumption 2 and  $g$  by construction).  $\square$

**4.2.2 Variable Transfer Time.** The edge costs of the train-route digraph in this case are defined as follows (see also Figure 4).

- An edge  $(p_i^u, u) \in I$  has zero cost.
- An edge  $(u, p_i^u) \in D_u$  has zero cost. An edge  $(p_i^u, p_{i'}^u) \in \overline{D}_u$  has cost determined by a function  $f_{(p_i^u, p_{i'}^u)} : T \rightarrow T$  such that  $f_{(p_i^u, p_{i'}^u)}(t)$  represents the time required by a passenger who reached  $station(u)$  at time  $t$  with a train of the train route  $p_i^u$ , to be transferred to the first possible train of route  $p_{i'}^u$ . In particular,  $f_{(p_i^u, p_{i'}^u)}(t) = t + \tau_{change(i,i')}^u(t)$ , where  $\tau_{change(i,i')}^u(t)$  is the function that, for each arriving time, returns the corresponding transfer time.
- An edge  $(p_i^u, p_j^v) \in R$  has cost determined by a function  $f_{(p_i^u, p_j^v)} : T \rightarrow T$  such that  $f_{(p_i^u, p_j^v)}(t)$  is the time at which  $p_j^v$  will be reached using the edge  $(p_i^u, p_j^v)$ , given that  $p_i^u$  was reached at time  $t$ .

As before, given a query  $(A, B, t_0)$ , all we need is to find a shortest path from station  $A$ – $B$  in the above graph, where for each edge we use its associated cost function as described above. This is again accomplished by the modified Dijkstra's algorithm (Section 3.2). We actually need to find a shortest path in the graph  $G = (V, E_{A,B})$  from  $s_A$  to  $s_B$  starting at time  $t_0$ , but now  $E_{A,B} \equiv D_{s_A} \cup I_{s_B} \cup \overline{D} \cup R$ . Note that all edges  $e \in D_{s_A}$  must have zero cost.

Let nodes  $p_i^u, p_j^u \in \mathcal{P}_u$ ,  $0 \leq i, j < P_u$ ,  $i \neq j$ , such that  $(p_i^u, p_j^u) \in \overline{D}$ . In addition, let node  $p_{i'}^v \in \mathcal{P}_v$ ,  $0 \leq i' < P_v$ , such that  $(p_{i'}^v, p_i^u) \in R$ . In order to be able to apply the above algorithm and solve EAP in this case, we have to ensure that the functions are nondecreasing. Assumption 2 ensures that  $f_{(p_{i'}^v, p_i^u)}(t)$  is nondecreasing. What we need to prove is that  $f_{(p_i^u, p_j^u)}(t)$  is also

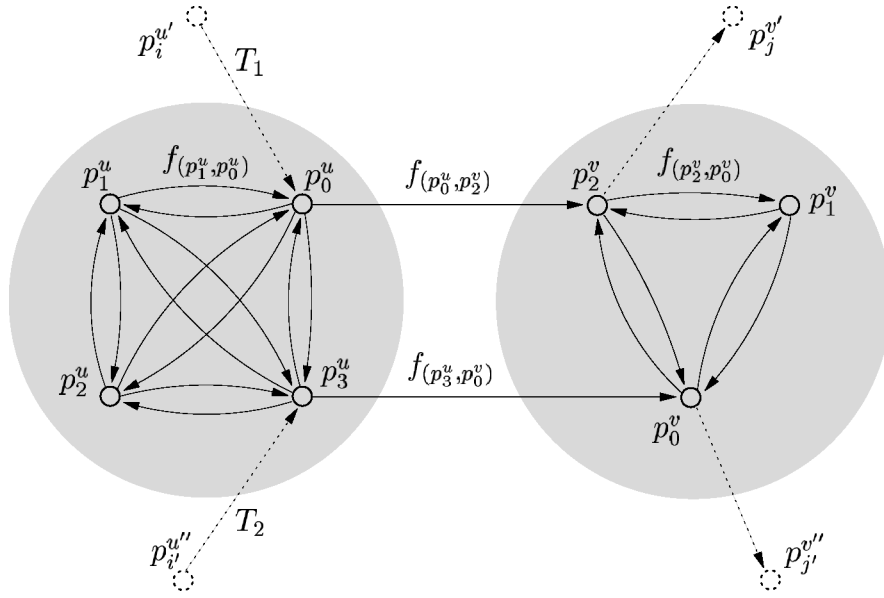


Fig. 4. An example of modeling through train routes with variable transfer times at the stations.

nondecreasing when  $t \in T_{u_i}$ , where  $T_{u_i} = \{t \mid t \text{ is the arrival time to } p_i^u \text{ from the edge } (p_i^v, p_i^u) \in R\}$ . Note that  $p_i^v$  is the only in-neighbor (i.e., tail of an incoming edge) of  $p_i^u$  that is not in  $\mathcal{P}_u$ .

LEMMA 1. *The function  $f_{(p_i^u, p_j^u)}(t)$  is nondecreasing when  $t \in T_{u_i}$ , where  $T_{u_i} = \{t \mid t \text{ is the arrival time to } p_i^u \text{ from the edge } (p_i^v, p_i^u) \in R\}$ .*

PROOF. Let  $\tau_u$  be the maximum time needed for a transfer at *station*( $u$ ), let  $\delta_{v_i^u}^{u_i}$  be the minimum interval between two successive arrivals to  $p_i^u$  from  $(p_i^v, p_i^u)$ , and let  $t_1, t_2 \in T_{u_i}$ , where  $t_1 < t_2$ . Since  $t_1, t_2$  are two distinct arrival times at  $p_i^u$  from  $(p_i^v, p_i^u)$ , then,  $t_2 - t_1 \geq \delta_{v_i^u}^{u_i}$ . By Assumption 3, we have that  $\delta_{v_i^u}^{u_i} \geq \tau_u$ . Also,  $f_{(p_i^u, p_j^u)}(t) - t \leq \tau_u$ ,  $t \in T$ , since  $f_{(p_i^u, p_j^u)}(t) - t$  is the time needed for a transfer from  $p_i^u$  to  $p_j^u$  on time  $t$ . Consequently,

$$\begin{aligned} t_2 - t_1 &\geq \delta_{v_i^u}^{u_i} \geq \tau_u \geq f_{(p_i^u, p_j^u)}(t_1) - t_1 \\ &\Rightarrow t_2 - t_1 \geq f_{(p_i^u, p_j^u)}(t_1) - t_1 \\ &\Rightarrow f_{(p_i^u, p_j^u)}(t_1) \leq t_2 \leq f_{(p_i^u, p_j^u)}(t_2) \\ &\Rightarrow f_{(p_i^u, p_j^u)}(t_1) \leq f_{(p_i^u, p_j^u)}(t_2) \end{aligned}$$

which completes the proof of the lemma.  $\square$

Lemma 1 and the above discussion establish the next theorem whose proof follows that of Theorem 4 (cf. Section 4.2.1).

THEOREM 5. *The above algorithm solves the realistic version of EAP with variable transfer times at stations in the extended time-dependent model, provided that Assumptions 2 and 3 hold, in time  $O(m_d \log W + n_d \log n_d)$ , where  $n_d$*

(resp.  $m_d$ ) is the number of nodes (resp. edges) of the train-route digraph, and  $W$  is the maximum number of elementary connections associated with an edge.

## 5. INCORPORATING TRAFFIC DAYS

For all models described so far we have assumed that every train operates daily, i.e., the timetable is identical every day. In this section, we discuss how different traffic days can be integrated in the models. For each elementary connection we are given the information on which day of the timetable it is valid.

### 5.1 Time-Expanded Model

When traffic days are used, an optimal connection may stay for more than a day at an intermediate station (e.g., assume on a holiday no trains are operated at all at that station). Such connections do not correspond to simple paths in the previous time-expanded digraph, and the problem cannot be solved directly using that graph. Therefore, we introduce the *fully time-expanded* digraph to tackle the problem in the time-expanded approach. As we will see below, we do not have to explicitly maintain this graph. The execution of an algorithm on the fully time-expanded digraph can be simulated on the original (Section 3.1) or on the realistic time-expanded digraph (Section 4.1).

If the timetable is valid for  $N$  days, the fully time-expanded digraph is based on  $N$  copies of the (original or realistic) time-expanded digraph. Whenever there is an overnight edge in the  $i$ th copy, the edge is redirected to the corresponding node in the  $i + 1$ th copy, if  $i < N$ ; in the  $N$ th copy, overnight edges are deleted. Furthermore, in the  $i$ th copy all train edges that correspond to elementary connections that are not valid on day  $i$  are deleted from the graph. In the following, we assume that each node in the fully-expanded digraph is not only assigned the time of the day  $t_d$  in the interval  $[0, 1439]$ , but also the absolute time in the timetable, i.e., a node in the copy corresponding to day  $i$  is assigned time  $t = t_d + i \cdot 1440$ . In contrast to the previous (original or extended) time-expanded models, there is an obvious one-to-one correspondence between (valid and consistent) connections and paths in the fully time-expanded digraph, which immediately yields the following.

**LEMMA 2.** *A shortest path in the original (resp. realistic) fully time-expanded digraph constitutes a solution to the simplified (resp. realistic) version of EAP when elementary connections are valid on specific traffic days only.*

Since the size of the fully time-expanded digraph is huge ( $N$  times the size of the time-expanded digraph), we consider now how we can avoid to maintain this huge graph explicitly. By construction, all edge costs in the fully time-expanded digraph are less than a day. Assume an application of Dijkstra's algorithm to the fully time-expanded digraph, and let  $t$  be the time associated with the first node in the priority queue. All other nodes in the priority queue have an associated time larger than or equal to  $t$  and less than  $t + 1440$ . This observation allows the simulation of Dijkstra's algorithm on the time-expanded digraph: the time-expanded digraph reflects the subgraph of the fully time-expanded graph induced by the nodes with associated time in the interval  $[t, t + 1440]$ , and

by ignoring all train edges that correspond to trains, which do not operate on the considered day. The simulation works as follows. Whenever a departure node  $v$  gets settled, we consider the day specified by the label of that node (i.e., by dividing its time label by 1440). If its corresponding elementary connection (departure) is not valid on that day, then the outgoing, edge of  $v$  is ignored. However, since  $v$  is settled, although it corresponds to a nonvalid connection, we consider  $v$  as untouched again by setting its distance label to infinity and reinserting it in the priority queue, because from that point on it is considered to be the copy of the node in the next day.

We can actually avoid resetting the distance labels of the settled nodes to infinity, by inserting in the priority queue only those nodes that correspond to valid connections for the specific day. To achieve this, the algorithm is modified as follows.

Call a transfer node *valid* if its corresponding departure is an elementary connection that is valid at the day specified by the label of the transfer node. Suppose that instead of using the first transfer node that has a departure time greater than or equal to the given starting time, we start Dijkstra's algorithm from the first such node that also corresponds to a valid elementary connection.

Let  $v_0, v_1, v_2, \dots, v_k$  be the different transfer nodes of the same station, such that they are connected with each other through the edges  $(v_i, v_{(i+1) \bmod k})$ ,  $0 \leq i \leq k$ . Let  $wt(e)$  be the cost of edge  $e$ . Node  $v_i$  has two outgoing edges,  $(v_i, d_{v_i})$  and  $(v_i, v_{(i+1) \bmod k})$ , where  $d_{v_i}$  is the corresponding departure node. When  $v_i$  gets settled, we know that the train connection of  $d_{v_i}$  is valid for the day that is specified by the label of  $v_i$ . Thus,  $(v_i, d_{v_i})$  is relaxed. Now, instead of relaxing the edge  $(v_i, v_{(i+1) \bmod k})$ , so as to perform an update to the label of  $v_{(i+1) \bmod k}$ , we do the following. Starting from node  $v_{(i+1) \bmod k}$ , we check the following two conditions: (a) whether its corresponding elementary connection is valid for the day that is specified by the label  $l_{(i+1) \bmod k} = l(v_i) + wt(v_i, v_{(i+1) \bmod k})$ , and (b) whether  $v_{(i+1) \bmod k}$  has not already been settled. If both the above conditions hold for  $v_{(i+1) \bmod k}$ , then we check whether the current label of  $v_{(i+1) \bmod k}$  needs to be updated by  $l_{(i+1) \bmod k}$ . If any of the above conditions does not hold, then we proceed with  $v_{(i+2) \bmod k}$ , setting  $l_{(i+2) \bmod k} = l_{(i+1) \bmod k} + wt(v_{(i+1) \bmod k}, v_{(i+2) \bmod k}) = l(v_i) + wt(v_i, v_{(i+1) \bmod k}) + wt(v_{(i+1) \bmod k}, v_{(i+2) \bmod k})$  and so on, until we either find a node  $v_j$ ,  $0 \leq j \leq k$ ,  $j \neq i$ , that satisfies the above conditions (in which case we check whether the label of  $v_j$  needs to be updated by  $l_j$ ), or we reach again  $v_i$ . In the latter case, no suitable transfer node has been found and so no other update takes place. The above procedure guarantees that the transfer nodes and departure nodes in the priority queue will only correspond to valid connections.

Now, if  $a$  is an arrival node and there exists an edge  $(a, v_i)$ , where  $v_i$  is a transfer node, then when  $a$  gets settled with label  $l_a$ , instead of trying to update the label of the corresponding transfer node  $v_i$ , we perform again the previous procedure for the transfer nodes, starting with label  $l_i = l_a + wt(a, v_i)$  for the node  $v_i$ . Node  $a$  has also an outgoing edge leading to a departure node. This edge will be relaxed only if the conditions mentioned above for a transfer node are also satisfied by that departure node.

Let us now discuss the correctness of the aforementioned approach. Given the fact that we start from a valid transfer node and that we can only insert a transfer node in the priority queue if it is valid, we ensure that all transfer nodes that get settled are valid. This means that the corresponding outgoing edge that leads to a departure node is relaxed only if that departure is valid at the specific day.

Exactly for these reasons, there is no longer the need to reset the label of a node to infinity. The next time this node will be considered, the potential distance label that could be given to it will be greater than (or equal to) the label that it had when it was settled, and since it was valid for that label, then the new label cannot correspond to a better path.

## 5.2 Time-Dependent Model

The model as defined in Section 3.2 or in Section 4.2 is inherently able to handle traffic days. For route edges, the cost is determined by the first elementary connection on that edge leaving later than the arrival time. Now, the cost is determined by the first edge leaving later than the arrival time that is valid on the day under consideration (the day is computed by dividing the arrival time by 1440).

## 6. THE MINIMUM NUMBER OF TRANSFERS PROBLEM

The graphs defined for the realistic version of the earliest arrival problem in both the time-expanded (Section 4.1) and the time-dependent (Section 4.2) approach can be used to solve the minimum number of transfers (MNT) problem with a similar method: edges that model transfers are assigned a cost of one, and all the other edges are assigned cost zero. In the time-expanded case, all incoming edges of transfer nodes have cost one, whereas in the time-dependent case the edges in the set  $D$  or  $\bar{D}$  (depending on whether there are constant or variable transfer times at the station), except those belonging to the departure station, are assigned cost one, and all other edges have cost zero. Note that the edge costs in the time-dependent digraph are all static in this problem. The correctness of this modeling for finding MNT-optimal connections as shortest paths can be easily verified as in the proof of Theorem 1. This establishes the following.

**THEOREM 6.** *A shortest path in the realistic time expanded (resp. train route) digraph, with the edge costs given above, from a node belonging to (resp. representing) the departure station to a node belonging to (resp. representing) the arrival station is a solution to the MNTP.*

## 7. BICRITERIA OPTIMIZATION PROBLEMS

We consider bicriteria optimization problems with the earliest arrival (EA) and the minimum number of transfers (MNT) as the two criteria. We investigate three problem variants: (1) finding all Pareto-optimal solutions (the Pareto-curve); (2) finding the solution that minimizes one criterion while retaining the second below a given threshold; (3) finding the lexicographically first

Pareto-optimal solution (e.g., find among all connections that minimize EA the one with minimum number of transfers). Variant (2) is investigated only in the time-dependent model, since it is used as an intermediate step to solve variant (1). In the following, we parameterize the bicriterion problem we consider by  $(X, Y)$ , where  $X$  (resp.  $Y$ ) is the first (resp. second) criterion we want to optimize and  $X, Y \in \{EA, MNT\}$ .

### 7.1 Time-Expanded Model

We use the realistic time-expanded digraph (Section 4.1) to find the lexicographically first Pareto-optimum as well as all Pareto-optimal solutions.

**7.1.1 Lexicographically First Pareto-optima.** We first consider the  $(EA, MNT)$  case. Every edge  $e$  of the realistic time-expanded digraph is now associated with a pair of costs  $(a, b) = (EA(e), MNT(e))$ , where  $EA(e)$  is the cost of  $e$  when solving the realistic version of EAP (Section 4.1) and  $MNT(e)$  is the cost of  $e$  when solving MNTP (Section 6). Define on these cost pairs  $(a, b)$  the canonical addition, i.e.,  $(a, b) + (a', b') = (a + a', b + b')$ , and the lexicographic comparison, i.e.,  $(a, b) < (a', b') \Leftrightarrow (a < a') \text{ or } (a = a' \text{ and } b < b')$ . To find the lexicographically first  $(EA, MNT)$  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  $(0, 0)$ . The optimal solution is found when a node at the destination station is settled for the first time during the execution of the algorithm. The  $(MNT, EA)$  case is symmetric to the above and can be similarly solved. The proof of Theorem 1 can be easily adopted to establish the following.

**THEOREM 7.** *A shortest path in the realistic time-expanded digraph using cost pairs associated with its edges as defined above constitutes a solution to the problem of finding the lexicographically first Pareto-optimal connection (among all connections that minimize the first criterion, the one with minimum value in the second criterion). The actual path can be found by Dijkstra's algorithm in time  $O(m_e + n_e \log n_e)$ , where  $n_e$  (resp.  $m_e$ ) is the number of nodes (resp. edges) of the realistic time-expanded digraph.*

**7.1.2 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 consider the realistic fully time-expanded digraph used in Section 5, then every node in the time-expanded graph can have only one Pareto-optimum. Hence, we can compute for each node of the destination station the shortest path according to the cost pairs  $(a, b) = (EA(e), MNT(e))$  with the canonical addition and the lexicographic comparison. This is done as follows. When the first node of the destination station is settled, we have found the first  $(EA, MNT)$ -Pareto-optimal connection. We then let Dijkstra's algorithm continue; whenever a node of the destination station is settled with a smaller number of transfers than in any of the already found Pareto-optimal solutions, a new Pareto-optimal connection (which corresponds to the shortest path to that node) is found. The algorithm can be stopped when the Pareto-optimal solution

with the lowest possible number of transfers (i.e., the solution of MNTP) is found.

**THEOREM 8.** *All Pareto-optimal solutions for the two criteria EA and MNT can be enumerated during one run of Dijkstra’s algorithm in the realistic fully time-expanded digraph using cost pairs associated with its edges as defined above in time  $O(N(m_e + n_e \log(Nn_e)))$ , where  $n_e$  (resp.  $m_e$ ) is the number of nodes (resp. edges) of the realistic time-expanded digraph and  $N$  is the number of traffic days a timetable is valid.*

**PROOF.** It suffices to prove that each node of the destination station in the fully time-expanded digraph has only one Pareto-optimal solution. In that graph, every node  $v$  is associated with an absolute time value  $t(v)$  (not only the time of the day). Thus, every  $v_1$ - $v_2$  path has cost  $l = t(v_2) - t(v_1)$  according to the EA criterion, and, if  $k$  is the minimum number of transfers from  $v_1$  to  $v_2$ , then  $(l, k)$  is the only Pareto-optimal solution for  $v_2$ , since all other solutions have the same EA cost  $l$  and an equal or larger than  $k$  number of transfers. The time bound follows by the fact that the realistic fully time-expanded digraph is  $N$  times the size of the realistic time-expanded digraph.  $\square$

Now, similarly to Section 5.1, the realistic fully time-expanded digraph does not need to be maintained explicitly. The above algorithm can again be simulated on the realistic time-expanded digraph. The simulation is identical to that described in Section 5.1, with the exception of the procedure that prevents insertion in the priority queue of any node that corresponds to a nonvalid elementary connection. This procedure needs a slight modification, which is described next.

Let the label of a node  $v$  that is settled be  $(t_v, \tau_v)$ , where  $t_v$  is the time attribute and  $\tau_v$  is the number of transfers performed, and let  $(u, v)$  be an incoming edge of  $v$ . Assume that at some subsequent point of the algorithm node  $u$  is settled, and, consequently,  $(u, v)$  is relaxed resulting on a new possible label  $(t'_v, \tau'_v)$  for  $v$ . We know that  $t'_v \geq t_v$ . If  $\tau'_v < \tau_v$ , then we reinsert node  $v$  in the priority queue with label  $(t'_v, \tau'_v)$  and continue the algorithm, as if the node has not been settled (and storing label  $(t_v, \tau_v)$  as Pareto-optimum if  $v$  belongs to the destination station). This implies that the conditions mentioned in Section 5.1, regarding when to update the label of a transfer or departure node  $v$  with a new label  $(t'_v, \tau'_v)$ , has to change as follows: (a)  $v$  is valid for the day specified by  $t'_v$ ; and (b)  $v$  is not settled or  $v$  has been settled but with a number of transfers greater than  $\tau'_v$ . Again, we terminate the algorithm when a solution with transfer value equal to the solution of MNTP is found.

Note that the simulation changes the time for enumerating all Pareto-optimal solutions to  $O(Mm_e + MQn_e + Mn_e \log(Mn_e))$ , where  $M$  is the maximum number of transfers achieved by the solution of EAP and  $Q$  is the maximum number of transfer nodes within a station. This is because every node will be inserted in the priority queue, at most,  $M + 1$  times, every edge will be relaxed, at most,  $M + 1$  times, and because of the procedure, which prevents insertion in the priority queue of any node that corresponds to a nonvalid elementary connection, at most,  $Q$  stay edges may have to be examined, for

every node deleted from the priority queue, until the next valid transfer node is found.

## 7.2 Time-Dependent Model

We use the train-route digraph that models the realistic version of EAP (Section 4.2) to solve the three variants of the bicriteria optimization problems.

**7.2.1 Lexicographically First Pareto-optima.** We present an approach similar to that described for the time-expanded case, but which finds only the lexicographically first (MNT,EA) Pareto-optimal solution. Later, we will explain why it fails to do the same for the (EA,MNT) case.

A solution to the lexicographically first (MNT,EA) Pareto-optimum problem can be easily achieved, if instead of using a single value for the cost of an edge, we use a pair  $(a, b)$  of values, and define the canonical addition and lexicographic comparison to these pairs (exactly as in Section 7.1.1). The attribute values  $a, b$  of the pairs are updated separately. For the (MNT,EA) case,  $a$  is the MNT cost defined on  $\mathbb{N}$  (nonnegative integers), and  $b$  is the EA cost defined on  $T$  (set representing time). An edge  $e \in E$  has cost determined by a function  $h_e : (\mathbb{N}, T) \rightarrow (\mathbb{N}, T)$ , such that each attribute value is determined by the corresponding edge function.

**THEOREM 9.** *A shortest path in the train-route digraph using cost pairs associated with its edges, as defined above, constitutes a solution to the problem of finding the lexicographically first (MNT,EA) Pareto-optimal connection. The actual path can be found in time  $O(m_d \log W + n_d \log n_d)$ , where  $n_d$  (resp.  $m_d$ ) is the number of nodes (resp. edges) of the train-route digraph, and  $W$  is the maximum number of elementary connections associated with an edge.*

**PROOF.** To prove the correctness, it suffices to show that the new edge cost function  $h_e$  is nondecreasing and with nonnegative delay. Consider the modeling of the constant transfer cost case, and let  $\tau, \tau_1, \tau_2 \in \mathbb{N}$  and  $t, t_1, t_2 \in T$ .

- If  $e \in I$ , then  $h_e(\tau, t) = (\tau, t)$ . If  $(\tau_1, t_1) \leq (\tau_2, t_2)$ , then  $h_e(\tau_1, t_1) \leq h_e(\tau_2, t_2)$ .
- If  $e \in D$ , then  $h_e(\tau, t) = (\tau + 1, t + x) \geq (\tau, t)$ , where  $x \geq 0 \in T$  is the transfer time at the station that  $e$  belongs to. If  $(\tau_1, t_1) \leq (\tau_2, t_2)$ , then  $h_e(\tau_1, t_1) = (\tau_1 + 1, t_1 + x) \leq (\tau_2 + 1, t_2 + x) = h_e(\tau_2, t_2)$ .
- If  $e \in R$ , then  $h_e(\tau, t) = (\tau, f_e(t)) \geq (\tau, t)$ , since  $f_e$  has nonnegative delay. If  $(\tau_1, t_1) \leq (\tau_2, t_2)$ , then
  - if  $\tau_1 < \tau_2$ , then  $h_e(\tau_1, t_1) = (\tau_1, f_e(t_1)) < (\tau_2, f_e(t_2)) = h_e(\tau_2, t_2)$ , and
  - if  $\tau_1 = \tau_2 = \tau$  and  $t_1 \leq t_2$ , then  $h_e(\tau_1, t_1) = (\tau, f_e(t_1)) \leq (\tau, f_e(t_2)) = h_e(\tau_2, t_2)$ .

The case with variable transfer cost can be similarly proved. The time bound follows by that of Theorems 4 and 5.  $\square$

In contrast to the time-expanded case, the symmetric problem of finding a lexicographically first (EA,MNT) Pareto-optimal solution cannot be solved by just using appropriate edge cost pairs on the train-route digraph, as in Section 7.1.1. To show that such an approach may fail, consider the train-route

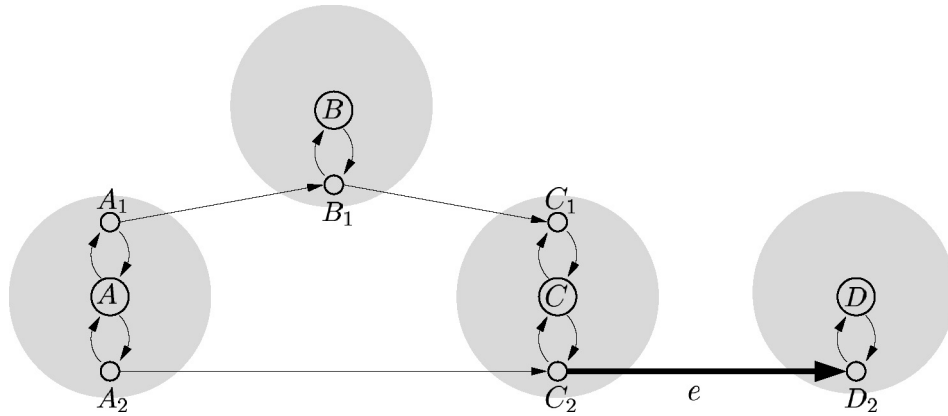


Fig. 5. Lexicographically first (EA,MNT) Pareto-optimal connections cannot be found by simply using pairs as edge costs in the train-route digraph.

digraph shown in Figure 5 with a query to find the lexicographically first (EA,MNT) Pareto-optimal connection from  $A$  to  $D$ . Assume there is a train  $T_1$  on the first route from  $A$  to  $C$  via  $B$ , departing from  $A$  at 9:00,<sup>4</sup> arriving (resp. departing) at (resp. from)  $B$  at 9:10 (resp. 9:11), and arriving to  $C$  at 9:20. Let now  $T_2$  be another train on the second route from  $A$  to  $D$  via  $C$ , departing from  $A$  at 9:00, arriving (resp. departing) at (resp. from)  $C$  at 9:30 (resp. 9:31), and arriving to  $D$  at 10:00. Assume that the transfer time at  $C$  is 5 min, and that the given departure time at  $A$  is 9:00. Consider, now, an application of the time-dependent shortest-path algorithm with edge cost pairs  $(a, b)$ , where  $a$  is the EA cost and  $b$  is the MNT cost starting at  $A$ . At some point, the algorithm will relax the edges  $(A_1, B_1)$ ,  $(B_1, C_1)$ ,  $(C_1, C)$ , and  $(A_2, C_2)$ , giving labels  $(9:10, 0)$  to  $B_1$ ,  $(9:20, 0)$  to  $C_1$  and  $C$ , and  $(9:30, 0)$  to  $C_2$ . As a result of the relaxation of edge  $(C, C_2)$ , the label of  $C_2$  becomes  $(9:25, 1)$ , since  $(9:25, 1) < (9:30, 0)$ . Consequently, the relaxation of edge  $e = (C_2, D_2)$  will have label  $h_e(9:25, 1) = (10:00, 1)$  and output a connection with that destination label, i.e., a connection using trains  $T_1$  and  $T_2$  with one change at station  $C$ . However, this is not the lexicographically first (EA,MNT) Pareto-optimal connection, since the connection that uses only train  $T_2$  yields a lexicographically smaller destination label  $(10:00, 0)$ . The algorithm fails to find the optimal solution, because the time-dependent function  $h_e$ , in this case, is *not* nondecreasing: we have that  $(9:25, 1) < (9:30, 0)$ , but  $h_e(9:25, 1) = (10:00, 1) > (10:00, 0) = h_e(9:30, 0)$ .

**7.2.2 Earliest Arrival with Bounded Number of Transfers.** Given two stations  $F$  and  $H$ , 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 and consistent connection from  $F$  to  $H$  such that the arrival time at  $H$  is the earliest possible, and subject to the additional constraint that the total

<sup>4</sup>For convenience, we use the usual “hour:minutes” format for time.

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.

In this section, we describe two algorithms for solving the EABT problem. The first one is an adaptation of the graph-copying method proposed in Brodal and Jacob [2004] to our realistic time-dependent model (train-route digraph). The second one is an adaptation of the labeling approach (see e.g., [Ziegelmann 2001]) for solving resource-constraint shortest paths to our realistic time-dependent model.

The idea of Brodal and Jacob [2004] adapted to our realistic time-dependent model is as follows. We construct a new digraph  $G' = (V', E')$  consisting of  $k + 1$  levels. Each level contains a copy of the train-route digraph  $G = (V, E)$ , where  $E = I \cup D \cup \overline{D} \cup R$  (cf. Section 4.2). 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  $(u, v) \in I \cup R$ , we place in  $E'$  the edges  $(u_i, v_i)$ ,  $\forall 0 \leq i \leq k$ . For each edge  $(u, v) \in D \cup \overline{D}$ , 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 verify that a path from some node  $s_0$  (copy of  $s$  at the 0th level) to a node  $x_l$  (copy of  $x$  at the  $l$ th level) represents a path from  $station(s)$  to  $station(x)$  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  $F = station(s)$ , 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  $H = station(u)$ . Let  $n_d = |V|$ ,  $m_d = |E|$ , and let  $W$  be the maximum number of elementary connections associated with an edge  $e \in E$ . Since  $G'$  consists of  $k + 1$  copies of  $G$  and since during the shortest-path computation in  $G'$  a binary search has to be performed when an edge is relaxed to determine its cost, it follows that an application of Dijkstra's algorithm on  $G'$  for solving EABT takes  $O(m_d k \log W + n_d k \log(n_d k))$  time.

The adaptation of the labeling approach to our train-route digraph  $G = (V, E)$  is as follows. We use the modified Dijkstra's algorithm (cf. Section 4.2), where we now 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  $(t_l, l)_u$ ,  $0 \leq l \leq k$ , representing the currently best time  $t_l$  to reach node  $u$  by performing exactly  $l$  transfers.

Let  $s$  be the node for which  $F = station(s)$ . 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, we then delete from the priority queue all labels of the form  $(t_r, r)_u$  for  $l < r < l'$ , 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  $(t_r, r)_u$  it holds that  $t_l \leq t_r$  (as  $(t_l, l)_u$  was extracted before  $(t_r, r)_u$ ) and  $l < r$ . 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  $(t_r, r)_u$ , 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 be safely deleted is  $O(k)$ , while the total number of deletions is  $O(n_d k)$ , where  $n_d = |V|$ . This is because of the fact that we only check the labels from the current number of transfers (given by the ultimate *delete-min* operation) until its previous value (given by the penultimate *delete-min* operation). 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 is  $O(m_d k)$ , resulting in a total time of  $O(m_d k \log W)$  for relaxations, where  $m_d = |E|$  and  $W$  is the maximum number of elementary connections associated with an edge  $e \in E$ . We can also see that the total number of labels that is in the priority queue is, at most,  $O(n_d k)$ . Because of this, the time for a *delete-min* or a *delete* operation is  $O(\log(n_d k))$ . This means that the total time needed for the algorithm is  $O(m_d k \log W + n_d k \cdot \log(n_d k))$ , which is asymptotically the same as the previous one. The discussion in this section establishes the following.

**THEOREM 10.** *The earliest-arrival with bounded number of transfers problem can be solved in the extended time-dependent model in time  $O(m_d k \log W + n_d k \log(n_d k))$ , where  $n_d$  (resp.  $m_d$ ) is the number of nodes (resp. edges) of the train-route digraph,  $W$  is the maximum number of elementary connections associated with an edge, and  $k$  is the bound in the number of transfers.*

**7.2.3 All Pareto-optima.** The solution to EABT can be used to generate all Pareto-optimal solutions in the time-dependent model. In particular, the following approach finds all Pareto-optima. First, solve EAP and count the number of transfers found, say  $M$ . Then, run the labeling algorithm of Section 7.2.2 for solving EABT for all values  $M - 1, M - 2, \dots, 0$  of transfers. Note that the labeling algorithm can actually be used to speedup the process of solving  $M$  EABT problems. Recall that the algorithm maintains for each node  $M + 1$  labels, where label  $0 \leq i \leq M$  stores the best EA solution performing exactly  $i$  transfers (provided that such a solution exists), and discards dominated paths. Hence, instead of stopping the algorithm when the optimal solution with, at most,  $M - 1$  transfers at the destination is found and repeat with the next transfer bound of  $M - 2$ , we can just continue with the execution of the algorithm to produce the next solution with, at most,  $M - 2$  transfers, considering the bound  $k$  being equal to  $M - 2$ , and so on, until no new path can be found. Note that during the above process, we can discard previously found paths, which are dominated by subsequent solutions. The above discussion is summarized as follows.

**THEOREM 11.** *The modified labeling algorithm can enumerate all Pareto-optimal solutions for the two criteria EA and MNT in the extended time-dependent model in  $O(m_d M \log W + n_d M \log(n_d M))$  time, where  $n_d$  (resp.  $m_d$ )*

is the number of nodes (resp. edges) of the train-route digraph,  $W$  is the maximum number of elementary connections associated with an edge, and  $M$  is the maximum number of transfers achieved by the solution of EAP.

## 8. HEURISTICS FOR SPEEDING-UP QUERY TIME

Previous experimental studies with the time-expanded model [Schulz et al. 2000, 2002; Wagner et al. 2006] have shown that heuristic improvements can considerably speedup performance. These techniques allow for generation of additional data during preprocessing that can be used in the on-line phase to speedup the algorithm. Since one of our goals is to investigate the practical performance of the time-dependent model and its extensions, and, in particular, in comparison to the time-expanded model, we have considered several heuristics for both models. We focus on a general heuristic (goal-directed search) that can be applied to both models and also to model specific speedup heuristics. Some of the heuristics have been introduced elsewhere, while some others are new. Since all of them are considered in our experimental study, in the rest of the section we briefly review the former, while we present the latter in more detail.

### 8.1 Time-Expanded Model

**8.1.1 Goal-Directed Search.** The most natural heuristic to consider is the goal-directed search or the method of potentials (see e.g., [Lengauer 1990]), in order to exploit the geometric information associated with the nodes (coordinates of stations). In this heuristic the cost of every edge is modified in a way that if the edge points toward the destination its new cost gets smaller, while if the edge points away from the destination node, then its new cost gets larger. More precisely, for an edge  $(u, v)$  with cost  $wt(u, v)$ , its new cost  $wt'(u, v)$  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 nonnegative.

This heuristic was considered in Schulz et al. [2000] 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,<sup>5</sup> 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.

**8.1.2 Omitting Nodes.** There is a simple way to reduce the size of the graphs in the time-expanded model, based on the fact that there are a lot of nodes with out-degree one. Any path through such a node must continue to the head of the single outgoing edge. Thus, we can safely delete such nodes from the graph and redirect the incoming edges to the head of the single outgoing edge. The cost of a redirected edge is the sum of the incoming edge plus the cost of the outgoing edge.

<sup>5</sup>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 speeds of elementary connections.

In the original time-expanded digraph, we omit arrival nodes except for those in the destination station. Since the latter are only known when a query is issued, omitting nodes is accomplished as follows. We deactivate (mark as deleted) all arrival nodes in the graph. When, during the execution of the algorithm for answering the query, an edge leading to a node in the destination station is relaxed, the corresponding arrival node is activated. In the realistic time-expanded digraph we omit departure nodes, and, hence, no particular treatment is required at query time.

For the original time-expanded digraph, this technique yields a graph of roughly one-half the original size, since all arrival nodes have out-degree one. In the realistic time-expanded digraph, every departure node has out-degree one and, hence, this graph is reduced by about one-third.

## 8.2 Time-Dependent Model

**8.2.1 Goal-Directed Search.** The goal-directed search heuristic described in Section 8.1.1 can be also applied in the time-dependent model. However, preliminary experiments with exactly the same technique showed no improvement of the running time in the time-dependent model. Therefore, we had to invent some new variant of the goal-directed search method to improve the running time. Our new variant uses: (1) a different potential function, which depends on the destination station; (2) Manhattan, besides Euclidean, distances between stations; and (3) integral potentials to avoid expensive floating-point operations. The details are as follows.

As in Section 8.1.1, for an edge  $(u, v) \in E$  with cost function  $wt(u, v, \tau)$ , where  $\tau$  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

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

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  $\lambda_t$  is called the *scaling factor* and is a nonnegative number (which can be different for each destination node  $t$ ) that is used to scale the distances so that  $wt'(u, v, \tau)$  is nonnegative. 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  $\lambda_t$  corresponds to the inverse of the maximum speed considered in Section 8.1.1; however, in Section 8.1.1 the maximum speed is the same for all destinations  $t$ , while  $\lambda_t$  depends on  $t$ . Potentials  $p[\cdot]$ , as defined above, are valid as the next lemma shows.

**LEMMA 3.** *The function  $p[\cdot]$  is a valid potential function for the goal-directed search, i.e.,  $wt'(u, v)$  is nonnegative for each edge  $(u, v)$ .*

**PROOF.** 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  $d(\alpha, t) - d(\beta, t) > 0$ , since otherwise there would be no way reaching  $t$  – the distance to

get there from any 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

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

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

From Eq. (2) we get

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

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  $wt'(u, v) = wt(u, v) - \lambda_t[d(u, t) - d(v, t)] \geq wt(u, v) \geq 0$ .  $\square$

Even if all edge cost 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 lemma shows.

**LEMMA 4.** *If, for the nonnegative 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,  $0 < \lfloor a \rfloor - \lfloor b \rfloor < fr(b) - fr(a)$ . However,  $fr(b) - fr(a) < 1$  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.  $\square$

Consideration of the integral parts of the floating-point potentials turned out to be rather beneficial in certain cases, as our experiments show.

**8.2.2 Avoiding Binary Search.** For the original time-dependent model, we also considered the heuristic that avoids the binary search as described in Brodal and Jacob [2004]. This speedup technique assumes that there exists a single timetable with no exception in which day a train operates. The idea is as follows. Let  $k$  be the out-degree of a node  $v$  in the time-dependent digraph and let  $(v, u_1), \dots, (v, u_k)$  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.

**8.2.3 Further Speedup When Modeling with Train Routes.** In order to quickly find the most suited connection for each edge, we need to store the information efficiently. In this section, we elaborate on implementation issues regarding the efficient storage and retrieval of information concerning the computation of edge costs.

From the input we know that every edge  $(x, y) \in E$  is associated with a timetable. The entries of the timetable are stored lexicographically, i.e., if  $(d_1, a_1), (d_2, a_2)$  are two such entries, then  $(d_1, a_1) < (d_2, a_2)$  iff  $d_1 < d_2$ , or  $d_1 = d_2$  and  $a_1 < a_2$ . Recall that for each edge  $(x, y) \in E$ , there is a function  $f'_{(x,y)} : T \rightarrow T$ , which determines the cost of the edge. In this way, before relaxing the outgoing edges of a node, we need to call the corresponding function  $f'$  to compute the cost of the edge at the specified time.

Consider the case of variable transfer costs within the same station. Note that the different time needed for each transfer at some station between the same train routes is caused by the fact that the trains that perform the same route do not stop every time at the same platform of the station. The input regarding the time for the transfers takes into consideration the platforms at which the trains stop. This means that for every event  $e$  there must be some information about the first train of every other train route that can be followed. As a result, for the case of variable transfer costs, each edge  $(p_i^u, p_i^u) \in \overline{D}_u$ ,  $u \in S$ , has a timetable with one entry for each possible arrival time to  $p_i^u$ . Because of this, all edge functions  $f'$  must perform a search in order to find the most suitable entry in the edge's timetable. To avoid the search, we can store additional information at the edges  $(x, y) \in R$ , instead of storing a timetable at each edge  $(x, y) \in \overline{D}$ .

Let  $(p_j^v, p_i^u), (p_i^u, p_i^w) \in R$  be two edges which belong to the same route and let  $\mathcal{T}_{(p_j^v, p_i^u)}, \mathcal{T}_{(p_i^u, p_i^w)}$  be their corresponding timetables. Also, let  $s$  be the station that  $p_i^u$  belongs to. For each event (entry)  $e \in \mathcal{T}_{(p_j^v, p_i^u)}$  we store a pointer *same\_train<sub>e</sub>* to the event  $e'$  of  $\mathcal{T}_{(p_i^u, p_i^w)}$  such that  $e, e'$  are performed successively by the same train. (We assume that the first train that leaves  $p_i^u$  after the arrival time of  $e$  is the same train that  $e$  belongs to.) In this way when we need to relax the only

outgoing edge of  $p_i^u$  that belongs to  $R$  (no transfer is performed), we just need to follow the *same\_train<sub>e</sub>* of the event  $e$  that brought us to that node.

Now, we consider the case where we need to relax an outgoing edge  $(p_i^u, p_{i'}^u) \in \overline{D}$  of  $p_i^u$ . Let  $(p_{i'}^u, p_{i''}^{w'})$ , for  $p_{i''}^{w'} \in V$ , be the outgoing edge of  $p_{i'}^u$  that belongs to  $R$ , and  $\mathcal{T}_{(p_{i'}^u, p_{i''}^{w'})}$  its respective timetable. Instead of storing a timetable for  $(p_i^u, p_{i'}^u)$ , we store for each  $e \in \mathcal{T}_{(p_i^u, p_{i'}^u)}$  a pointer *very\_next<sub>e, i'</sub>*, to the event  $e'$  of  $\mathcal{T}_{(p_{i'}^u, p_{i''}^{w'})}$  such that  $e'$  is the first event of  $(p_{i'}^u, p_{i''}^{w'})$  that can be followed given that  $p_i^u$  was reached by use of  $e$ . Having one such pointer *very\_next<sub>e, i'</sub>* for all  $p_i^u$ , we can relax at once the outgoing edges of all nodes of station  $s$  belonging to  $R$ , and thus avoid the relaxation of all other edges (not in  $R$ ) among those nodes. This means that we no longer need to store any other information regarding these latter edges. Note also that the *same\_train<sub>e</sub>* pointer is *very\_next<sub>e, i</sub>* so *same\_train* pointers need not be stored explicitly either.

For the case of constant transfer time at each station, we can avoid some of the binary searches when solving one of the problems by expanding, if necessary, the graph a little more. Suppose that when the trains are separated into train routes at the construction of the graph, we impose the additional constraint that for each train route the events that belong to the same train of the route are always indexed by the same number at the (sorted) timetables of the (timetable) edges of the corresponding route. Suppose, also, that at each route node  $u$ , an index *next\_departure<sub>u</sub>* is maintained. If during the execution of the algorithm  $u$  is reached by its incoming (if any) route edge, then *next\_departure<sub>u</sub>* is set to the index of the event that was used on the incoming edge. If, on the other hand,  $u$  was reached through a transfer edge, then *next\_departure<sub>u</sub>* is set to  $-1$ . In this way, when  $u$  is extracted from the priority queue, if *next\_departure<sub>u</sub>* is nonnegative, the corresponding event can be used to relax its outgoing route edge, without the need to perform a binary search. If not, a binary search must be performed. Moreover, a negative *next\_departure<sub>u</sub>* value means that the shortest path from the source node to  $u$  passes through the central node of  $u$ 's station. Thus, the outgoing (transfer) edge of  $u$  that leads to the central node does not need to be relaxed.

## 9. EXPERIMENTS

The main goal of our experimental study is to compare, in practice, the performance of the time-expanded and the time-dependent approach, both in their original and in their extended versions.

Given two different implementations and a timetable, we define the *relative performance* or *speedup* 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 speedup achieved when the time-dependent approach is used instead of the time-expanded approach.

All of our 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 512 MB of memory running Linux (kernel

Table I. Parameters of the Considered Graphs for Each of the Timetables<sup>a</sup>

Timetable	Time-Expanded		Time-Dependent			
	Nodes	Edges	Nodes	Edges	Elem. Conn. per Node	Elem. 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

<sup>a</sup>The two columns on the left show the size of the graph used in the time-expanded model with the optimization described in Section 8.1.2. 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.

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.

### 9.1 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. For both models we also investigate the heuristics described in Section 8.

**9.1.1 Data.** We have used real-world data from the German and French railways. In particular, the following five timetables were used. The first timetable contains French long-distance railway 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 railway traffic in Germany (`ger-longdist`), two contain local traffic in Berlin/Brandenburg (`ger-local1`) and in the Rhein/Main region (`ger-local2`), and the last is the union of all these three German timetables (`ger-all`). We would like to note that Hafas [HAFAS], the timetable information system used by the German railway company Deutsche Bahn, is based on data of the same format. Table I shows the characteristics of the graphs used in these models for the above mentioned timetables.

Real-world queries were available only for the timetables `ger-longdist` and `ger-all`, so, in addition, we generated random queries for every timetable. Each query-set consists of 50,000 queries of the form departure station, destination station, and earliest departure time.

**9.1.2 Implementation Environment and Performance Parameters.** For the time-expanded model the implementation is based on that used in Schulz et al. [2000]; the optimization technique to ignore the arrival events described in Section 8.1.2 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 implementations (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 `ger-longdist` and `ger-all` we, in addition, performed the corresponding real-world queries) and measured the following performance parameters as *average 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 by binary search for calculating the edge costs.

**9.1.3 Results and Discussion.** Our experimental results are reported in Figure 6 and Tables II, III, and IV. The reported results clearly show that the time-dependent model solves the simplified earliest-arrival problem considerably faster than the time-expanded model, for every considered data set. Regarding CPU time, the speedup ranges between 12 (`france`) and 40 (`ger-local2`) when the basic implementations are used (see Figure 6 and Table II), and between 17 (`france`) and 57 (`ger-local2`) when comparison concerns the best implementations (including heuristics) in both models (see Tables III and IV).

Concerning the time-dependent model, we observe that it is better to use the “avoid binary search” technique (see Tables III and IV). Compared to the binary search implementation, the speedup 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 paid 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 because of the additional computations required to calculate the edge costs 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,<sup>6</sup> and thus bad performance of the goal-directed search technique.

## 9.2 Comparison of Realistic Models

In this section, we investigate how the time-expanded and the time-dependent approaches compare when the realistic versions of the problems and models are considered. As input data, we used a modification of the `ger-longdist` timetable, which we refer to as `ger-longdist1`. Since train changes are considered here, we only used the trains of the timetable that operate on one specific

---

<sup>6</sup>In both models, the potentials are inversely proportional to the maximum speed; for the time-expanded case this is clear by definition; for the time-dependent case, see the definition of the factor  $\lambda_t$ .

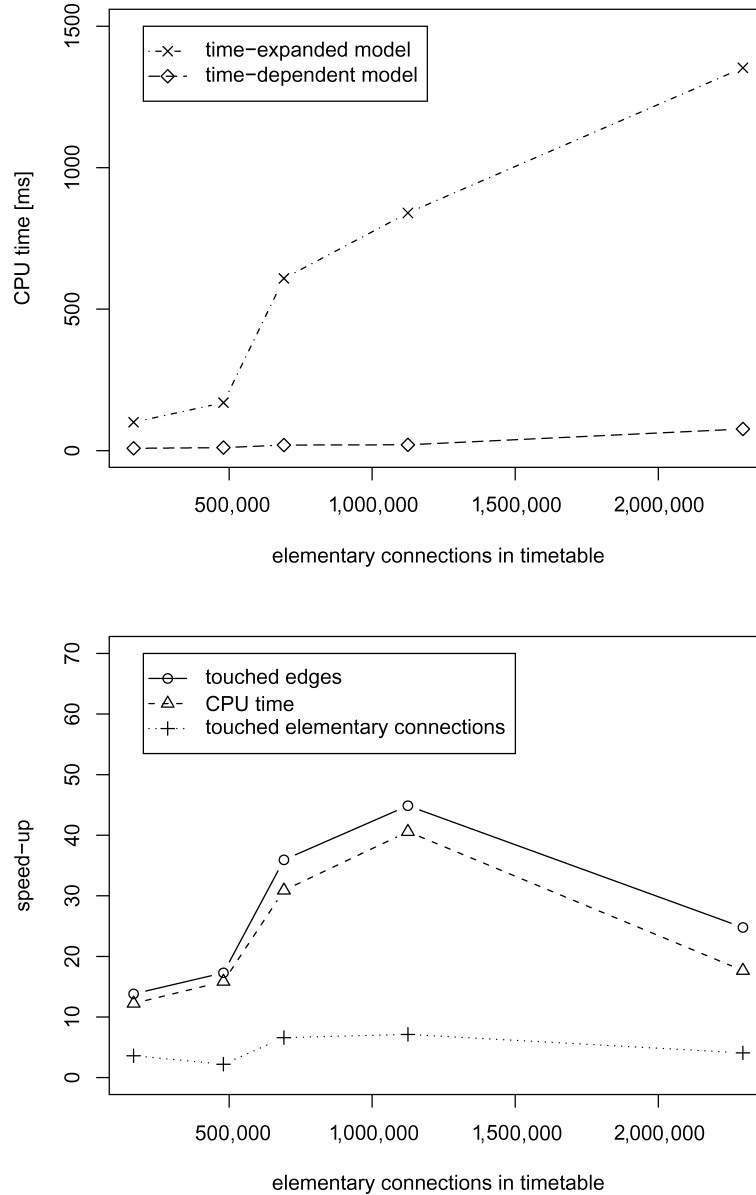


Fig. 6. Performance of the basic implementations of the time-expanded and time-dependent models for the simplified earliest-arrival problem (no goal-directed search, binary search in the time-dependent model) regarding the five timetables and the random queries. Each point represents the average over 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 speedup with respect to the number of touched edges, CPU time, and number of touched elementary connections is shown.

Table II. Average CPU Time and Operation Counts for Solving a Single Query for the Time-Expanded (Upper Part) and the Time-Dependent Model (Lower Part)<sup>a</sup>

	Timetable	Real	Time [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

<sup>a</sup>The arrival nodes are omitted in the time-expanded model (see Section 8.1.2), 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.

day, whereas in the simplified case discussed before, we assumed that all trains operate daily. We used the modeling with constant transfer times at stations, since only the maximum possible transfer time for each station was available in the input data. We also used the real-world and random queries as described in Section 9.1.1. Table V shows the parameters of the graphs used in the realistic models compared to the original models.

**9.2.1 Implementation Environment.** For both approaches we implemented the described solutions for the realistic earliest-arrival (EA-realistic) problem (Section 4), the minimum number of transfers (MNT) problem (Section 6), the all Pareto-optima (All Pareto-Opt) problem involving EA and MNT as the two criteria (Section 7), the lexicographically first (MNT,EA) Pareto-optimum (Lex-F(MNT,EA)), and the lexicographically first (EA,MNT) Pareto-optimum (Lex-F(EA,MNT))—the latter only for the time-expanded model see Section 7.2.1). Moreover, for the time-dependent model, we have also considered the two algorithms for solving the earliest arrival with bounded number of transfers problem (Section 7.2.2): the one based on the graph-copying approach (EABT-BJ) and the one based on the labeling approach (EABT-L).

In the time-expanded implementations we again reduced the node set by omitting the departure nodes in the realistic time-expanded graph (see Section 8.1.2). In the realistic time-dependent implementations, we also applied the heuristic described in Section 8.2.3.

**9.2.2 Results.** Table V (lower part) shows the comparison between the problems solved in both realistic versions of the time and the time-dependent approaches. The key parameters—number of touched nodes, edges, and average running time—are displayed for the real-world queries. More details, as well as results on random queries and other problems, are shown in Table VI.

Table III. Comparison of the Time-Dependent Implementations that Use Binary Search and Four Different Versions of Goal-Directed Search: Euclidean Distance with Integer and Float Potentials and Manhattan Distance with Integer and Float Potentials<sup>a</sup>

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

<sup>a</sup>Columns are as in Table II.

Concerning CPU time, the results show that the time-dependent approach still performs better than the time-expanded approach in all cases considered. However, the gap for the realistic EAP is not as large as it was for the simplified EAP; in fact, for real-world queries the speedup is now only 1.5. Moreover, if one considers the total number of touched edges, for real-world queries the speedup is smaller than 1 (i.e., the time-expanded approach is better). However, this does not affect the running time, since less than one-half of the total touched edges require extra operations for computing their costs (timetable edges touched; see Table VI, lower part) and the rest has a fixed cost (transfer edges touched).

For the other problems considered, the time speedup ranges from 1.5 to 3.3, while the touched-edges speedup is 1.6, in all cases. In particular, for the MNT problem, the extended time-dependent model (train-route digraph) is clearly superior to the extended time-expanded model (realistic time-expanded digraph); the speedup with respect to CPU time is 3.3 for real-world queries and 4.5 for random queries. A similar observation holds for the lexicographically

Table IV. 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)<sup>a</sup>

		Time-expanded Model				
	Timetable	Real	Time [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
<hr/>						
		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	3 618	1272	2041
	ger-all	×	19.1	17088	5615	9524

<sup>a</sup>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 II.

first (MNT,EA) Pareto-optimum problem, where the CPU speedup is 1.9 for real-world queries and 2 for random queries, and for the all Pareto-optima problem, where the CPU speedup is 1.5 for real-world queries and 1.8 for random queries.

We would like to note that the solution to the problem of finding all Pareto-optima exhibits a quite stable behavior compared to EAP in both approaches. It is 3.6 times slower than EAP for real-world queries in both models and about the same factor slower when random queries are considered (3.3 in the time-expanded model and 4 times slower in the time-dependent model). This confirms the theoretical comparison of the complexity bounds between finding an EAP and an all Pareto-optima solution; that is, the all Pareto-optima solution is about  $M$  times slower than that of EAP solution, since for a single day, all connections are valid and, hence, the term  $MQn_e$  vanishes (i.e., the procedure for transfer nodes is not executed, since all outgoing edges correspond to valid

Table V. Graph Parameters and Main Results for Realistic Models, Applied to the Same Input Timetable `ger-longdist1`<sup>a</sup>

Graph	Time-expanded		Time-dependent			
	Nodes	Edges	Station Nodes	Route Nodes	Timetable Edges	Transfer Edges
Simplified	289432	578864	6685	—	17577	—
Realistic	578864	1131164	6685	79784	72779	159568

Problem	Time-expanded			Time-dependent		
	Nodes	Edges	Time [ms]	Nodes	Edges	Time [ms]
EA-realistic	40624	73104	78	44731	83662	50
MNT	101731	138417	125	26680	83173	38
Lex-F(MNT,EA)	99061	137075	161	28272	83363	83
All Pareto-Opt	123943	236887	287	78412	145444	181

<sup>a</sup>The upper part shows the number of nodes and edges of the graphs compared to the simplified, original models. The lower part contains the average values over all real-world queries of the performance parameters for the different problems solved.

connections). A closer look at our data reveals that the average  $\hat{M}$  over all values of  $M$  is 3.62 for real-world queries and 4 for random queries (the value of  $M$  for each query is the number of transfers achieved by the solution of EAP for that query).

Further observations regarding each model separately are as follows.

**9.2.2.1 Time-Expanded Model.** The graph used in the realistic EAP (Table VI) 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 (11% more time using real-world queries and 16% more using random queries). The lexicographically first (EA,MNT) Pareto-optimal problem is solved in a very similar way as the realistic EAP. It is interesting to observe that the CPU time, as well as the average number of nodes and edges touched, are almost identical. In contrast, the MNT, the lexicographically first (MNT,EA) Pareto-optimum, and the all Pareto-optima problems require more CPU time than the realistic EAP, since a much bigger part of the graph has to be explored.

**9.2.2.2 Time-Dependent Model.** Although the train-route digraph has approximately 13 times more nodes and edges than the digraph of the original time-dependent model (see Table V), the experimental results reported in Table VI show that the time required for solving the realistic EAP is only five times slower than that of the simplified EAP. This is because of the fact that the structure of the two graphs is quite different. While the original time-dependent digraph has only timetable edges, almost 70% of the train-route digraph edges have constant cost and the timetables of the other edges have much fewer events than their corresponding edges in the original time-dependent digraph.

The MNT problem is solved faster than the realistic EAP, since, in this case, all edge costs in the train-route digraph are static and, hence, no binary search is needed.

Table VI. Detailed Results for the Realistic Problems<sup>a</sup>

Time-expanded Model						
Problem	Real Queries	Time [ms]	Average Nodes	Average Edges		
EA-simplified	×	70	20760	41519		
EA-realistic	×	78	40624	73104		
MNT	×	125	101731	138417		
Lex-F(EA,MNT)	×	82	40628	73123		
Lex-F(MNT,EA)	×	161	99061	137075		
All Pareto-Opt	×	287	123943	236887		
EA-simplified		106	34469	61955		
EA-realistic		122	61159	111301		
MNT		212	169299	239841		
Lex-F(EA,MNT)		129	61195	111386		
Lex-F(MNT,EA)		259	163438	234297		
All Pareto-Opt		405	170946	330150		

Time-dependent Model						
Problem	Real Queries	Time (ms)	Nodes Touched	Timetable Edges Touched	Transfer Edges Touched	Bin. Steps per Timetable Edge
EA-simplified	×	10	2967	4365	–	3.126
EA-realistic	×	50	44731	38168	45494	0.319
MNT	×	38	26680	21558	61615	0
Lex-F(MNT,EA)	×	83	28272	22901	60462	0.178
EABT-L	×	79	39070	32093	39034	0.247
EABT-BJ	×	77	46127	37499	45250	0.261
All Pareto-Opt	×	181	78412	65753	79691	0.211
EA-simplified		11	3315	4811	–	3.005
EA-realistic		54	48200	41011	48942	0.311
MNT		47	33455	27235	69411	0
Lex-F(MNT,EA)		106	35262	28779	69054	0.172
EABT-L		104	49179	40638	49768	0.242
EABT-BJ		107	60493	49392	59906	0.254
All Pareto-Opt		219	92378	77610	94904	0.204

<sup>a</sup>The upper table concerns the time-expanded implementations; the lower table the time-dependent ones. All results are based on the `ger-longdist1` timetable. For comparison with the original models, we have included the results for the simplified version of the earliest-arrival problem (EA-simplified).

The solution of the lexicographically first (MNT,EA) Pareto-optimum problem involves (again) time-dependent edge costs and is slower than both MNTP and the realistic EAP, even though its performance parameters (number of nodes and edges touched) are similar to those of MNTP. This is because of the fact that in this problem more computations are needed in order to maintain the priority heap. It is interesting to observe that the algorithm for solving the lexicographically first (MNT,EA) Pareto-optimum problem performs roughly one-half of the binary steps performed by that of solving the realistic EAP. This can be explained as follows. While solving this problem, the first connections to be considered are the ones with no transfers (i.e., those that belong to the train routes that pass through the departure station); the next connections will be those with only one transfer, and so on, thus avoiding to perform a transfer

as long as possible. Now, the implementation does not use binary search at a timetable edge, unless the source of the edge has been reached through a transfer edge of the same station. In addition, the time of this problem is greater than the time of EAP, since, in this case, the comparison of two labels may demand comparisons between two pairs of values instead of just two values.

Concerning the EABT problem, we can see that even though the labeling approach (EABT-L) touches much fewer nodes and edges than the graph-copying approach (EABT-BJ), both algorithms require practically the same time. This can be explained as follows. The labeling approach requires some additional information in order to efficiently manage the  $k$  labels at each node and easily perform the deletion of the labels that have become obsolete. The space needed for this information is linear in the number of nodes. On the other hand, the graph-copying approach uses storage space linear in the product of  $k$  and the size of the graph (since it produces  $k + 1$  copies of the graph), while both approaches require the same space for storing the priority queue. This means that when  $k$  becomes quite large, the graph-copying approach will need much more storage and time than the labeling approach. In the experiments conducted, the average  $\hat{k}$  over all values of  $k$  was rather small, with a value of 2.66 for real-world queries and 2.99 for random queries (we took as the value of  $k$  for each query the average of the minimum and maximum number of transfers required for the query). Consequently, the running times of the two approaches are similar. If the value of  $k$  was larger, then the running time of the labeling approach would be smaller than the one of the graph-copying approach.

The algorithm for enumerating all Pareto-optimal solutions uses the computation of the EABT problem as a subprocedure. We observe that it needs, at most, double the time it is required for the solution of EABT. This can be again explained by comparing the values of  $\hat{k}$  and  $\hat{M}$ . Since  $\hat{M}$  is, at most, twice the value of  $\hat{k}$ , we expect (from the worst-case bounds of EABT and all Pareto-optima problems) a similar behavior in the actual running times.

## 10. CONCLUSIONS AND FURTHER DISCUSSION

We have discussed time-expanded and time-dependent models for several kinds of single-criteria and bicriteria optimization problems on timetable information systems. In the time-expanded case, extensions that model more realistic requirements (like modeling train transfers) 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 bicriteria optimization problems. In the time-dependent case, the central characteristic of having one node per station had to be violated when more realistic requirements (like the integration of minimum transfer times at stations) were considered and more sophisticated techniques in the bicriteria optimization problems had to be used for their effective solution. Nevertheless, all the problems under consideration could be efficiently modeled in an extension of the time-dependent model introduced here. Moreover, it turned

out that such modeling was more compact than its counterpart in the extended time-expanded model, thus resulting in better performance in practice.

Our experimental study showed that the time-dependent approach is clearly superior with respect to performance when the original version of the models is considered and speedup factors in the range from 12 to 57 were observed. When considering extensions of the models for the solution of realistic versions of optimization problems, the time-dependent approach still performs better, but with a much smaller difference (the speedup is now reduced in the range of 1.5 to 3.3). The time-expanded approach benefits, in this case, from the straightforward modeling that allows more direct extensions and effective solutions.

Finally, we would like to comment on the solution of the latest departure problem. In the time-expanded model, it can be solved in the same way as the lexicographically first Pareto-optimum problem (Section 7.1.1) by minimizing the difference between arrival time and actual departure time as second criterion. In the time-dependent model, this approach does not work, since the use of the EA cost as the first criterion may again result in not nondecreasing functions (as in the case of the lexicographically first (EA,MNT) Pareto-optimum problem; see Section 7.2.1). The latest departure problem can be solved by computing two paths in the EAP setting. In particular, we first compute a shortest (EA) path from the source station  $A$  to the destination station  $B$  (starting not earlier than a given time  $t_A$ ). Let  $t_B$  be the arrival time of this path at  $B$ . We then compute a *maximum cost path* from  $B$  (departing at time  $t_B$ ) to  $A$  in the reverse digraph  $G'$  of the time-dependent (or train-route) digraph  $G$ , where the directions of all edges have been reversed and the edge costs have become non-positive. More precisely, if  $(u, v)$  is an edge of  $G$  with constant cost  $c$ , then the cost of  $(v, u)$  in  $G'$  will be  $-c$ . If  $(u, v)$  is an edge of  $G$  with a timetable (i.e., associated with several elementary connections), then to compute the cost of  $(v, u)$  in  $G'$  we do the following. Let  $t_v$  be the time at which node  $v$  is reached during the computation of a maximum cost path in  $G'$ . To find the time (label) that should be associated with  $u$ , we find the event  $e = (d_e, a_e)$  of  $(u, v)$  that has the latest possible arrival time, subject to the constraint that  $t_v \geq a_e \pmod{1440}$ . The cost of the edge  $(v, u)$  will then be equal to  $-(\text{cycle difference}(a_e, t_v) + \text{length}(e))$ . The edge relaxations are performed in the same way as in Dijkstra's algorithm, taking into consideration that we are now looking for a maximum cost path. It can be easily verified that the computed path in  $G'$  is the earliest-arrival connection to  $B$  with the latest departure from  $A$ .

## REFERENCES

- BRODAL, G. S. AND JACOB, R. 2004. 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, vol. 92. Elsevier.
- DIJKSTRA, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik 1*, 269–271.
- HAFAS. A timetable information system by HaCon Ingenieuresellschaft mbH, Hannover, Germany. <http://www.hacon.de/hafas/>.
- LENGAUER, T. 1990. *Combinatorial Algorithms for Integrated Circuit Layout*. Wiley, New York.

- MÖHRING, R. 1999. *Angewandte Mathematik—insbesondere Informatik*. Vieweg, Wiesbaden, Germany. 192–220.
- MÜLLER-HANNEMANN, M. AND WEIHE, K. 2001. Pareto shortest paths is often feasible in practice. In *Algorithm Engineering—WAE 2001*. Lecture Notes in Computer Science, vol. 2141. Springer, New York. 185–198.
- MÜLLER-HANNEMANN, M., SCHNEE, M., AND WEIHE, K. 2002. Getting train timetables into the main storage. In *Proc. 2nd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2002)*. Electronic Notes in Theoretical Computer Science, vol. 66. Elsevier, New York.
- NACHTIGAL, K. 1995. Time depending shortest-path problems with applications to railway networks. *European Journal of Operations Research* 83, 154–166.
- ORDA, A. AND ROM, R. 1990. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM* 37, 3.
- ORDA, A. AND ROM, R. 1991. Minimum weight paths in time-dependent networks. *Networks* 21.
- PALLOTTINO, S. AND SCUTELLÀ, M. G. 1998. *Equilibrium and Advanced Transportation Modelling*. Kluwer Academic Publ. Boston, MA. Chapter 11.
- PYRGA, E., SCHULZ, F., WAGNER, D., AND ZAROLIAGIS, C. 2004a. Experimental comparison of shortest path approaches for timetable information. In *Algorithm Engineering and Experiments—ALENEX 2004*. SIAM, 88–99.
- PYRGA, E., SCHULZ, F., WAGNER, D., AND ZAROLIAGIS, C. 2004b. Towards realistic modeling of timetable 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, vol. 92. Elsevier, New York. 85–103.
- SCHULZ, F., WAGNER, D., AND WEIHE, K. 2000. Dijkstra’s algorithm on-line: An empirical case study from public railroad transport. *ACM Journal of Experimental Algorithmics* 5, 12.
- SCHULZ, F., WAGNER, D., AND ZAROLIAGIS, C. 2002. Using multi-level graphs for timetable information in railway systems. In *Algorithm Engineering and Experiments—ALENEX 2002*. Lecture Notes in Computer Science, vol. 2409. Springer, New York. 43–59.
- WAGNER, D., WILLHALM, T., AND ZAROLIAGIS, C. 2006. Geometric containers for efficient shortest path computation. *ACM Journal of Experimental Algorithmics* 10, 1.3, 1–30.
- ZIEGELMANN, M. 2001. Constrained shortest paths and related problems. Ph.D. thesis, Naturwissenschaftlich-Technischen Fakultät der Universität des Saarlandes.

Received April 2004; revised June 2006; accepted December 2006