Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

ELSEVIER Electronic Notes in Theoretical Computer Science 92 (2004) 85–103
www.elsevier.com/locate/entcs

Towards Realistic Modeling of Time-Table Information through the Time-Dependent Approach*

Evangelia Pyrga[a,1], Frank Schulz[b,2], Dorothea Wagner[b,3], Christos Zaroliagis[a,4]

[a] *Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece, and Department of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece.*

[b] *Department of Computer Science, University of Karlsruhe, 76128 Karlsruhe, Germany.*

Abstract

We consider optimal itinerary problems in time-table information systems supporting a vast number of on-line queries. We exhibit two important extensions of the time-dependent approach to model realistic versions of the Earliest Arrival and Minimum Number of Transfer problems, as well as of a combination of them, that could not be modeled by the original version of the time-dependent approach. We also provide heuristics that speed up implementations and present preliminary experimental results with real-world data.

Keywords: Time-table information, time-dependent model, shortest path, earliest arrival, minimum number of transfers.

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

[1] pirga@ceid.upatras.gr
[2] fschulz@ira.uka.de
[3] dwagner@ira.uka.de
[4] zaro@ceid.upatras.gr

doi:10.1016/j.entcs.2003.12.024

1 Introduction

Modeling time-table information to efficiently answer queries asking for optimal itineraries is a considerably important problem in public transportation, and in particular in railways. The main target underlying the modeling is to process a vast number of on-line queries as fast as possible. In this paper, we are concerned with a specific, query-intensive scenario arising in public railway transport, where a central server is directly accessible to any customer either through terminals in train stations or through a web interface, and has to answer a potentially infinite number of queries. The main goal in such an application is to reduce the average response time for a query.

Time-table information is usually studied under two approaches: the *time-expanded* [3,7,8,9], and the *time-dependent* approach [1,4,5,6]. The common characteristic of both approaches is that a query is answered by applying some shortest path algorithm to a suitably constructed digraph. In the former case, the constructed digraph represents explicitly every event of the time-table (departure or arrival at a station), and hence results in the generation of a very large graph. In the latter case, the produced digraph turns out to have much smaller size (usually proportional to the number of stations) and hence it is expected to speed up computations of queries. Consequently, in this paper we focus on the time-dependent approach and in particular on the recently proposed model in [1].

The time-dependent approach [1] constructs the time-dependent digraph $G = (V, E)$ 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 (i.e., served by a train that does not stop in-between). 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. In other words, if T is a set denoting time, and (v, w) is an edge of the graph, then its cost is given by $f_{(v,w)}(t) - t$, where t is the departure time at v, $f_{(v,w)} : T \to T$ is a function such that $f_{(v,w)}(t) = t'$, and $t' \geq t$ is the earliest possible arrival time in w. A *timed* (v_1, v_k, t)*-path* in G – corresponding to an itinerary we wish to find – is a sequence $v_1, v_2, \ldots, v_k \in V$ and a sequence $t_1 = t, t_2, \ldots, t_k \in T$ such that $(v_i, v_{i+1}) \in E$ and $t_{i+1} = f_{(v_i,v_{i+1})}(t_i)$, $\forall 1 \leq i < k$.

Two fundamental time-table information problems are the *Earliest Arrival Problem (EAP)* and the *Minimum Number of Transfers Problem (MNTP)*. Given two stations a and b, the EAP is defined to be the problem of finding a timed (a, b, t)-path subject to the additional constraint that the arrival time in b is the earliest possible, while for the MNTP we search for a timed (a, b, t)-path subject to the constraint that we perform as less transfers from one

train to another as possible. Of certain interest is also a combination of the above problems: among all answers to MNTP select the one which attains the earliest arrival time.

The time-dependent model can solve efficiently the earliest arrival problem by reducing it to a shortest path computation on G, under the assumption that a transfer from one train to another can be performed in zero time [1]. However, this is clearly not a realistic assumption. Moreover, this model cannot solve the minimum number of transfers problem, since even if we keep track of when there is a transfer and use for each edge the event that avoids having a transfer instead of the earliest one in time, it will not provide us with the optimal solution. When we have no other option but to make a transfer, there is no way to choose the best event among all possible ones.

In this paper, we present two extensions of the time-dependent approach that can efficiently solve both the realistic version of EAP (i.e., the one with non-zero transfer time per station) and the MNTP, as well as their combination mentioned above. The first extension considers incorporation of platform information and solves the realistic version of EAP, but not the MNTP. The second extension considers incorporation of train route information, solves both the the realistic version of EAP and the MNTP, and constitutes an important alternative in the case where platform information is not provided, or the solution of MNTP is requested. Similar approaches for incorporating platform or train route information were also briefly discussed in [1] for the case of constant time for changing trains, but neither the full details were provided nor the case of non-constant time for train changes was considered. Except for the modeling, we also provide heuristics that speed up implementations and present preliminary experimental results on real-world data.

The rest of the paper is organized as follows. In Section 2, we present the original time-dependent model and discuss the solution of the EAP with zero transfer times. In Section 3, we present our modelings through platform and train route information for the realistic version of EAP with constant or variable non-zero transfer times. The modeling and solution to the MNTP is provided in Section 4, while Section 5 discusses the solution to the combination of EAP and MNTP. Section 6 presents several heuristics that speed up computations of edge costs. In Section 7 we present our preliminary experimental results, and we conclude in Section 8.

2 Time-Dependent Model and Solving EAP with Zero Transfer Time

As mentioned above, the *time-dependent* digraph [1] contains one node per station, and there is an edge e from station A to station B if there is an elementary connection from A to B. The set of elementary connections from A to B is denoted by $\mathcal{C}(e)$. The *departure* and *arrival times* $d(c)$ and $a(c)$ of an elementary connection $c \in \mathcal{C}(e)$ within a day are integers in the interval $[0, 1439]$; the granularity of the timetable is one minute, and time values are minutes after midnight. Given two time values t 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*$(d(c), a(c))$.

Let D denote the departure station and t_0 the earliest departure time. A modification of Dijkstra's algorithm can be used to solve the earliest arrival problem in the time-dependent model [1]. The differences, w.r.t. Dijkstra's algorithm, are: set the distance label $\delta(D)$ of the starting node corresponding to the departure station D to t_0 (and not to 0), and calculate the edge lengths on-the-fly. The edge lengths (and implicitly the time-dependent function f) are calculated as follows. Since Dijkstra's algorithm is a label-setting shortest-path algorithm, whenever an edge $e = (A, B)$ is considered the distance label $\delta(A)$ of node A is optimal. In the time-dependent model, $\delta(A)$ denotes the earliest arrival time at station A. In other words, we indeed know the earliest arrival time at station A whenever the edge $e = (A, B)$ is considered, and therefore we know at that stage of the algorithm which train has to be taken to reach station B via A as early as possible: the first train that departs later than or equal to the earliest arrival time at A[5]. Let $t = \delta(A)$ and let $c^* \in \mathcal{C}(e)$ be the connection minimizing $\{\textit{cycle-difference}(t \bmod 1440, d(c)) \mid c \in \mathcal{C}(e)\}$. 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 length of e, $\ell_e(t)$, is then defined as $\ell_e(t) = \textit{cycle-difference}(t \bmod 1440, d(c^*)) + \textit{length}(c^*)$. Consequently, $f_e(t) = t + \ell_e(t)$.

The correctness of the above algorithm is based on the fact that f is *non-decreasing* ($t \leq t' \Rightarrow f(t) \leq f(t')$) and has *non-negative delay* ($\forall t, f(t) \geq t$). Because of the nature of the investigated application, we can safely assume that all functions defined throughout the paper have non-negative delay.

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

3 The Earliest Arrival Problem with Non-Zero Transfer Time

In this section, we will describe two extensions of the time-dependent model, in order to solve the realistic version of EAP, where transfer time between trains at a station is non-zero. In the following, we shall denote by $\mathcal{S}$ a set of nodes such that every $u \in \mathcal{S}$ corresponds to a station, and by $station(u)$, $u \in \mathcal{S}$, the actual station which u represents.

3.1 Modeling the Change of Trains through Platform Information

In order to wave the zero transfer time assumption in the time-dependent model, we expand the time-dependent graph in such a way that the change of platforms is incorporated in the model. We shall study both the cases of constant and variable transfer time. A similar idea for the former case was briefly mentioned in [1]. We assume that we are given a time-table along with the arrival/departure platform and time for every train that arrives/departs to/from a certain station. We will make use of the following assumption.

Assumption 3.1 *For any stations given, there are no two trains leaving station A (from the same platform) and arriving to station B (at the same platform), such that the train that leaves A second arrives first at B.*

3.1.1 Graph Model

In the following, we will describe the construction of a graph $G = (V, E)$ that models EAP with non-zero transfer times through platform information. Let $\mathcal{S}$ be the set of nodes representing the stations. For each node $u \in \mathcal{S}$ representing a station having $P_u > 0$ platforms, we construct a set of nodes $\mathcal{P}_u = \{p_0^u, p_1^u, ..., p_{P_u-1}^u\}$ where $p_i^u, 0 \leq i < P_u$, is the node representing the i-th platform of u. Then, the node set of G is defined as $V = \mathcal{S} \bigcup \mathcal{P}$, where $\mathcal{P} = \bigcup_{u \in \mathcal{S}} \mathcal{P}_u$.

The edge set $E = A \bigcup D \bigcup \overline{D} \bigcup R$ of G consists of four types of edges which are defined as follows.

- $A = \bigcup_{u \in \mathcal{S}} A_u$, where $A_u = \bigcup_{0 \leq i < P_u} \{(p_i^u, u)\}$.
- $D = \bigcup_{u \in \mathcal{S}} D_u$, where $D_u = \bigcup_{0 \leq i < P_u} \{(u, p_i^u)\}$.
- $\overline{D} = \bigcup_{u \in \mathcal{S}} \overline{D}_u$, where $\overline{D}_u = \emptyset$, if at each station the time for changing platforms is constant; otherwise, $\overline{D}_u$ is the set of edges between the nodes of $station(u)$ that represent the platforms in such way as to model how the platforms of $station(u)$ are connected.
- $R = \bigcup_{u,v \in \mathcal{S}, 0 \leq i < P_u, 0 \leq j < P_v} \{(p_i^u, p_j^v) : \exists$ at least one event departing from p_i^u

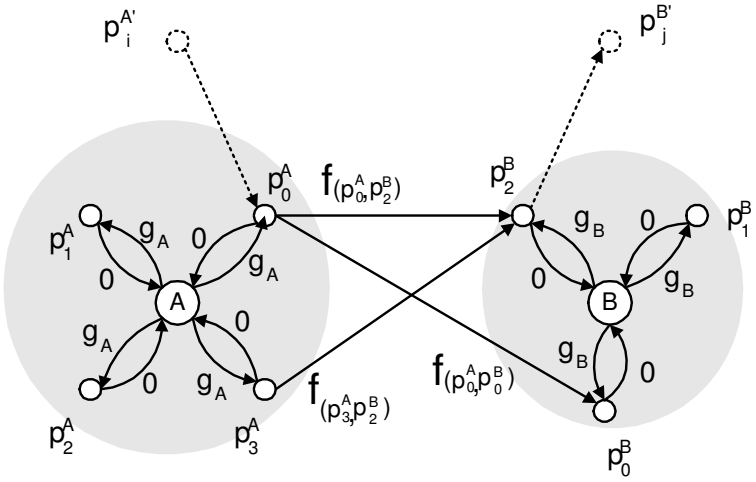

Fig. 1. An example of platform modeling with constant platform change per station.

and arriving at $p_j^v\}$.

If $u, v \in \mathcal{S}$, $0 \leq i, i' < P_u$, $0 \leq j < P_v$, then the edge costs are defined as follows (see Figures 1 and 2).

- An edge $(p_i^u, u) \in A$ is defined to have zero cost.
- When the time for moving from one platform to any other of the same station is constant for each station, then an edge $(u, p_i^u) \in D_u$ has cost determined by a function $g_u : T \to T$ such that $g_u(t) = t + \tau^u_{change}$ where τ^u_{change} is the time needed for transferring from one platform to another at station $station(u)$. If the cost for changing varies among the platforms (see Figure 2), then an edge $(u, p_i^u) \in D_u$ has zero cost, while an edge $(p_i^u, p_j^u) \in \overline{D}_u$ has cost determined by a function $g_{u_{i,j}} : T \to T$ such that $g_{u_{i,j}}(t) = t + \tau^u_{change(i,j)}$, where $\tau^u_{change(i,j)}$ is the time needed for a passenger that was at the i-th platform of u to change to platform j.
- An edge $(p_i^u, p_j^v) \in R$ has cost determined by a function $f_{(p_i^u, p_j^v)} : T \to T$ such that $f_{(p_i^u, p_j^v)}(t)$ models the time that p_j^v will be reached, given that p_i^u was reached at time t.

The correctness of the above modeling can be seen as follows. No two trains can be at the same platform at the same time. Furthermore, the time interval, from the moment that a train arrives at a platform until the moment that the immediately next train that stopped at the same platform departs, is sufficient for a passenger that arrived with the first train to step down and get on the second one. This means that if the information concerning the platforms is available, then the above model can solve the EAP with respect

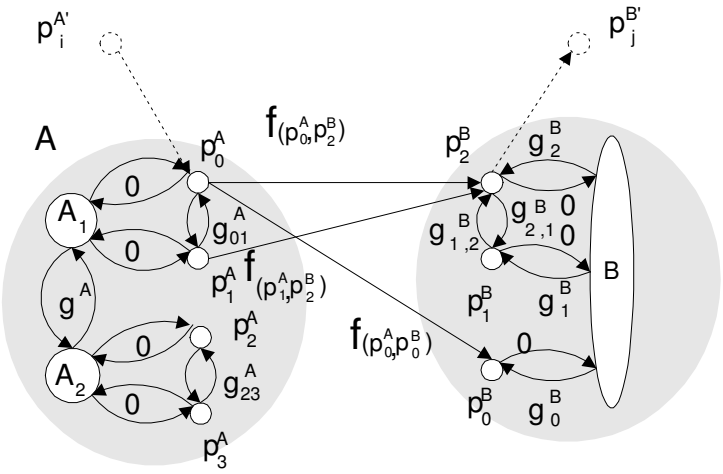

Fig. 2. An example of platform modeling with variable time for platform change per station. A_1, and A_2 denote virtual platforms introduced only to represent the way the platforms of station A are connected. For simplicity, the node of $\mathcal{S}$ that represents A and the edges that connect it with the other nodes of A are not shown.

to train changes.

Nevertheless, this model cannot solve the MNTP for about the same reasons as the original time-dependent model.

3.1.2 Algorithm

The algorithm used to find the shortest path from station α to β is the modified Dijkstra used in [1] (cf. Section 2), where for each edge we use the function associated with it (cf. Section 3.1.1). Given a query (α, β, t_0), then we need to find the shortest path in the graph $G = (V, E_{\alpha,\beta})$ from s_α to s_β starting at time t_0, where $\alpha = station(s_\alpha)$, $\beta = station(s_\beta)$, and $E_{\alpha,\beta}$ is defined as follows. When the cost for changing platforms is the same within the same station, then $E_{\alpha,\beta} \equiv A \bigcup D \bigcup R$ (see Figure 1); otherwise, $E_{\alpha,\beta} \equiv D_{s_\alpha} \bigcup A_{s_\beta} \bigcup \overline{D} \bigcup R$ (see Figure 2). Note that even when the time needed for a platform change is constant at every station, we use zero cost for all edges $e \in D_{s_\alpha}$.

The correctness of the algorithm is established by the fact that functions f and g are assumed to have non-negative delay and that they are non-decreasing (f by Assumption 3.1, and g by construction).

3.2 Modeling the Change of Trains through Train Routes

In this section we will use a different graph in order to model the train changes using the routes that trains may follow. We will examine both the cases of constant and variable transfer times. A similar idea for the former case was

briefly mentioned in [1]. Again, let $G = (V, E)$ be the new graph model whose node and edge sets will be defined in the following. We assume that we are given a set of train routes and their respective time schedules.

Let $\mathcal{S}$ be the set of nodes representing the stations. We say that nodes $s_0, s_1, ..., s_{k-1}$, $k > 0$, form a *train route* if there is some train starting its journey from $station(s_0)$ and visiting consecutively $station(s_1), ..., station(s_{k-1})$ in turn. If there are more than one trains following the same schedule (with respect to the order in which they visit the above nodes), then we say that they all belong to the same train route P. Note that it can be $station(s_i) \equiv station(s_j)$, $i \neq j$, $s_i \neq s_j$, $0 \leq i, j \leq k-1$, for example when the train performs a loop.

For $u \in \mathcal{S}$, let Σ_u be the set of different train routes that stop at $station(u)$, and let $\mathcal{P}_u$ be the set that contains exactly one node for each $P \in \Sigma_u$ that passes through $station(u)$ (note that if P performs a loop, then it contributes more than one nodes to $\mathcal{P}_u$). Also, let $P_u = |\mathcal{P}_u|$, and $\mathcal{P} = \bigcup_{u \in \mathcal{S}} \mathcal{P}_u$. Then, the node set V of G is defined as $V = \mathcal{S} \bigcup \mathcal{P}$. For $u \in \mathcal{S}$, we denote by $p_i^u, 0 \leq i < P_u$, the node representing the i-th train route that stops at u.

The edge set $E = A \bigcup D \bigcup \overline{D} \bigcup R$ of G consists of four types of edges which are defined as follows.

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

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

Assumption 3.2 *Let u, v be any two nodes in $\mathcal{S}$ 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 the trains that belong to the same train route into two (or more) different

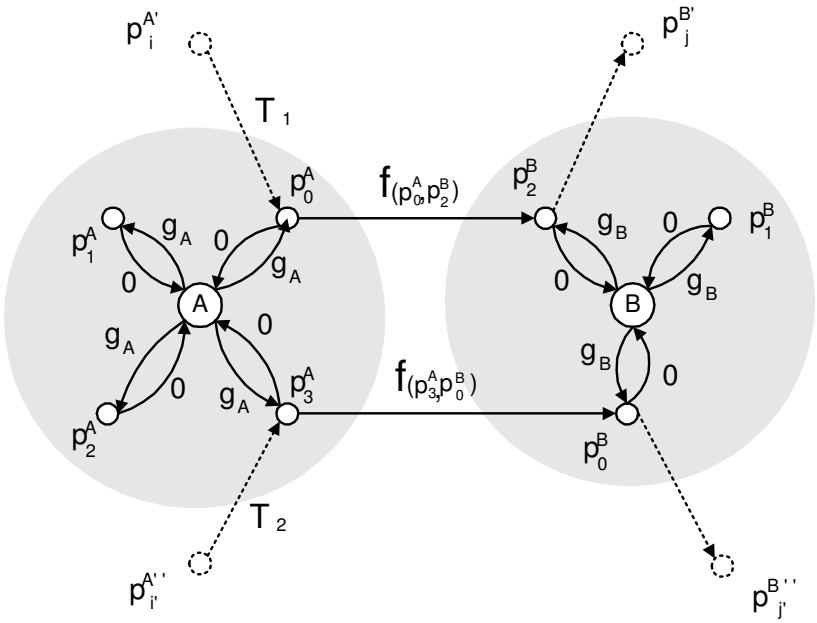

Fig. 3. An example of modeling through train routes with constant transfer time per station.

train routes that follow the same schedule as before. This separation can be done by separating the trains into different speed classes.

Assumption 3.3 *For any station node $u \in \mathcal{S}$ 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 τ_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 3.3 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 3.2 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 3.3 will only be needed in the case where we consider variable transfer times within the same station.

In the following, we shall assume that $u, v \in \mathcal{S}$, $0 \leq i, i' < P_u$, $0 \leq j < P_v$.

3.2.1 Constant Transfer Time

In this case, the edge costs are defined as follows (see Figure 3).

- An edge $(p_i^u, u) \in A$ is defined to have zero cost.

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

The correctness of the above model is similar to that of its corresponding model based on platform information (cf. Section 3.1.1). Moreover, the algorithm given in Section 3.1.2 can also be used to solve the EAP under the new modeling. Its correctness follows again by the non-negative delay assumptions of f and g, and by the fact that both functions are non-decreasing (f by Assumption 3.2, and g by construction).

3.2.2 Variable Transfer Time

The edge costs in this case are defined as follows (see also Figure 4).

- An edge $(p_i^u, u) \in A$ is defined to have 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 $f_{(p_i^u, p_{i'}^u)} : T \to T$ 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 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 $f_{(p_i^u, p_j^v)} : T \to T$ 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.

The correctness of the above model follows similarly to that of its corresponding model based on platform information (cf. Section 3.1.1).

Let nodes $p_i^u, p_j^u \in \mathcal{P}_u$, $0 \le i, j < P_u$, $i \ne j$, such that $(p_i^u, p_j^u) \in \overline{D}$. In addition let node $p_{i'}^v \in \mathcal{P}_v$, $0 \le i' < P_v$ such that $(p_{i'}^v, p_i^u) \in R$. In order to be able to apply the algorithm of Section 3.1.2 and solve the EAP in this case, we have to ensure that the functions are non-decreasing. Assumption 3.2 ensures that $f_{(p_{i'}^v, p_i^u)}(t)$ is non-decreasing. What we need to prove is that $f_{(p_i^u, p_j^u)}(t)$ is also non-decreasing when $t \in T_{u_i}$, where $T_{u_i} = \{t | t$ is the arrival time to p_i^u from the edge $(p_{i'}^v, p_i^u) \in R\}$. (Note that $p_{i'}^v$ is the only neighboring node of p_i^u that is not in $\mathcal{P}_u$.)

Lemma 3.4 *The function $f_{(p_i^u, p_j^u)}(t)$ is non-decreasing when $t \in T_{u_i}$, where $T_{u_i} = \{t | t$ is the arrival time to p_i^u from the edge $(p_{i'}^v, p_i^u) \in R\}$.*

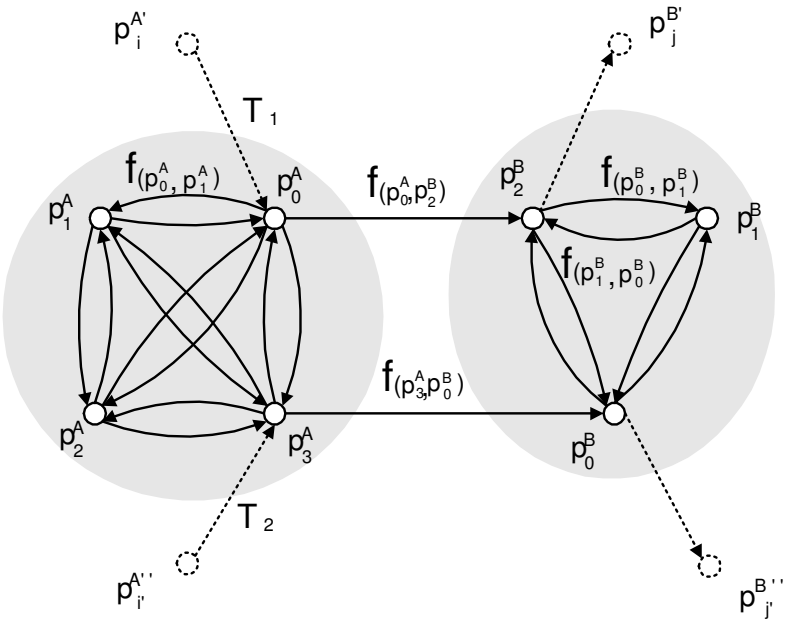

Fig. 4. An example of modeling through train routes with variable transfer times at the stations.

Proof. Let τ_u be the maximum time needed for a transfer at $station(u)$ and $\delta^{u_i}_{v_{i'}}$ be the minimum interval between two successive arrivals to p^u_i from $(p^v_{i'}, p^u_i)$ and $t_1, t_2 \in T_{u_i}$ where $t_1 < t_2$. Since t_1, t_2 are two distinct arrival times at p^u_i from $(p^v_{i'}, p^u_i)$, then $t_2 - t_1 \geq \delta^{u_i}_{v_{i'}}$. By Assumption 3.3, we have that $\delta^{u_i}_{v_{i'}} \geq \tau_u$. Also, $f_{(p^u_i,p^u_j)}(t) - t \leq \tau_u$, $t \in T$, since $f_{(p^u_i,p^u_j)}(t) - t$ is the time needed for a transfer from p^u_i to p^u_j on time t. Consequently,

$$t_2 - t_1 \geq \delta^{u_i}_{v_{i'}} \geq \tau_u \geq f_{(p^u_i,p^u_j)}(t_1) - t_1$$

$$\Rightarrow t_2 - t_1 \geq f_{(p^u_i,p^u_j)}(t_1) - t_1$$
$$\Rightarrow f_{(p^u_i,p^u_j)}(t_1) \leq t_2 \leq f_{(p^u_i,p^u_j)}(t_2)$$
$$\Rightarrow f_{(p^u_i,p^u_j)}(t_1) \leq f_{(p^u_i,p^u_j)}(t_2)$$

which completes the proof of the lemma. □

4 The Minimum Number of Transfers Problem

The models for EAP based on train routes and described in Section 3.2 can be used to solve the Minimum Number of Transfers problem. In this case we have a different set of edge costs defined as follows.

- An edge $(p^u_i, u) \in A$ is defined to have zero cost.

- The cost of every edge in D and $\overline{D}$ is defined to be 1, modeling the transfer between two trains.
- An edge $(p_i^u, p_j^v) \in R$ has zero cost.

Regarding the edges between stations, the algorithm considers either the connection that belongs to the same train, when no transfer took place, or the first possible event, when there was a transfer.

The algorithm given in Section 3.1.2 can be used in this case too, since we are again dealing with non-decreasing functions. Note that when applying the algorithm all edges in D_{s_α} must have cost zero.

We now turn to the correctness of the model. We give a proof for the case where we use the modeling of EAP based on train routes with variable transfer costs (cf. Section 3.2.2). The other case (constant transfer costs; cf. Section 3.2.1) can be proved similarly.

As we have already mentioned, the algorithm provides us with a shortest path in the graph we have constructed. The question now is whether the computed shortest path agrees with the real shortest routing possessing the minimum number of transfers that a passenger wishes to follow.

Assume that we are given an (s, t, τ) query, $s, t \in \mathcal{S}$, $s \neq t$, and suppose that we do not stop the algorithm when t has been settled, but we continue to compute a shortest path tree Δ. Let $\Pi = (s \equiv u_0, u_1, ..., u_k \equiv t)$, $k > 0$, be the nodes of G that the projection of the real shortest path visits in turn. Note that there is a unique edge between u_i and u_{i+1}, $0 \leq i < k$. Let $\delta_\Pi(u)$ and $\delta_\Delta(u)$, $\forall u \in \Pi$, be the cost of u in the real shortest path Π and the computed shortest path tree Δ, respectively.

Lemma 4.1 *For every $u \in \Pi$, $\delta_\Pi(u) = \delta_\Delta(u)$.*

Proof. We shall prove the lemma by first proving that $\delta_\Pi(u) \leq \delta_\Delta(u)$, and then $\delta_\Pi(u) \geq \delta_\Delta(u)$, $\forall u \in \Pi$.

Consider some route that a passenger may follow. If we break this route into the different train routes that the passenger has been on, one by one, we can see that these form a path in G. (In order to change from one train route to another the passenger has to get off to some station and make the transfer.) Note that G captures all changes between two different train routes as having one edge $e \in D \bigcup \overline{D}$ included in the path. The only transfers that are not included in this way are the ones that occur from one train that belongs to some train route P to another train that also belongs to P. No shortest path routing, however, will include such transfers, since continuing with the same train always provides us with a smaller number of transfers. (Recall that since both trains belong to P, they will both visit exactly the same nodes/stations in exactly the same order.) This means that all transfers that a real shortest

path would make are modeled by the use of edges belonging to $D \bigcup \overline{D}$, and consequently $\delta_\Pi(u) \leq \delta_\Delta(u)$, $\forall u \in \Pi$.

We now turn to the proof of $\delta_\Pi(u) \geq \delta_\Delta(u)$. We proceed by induction on the length k of Π. The basis, $k = 0$, holds trivially, since $\delta_\Pi(s) = \delta_\Delta(s) = 0$. Assume that it holds for all values smaller than k.

Let $u_k \equiv p_j^v$, for some $v \in \mathcal{S}$, $j < P_v$. There are two cases regarding u_{k-1}: either $u_{k-1} \equiv p_{j'}^v$, $j' < P_v$, $j \neq j'$; or $u_{k-1} \equiv p_{j'}^w$, $w \in \mathcal{S}$, $j' < P_w$, $w \neq v$ and $(p_{j'}^w, p_j^v) \in R$. By the induction hypothesis, we have $\delta_\Pi(u_{k-1}) = \delta_\Delta(u_{k-1})$. The first case means that there was a transfer in Π at $station(v)$, hence:

$$\delta_\Pi(u_k) = \delta_\Pi(p_j^v) = \delta_\Pi(p_{j'}^v) + 1 = \delta_\Pi(u_{k-1}) + 1$$

But when the algorithm permanently labeled $p_{j'}^v$, it must have updated all of its outgoing edges. Consequently,

$$\delta_\Delta(u_k) = \delta_\Delta(p_j^v) \leq \delta_\Delta(p_{j'}^v) + 1 = \delta_\Delta(u_{k-1}) + 1 = \delta_\Pi(u_{k-1}) + 1 = \delta_\Pi(u_k)$$

For the second case, $u_{k-1} \equiv p_{j'}^w$, we have no transfer from u_{k-1} to u_k, since they both belong to the same train route. Hence, $\delta_\Pi(u_k) = \delta_\Pi(u_{k-1})$ and

$$\delta_\Delta(u_k) = \delta_\Delta(p_j^v) \leq \delta_\Delta(p_{j'}^w) + 0 = \delta_\Delta(u_{k-1}) = \delta_\Pi(u_{k-1}) = \delta_\Pi(u_k)$$

which completes the proof of the lemma. □

5 Combination of MNTP and EAP

The train-route modeling can also be used to solve the combined problem of finding among all the paths with minimum number of transfers, the one that has the earliest arrival time. We shall refer to this problem as (MNTP, EAP). A solution to this 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 lexicographical comparison to these pairs: $(a_1, b_1) + (a_2, b_2) = (a_1 + b_1, a_2 + b_2)$; $(a_1, b_1) < (a_2, b_2)$ iff $a_1 < a_2$, or $a_1 = a_2$ and $b_1 < b_2$. The attribute values a, b of the pairs are updated separately. For (MNTP, EAP), a is the MNT cost and b is the EA cost. An edge $e \in E$ has cost determined by a function $h_e : (\mathbb{N}, T) \to (\mathbb{N}, T)$, such that each attribute value is determined by the corresponding edge function. The reason why this is correct when modeling with train routes is that the new edge cost function h_e remains non-decreasing and with non-negative delay. For example, for the constant transfer cost modeling, if $\tau, \tau_1, \tau_2 \in \mathbb{N}$ and $t, t_1, t_2 \in T$:

- If $e \in A$, 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 non-negative 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 proved similarly.

6 Implementation Issues when Modeling with Train Routes

In order to find quickly 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 $f'_{(x,y)} : T \to T$ 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 $(p_i^u, p_{i'}^u) \in \overline{D}_u, u \in \mathcal{S}$, 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 $(p_j^v, p_i^u), (p_i^u, p_l^w) \in R$ be two edges which belong to the same route and let $\mathcal{T}_{(p_j^v, p_i^u)}$, $\mathcal{T}_{(p_i^u, p_l^w)}$ 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 $same\_train_e$ to the event e' of $\mathcal{T}_{(p_i^u, p_l^w)}$ 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 $same\_train_e$ 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 $(p_{i'}^u, p_{l'}^{w'})$, for $p_{l'}^{w'} \in V$, be the outgoing edge of $p_{i'}^u$ that belongs to R, and $\mathcal{T}_{(p_{i'}^u, p_{l'}^{w'})}$ its respective timetable. Instead of storing a timetable for $(p_i^u, p_{i'}^u)$, we store for each $e \in \mathcal{T}_{(p_j^v, p_i^u)}$ a pointer $very\_next_{e,i'}$, to the event e' of $\mathcal{T}_{(p_{i'}^u, p_{l'}^{w'})}$ such that e' is the first event of $(p_{i'}^u, p_{l'}^{w'})$ that can be followed given that p_i^u was reached by use of e. Having one such pointer $very\_next_{e,i'}$ 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 $same\_train_e$ pointer is $very\_next_{e,i}$ 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 the graph a little more, if necessary. 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 $next\_departure_u$ is maintained. If during the execution of the algorithm u is reached by its incoming (if any) route edge, then $next\_departure_u$ 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 $next\_departure_u$ is set to -1. In this way, when u is extracted from the priority queue, if $next\_departure_u$ is non-negative, 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 $next\_departure_u$ 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.

7 Experiments

In this section, we present preliminary experimental results with the model based on train routes, since we wanted to experiment with all problems (EAP,

MNTP, and their combination). We consider the case of constant time for transfers at each station.

7.1 Experimental Set-up and Data

We have implemented the algorithm for the original time-dependent model [1], which can efficiently solve the EAP with Zero Transfer time (EAP-ZT), as well as the modified versions of it described in this paper regarding the extension of the model based on train routes with constant transfer time per station. The latter algorithms have been used to solve the EAP with Non-Zero Transfer Time (EAP-NZT), the MNTP, as well as the combination of MNTP and EAP-NZT as described in Section 5.

The implementation for the original time-dependent model uses binary search and it was written using the LEDA parameterized graph type. The implementation for the extension with train routes contains the index values at the route nodes, as described in Section 6. We have also used a heap-based priority queue that bears a great resemblance to the lazy variant of pairing heaps (see [2]). Special care has also been taken to avoid the initialization steps to both models at the beginning of each query.

The code is written in C++ and compiled with the GNU C++ compiler version 2.95.3; the experiments were run on a SunFire 280R with an UltraSPARC-III processor at 750 MHz and 512 MB of memory running Solaris 8.

The same input has been used to both models. The input timetable resembles the long-distance traffic in Germany for a given day, assuming that the same timetable is valid every day. Table 1 shows the characteristics of the graphs constructed by the two different models for the above input.

Model	#Station Nodes	#Route Nodes	#Timetable Edges	#Transfer Edges	Avg.Elem.Conn. per timetable edge
Zero Transfer Time	6685	–	17577	–	18.77
Non-Zero Transfer Time	6685	79784	72779	159568	4.56

Table 1
Key parameters of the graphs for the different models, given the same input.The Zero Transfer Time Model refers to the original time-dependent model, while the Non-Zero Transfer Time Model refers to the Train-Route modeling with constant transfer time.

Two different sets of 50,000 queries were used. One set consists of real-world queries, while the other consists of randomly generated queries. Each

query consists of a departure station, a destination station and an earliest departure time.

For each combination of problem and query set, we have measured the following performance parameters as mean values over the set of performed queries: the number of nodes touched by the algorithm, the number of timetable edges and the number of transfer edges touched by the algorithm, the number of binary steps per timetable edge, and the CPU-time in milliseconds. The results are reported in Table 2.

Problem	Queries	Average Nodes Touched	Average Timetable Edges touched	Average Transfer Edges touched	Avg.Binary steps per timetable edge	Average Time [msec]
EAP-ZT	real	2967	4365	–	3.125	9.7
EAP-NZT	real	44710	38146	45467	0.320	62.8
MNTP	real	26812	21677	61574	0	28.8
MNTP, EAP-NZT	real	28390	23008	60459	0.180	80.7
EAP-ZT	rand	3315	4811	–	3.006	10.5
EAP-NZT	rand	48127	40943	48859	0.315	68.0
MNTP	rand	33629	27398	69238	0	34.4
MNTP, EAP-NZT	rand	35406	28915	68944	0.176	101.3

Table 2
Results of the experiments run using the time-dependent model and the train-route model with constant transfer cost at each station.

7.2 Results and Discussion

From Table 1 we can see that for the same input, the graph based on train route modeling (referred to as train-route graph) has approximately 13 times more nodes and edges than the graph of the original time-dependent model. In Table 2 though, it is shown that both for the cases of real and random queries the time for solving EAP-NZT is only by a factor of 6.5 slower than the time of EAP-ZT. This is due to the fact that the structure of the two graphs is quite different. While the time-dependent graph has only timetable edges, almost 70% of the train-route graph edges have constant cost, while the timetables of the other edges have much less events than their corresponding edges in the time-dependent graph.

The MNTP is solved faster than the EAP-NZT, since no binary search is needed (due to the fact that we do not need to compute the time that each node is reached). The combined problem (MNTP, EAP-NZT) is slower than both MNTP and EAP-NZT, even though its performance parameters are similar to the ones of MNTP. This is due to the fact that in this problem more operations are needed in order to maintain the priority heap. The (MNTP, EAP-NZT) performs roughly half of the binary steps performed by EAP-NZT. 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 will be those with only one transfer, and so on, thus avoiding to perform a transfer as long as this is possible. Now, the implementation of the model 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-NZT, since in order to compare two node labels there will exist quite often the need to compare the second cost parameter (time), as the first one (number of transfers) is the same for much more nodes.

8 Conclusions and Future Work

We have presented two extensions of the time-dependent approach that model realistic versions of time-table information problems, namely the earliest arrival with non-zero transfer times at stations, the minimum number of transfers, as well as a combination of them. We have provided heuristics that improved actual performance for solving these problems and presented preliminary experiments for the case of constant transfer time.

We plan to consider modeling of other realistic time-table information problems as well as of further combinations of basic problems (e.g., Pareto optimal problems), and to complement the presented experimental study with further experiments regarding variable transfer time per station and more input data.

References

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

[2] Michael Fredman, Robert Sedgewick, Daniel Sleator and Robert Tarjan. The Pairing Heap: A New Form of Self-Adjusting Heap. *Algorithmica*, volume 1, pages 111-129, 1986.

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

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

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

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

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

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

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