Edges as Nodes - a New Approach to Timetable Information

Olaf Beyersdorff and Yevgen Nebesov

Institut für Theoretische Informatik, Leibniz-Universität Hannover, Germany
beyersdorff@thi.uni-hannover.de, yevgen.nebesov@stud.uni-hannover.de

Abstract. In this paper we suggest a new approach to timetable information by introducing the "edge-converted graph" of a timetable. Using this model we present simple algorithms that solve the earliest arrival problem (EAP) and the minimum number of transfers problem (MNTP). For constant-degree graphs this yields linear-time algorithms for EAP and MNTP which improves upon the known DIJKSTRA-based approaches. We also test the performance of our algorithms against the classical algorithms for EAP and MNTP in the time-expanded model.

Key words: timetable infomation, earliest arrival problem, minimum number of transfers problem, time-expanded model

1 Introduction

Algorithms for timetable information play an important role in public transportation systems and related applications [8]. A number of important algorithmic problems connecting to timetable information is studied in the literature. One of the most basic of these is the earliest arrival problem (EAP) asking for a route between two stations s and t that assures the earliest possible arrival at t and obeys the specified departure time at s.

While the systems used in practice typically employ heuristics to solve these problems (cf. [8]), there is also a number of exact methods. The two most common approaches are the *time-expanded* and the *time-dependent approach* which transform the initial network into a weighted digraph such that classical algorithms for path search such as DIJKSTRA become applicable [1, 9, 11, 12].

In this paper we propose a novel approach to timetable information which we call the *edge-converted approach*. Similarly as in the time-expanded and time-dependent model, we also convert the initial network into a digraph, but such that elementary connections are represented as nodes. Thus, in some sense, the role of edges and nodes is switched in our model. Based on this model we present two algorithms that solve the earliest arrival problem as well as the minimum number of transfers problem (MNTP). Both algorithms are conceptually simple as they are variants of depth-first and breadth-first search, respectively. Moreover, these algorithms are very efficient—they only use linear time in the size of their input, i.e., in terms of the size of the edge-converted network.

To compare the performance of these algorithms to the DIJKSTRA-based approaches in the time-expanded model [12], we need to compare the sizes of the time-expanded and edge-converted graphs. It turns out, that our model has the advantage to introduce less nodes but uses far more edges (up to $O(n^3)$ in the

J. Clausen, G. Di Stefano (Eds): ATMOS 2009
9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2009/2147

general case). However, we argue that for practical networks only a linear number of edges is needed which leads to linear-time algorithms for EAP and MNTP. In particular, for the class of constant-degree graphs our approach yields linear-time algorithms for EAP and MNTP where the running time is measured in the size of the initial network. This improves upon the known DIJKSTRA-based solutions which consume $O(n \log n)$ running time. We also implemented our algorithms and performed an experimental study which confirms our theoretical results.

This paper is organized as follows. In Sect. 2 we review basic definitions from timetable information including the definition of EAP and MNTP. Section 3 discusses the two main approaches towards these problems. In Sect. 4 we introduce our new model and compare it to the time-expanded approach. The following Sect. 5 contains our algorithmic solutions for EAP and MNTP which are then tested experimentally in Sect. 6. Finally, Sect. 7 concludes with a discussion of our results and directions for future research.

2 Itinerary Problems

A *timetable* is a network composed of nodes (station, bus stops, etc.) and some elementary connections between them. Each elementary connection is a train (or bus, etc.) which starts and arrives at certain nodes and has a certain departure and arrival time. So it can be interpreted as a 4-tuple $e = (s, t, d, a)$, where s and t are nodes, d is the departure time at s and a is the arrival time at t. We will also call s and t the *source node* and the *target node* of e, respectively. A *transfer* between two connections $e_1 = (s_1, t_1, d_1, a_1)$ and $e_2 = (s_2, t_2, d_2, a_2)$ is possible if $t_1 = s_2$ and $a_1 \leq d_2$. A *route* or an itinerary between two nodes s and t is a sequence of elementary connections $(e_1, \ldots, e_n)$, where s is the source node of e_1, t is the target node of e_n, and a transfer between each e_i and e_{i+1} is possible.

The time values are elements of a totally ordered set T with a defined addition operation. As a rule, T consists of integer numbers between 0 and 1439 and represents the time in minutes after midnight. The time may denote one or several successive days which can be integrated into one model by counting the time modulo 1440 and keeping track of the days [5]. In this paper, however, only one day is used as a time horizon.

A number of important problems on timetable information is described in [2, 4, 6, 8, 10, 11]. The *earliest arrival problem* (EAP) is the most basic and fundamental of them. Instances of EAP are 3-tuples (s, t, d), where s is a source node, t is a target node, and d is the earliest departure time at s. The task consists in finding a route from s to t which departs from s not earlier than the given earliest departure time and minimizes the difference between the arrival time at t and the earliest given departure time. EAP has a realistic and a simplified version. The realistic version considers the minimum transfer time at a station. The transfer time in the simplified version is assumed to be 0. In this paper we will only consider the simplified version of EAP.

Another problem in timetable information is the *minimum number of transfers problem* (MNTP). In this case, a query consists of a departure station s

and an arrival station t only. The task is to find an itinerary that minimizes the number of train transfers.

3 Related Work

The existing algorithms for path searches on static networks are not suitable for timetables, since the edges are available only temporarily within a given time window. The most common approaches for solving EAP and MNTP are based on time-expanded [11, 12] and time-dependent [1, 9] models. The definition and detailed analysis of both algorithms are described in [10]. The idea of time-expanded and time-dependent models is to transform or to extend the initial graph in such a way that the known algorithms for static graphs may be applied. Pyrga, Schulz, Wagner, and Zaroliagis [11] showed in an experimental comparison of the time-expanded and time-depended models that the time-dependent approach can be faster than the time-expanded up to factor 40. However, it is not considerably faster in the case of realistic models and has some drawbacks touching the extensions towards realistic models [10], for instance when modelling minimum transfer times at stations. Therefore, only the time-expanded model applied to EAP and MNTP will be considered and then compared to our approach. A comparison with the time-dependent approach is planned to be done in future research.

3.1 The Time-Expanded Model

The time-expanded model is based on the following transformation. Each elementary connection $e = (s, t, d, a)$ induces a copy of the source node s tagged with the departure time stamp d and a copy of the target node t tagged respectively with the arrival time stamp a. Thus, the initial connections become the connections between a pair of copies according to their time stamps.

Next, for each station s of the initial network all its copies will be captured and ordered ascending their time stamps. Let $v_1, \ldots, v_k$ be the copies of s in that order. Then, there is a set of stay-edges (v_i, v_{i+1}), $i = 1, \ldots, k-1$, connecting the two subsequent copies within a station and representing waiting time at that station between two time events. Thus, given a graph (S, E), where S is a set of stations and E is a set of edges or elementary connections, the time-expanded model will include as many as $2|E| - |S|$ stay-edges.

The example in Fig. 1 illustrates the transformation of an initial network to the time-expanded model. The timetable consists of five stations and seven elementary connections between these stations. The time stamps at the edges represent the departure and arrival time of the given connection.

Observation 1 *The route between two nodes consists not only of the elementary connections, but also of some stay-edge connections. It can also happen that there are many stay-edges belonging to only one station. For example, the dashed line in Fig. 1 shows, that the route between stations 1 and 5 includes two stay-edges at station 3. Hence the number of the edges on a route depends on the number of the transfers and on the number of initial events as a whole.*

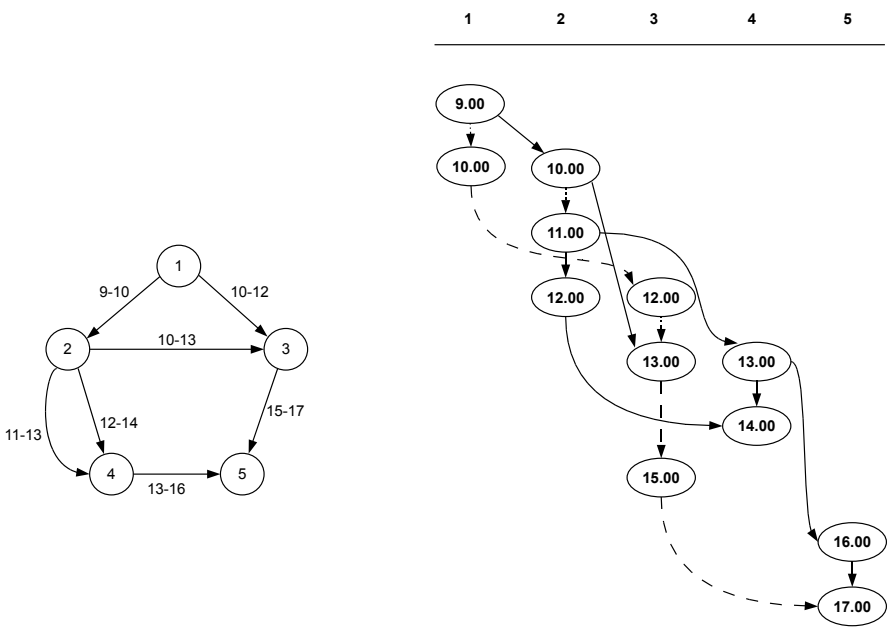

Fig. 1. An initial network and the transformed time-expanded network

The number of nodes in the time-expanded model is equal to the double number of elementary connections of the initial graph, since each connection produces a copy of its source and its target nodes. The number of edges in the time-expanded model includes the elementary connections and the stay-edge connections (cf. Table 1).

Table 1. The size of the time-expanded graph

	Initial graph (S, E)	Time-expanded graph
Number of nodes	$\|S\|$	$2\|E\|$
Number of edges	$\|E\|$	$\leq 3\|E\| - \|S\| \leq 3\|E\|$

3.2 EAP with the Time-Expanded Model

The original approach for solving the shortest-path problem is the Dijkstra algorithm [3]. Every edge in the time-expanded model has departure and arrival time stamps. The time difference between these time stamps can be attached as the weight to the given edge. Starting at the first copy of the source node, but, not earlier than allowed by the earliest departure time, we find a shortest path by reaching any copy of the target node [11]. Given a network $G = (S, E)$, the complexity of the Dijkstra algorithm is $O(|E| + |S| \log |S|)$. According to Table 1, for the timetable with $|S|$ stations and $|E|$ elementary connections, the run-time of the Dijkstra algorithm applied on the time-expanded model is equal to $c(3|E| - |S| + 2|E| \log 2|E|)$, where c is a constant stemming from the Dijkstra algorithm.

3.3 MNTP with the Time-Expanded Model

The DIJKSTRA algorithm can be also used for solving MNTP with the time-expanded model. The edges between copies of different stations are assigned a weight of 1, and stay-edges are assigned a weight of 0. Starting at the first possible copy of a source station, the shortest path to a copy of a target station yields a solution of MNTP. The complexity of MNTP with the time-expanded model coincides with the complexity of EAP, since it uses the same algorithm.

We remark that the above described applications of the time-expanded model refer to the earliest ideas of the time-expanded approach. Recently, many speed-up techniques for EAP and MNTP have been developed. The extensions and improvements of the time-expanded approach and shortest-path algorithms are described in [2, 5, 7, 11, 12]. In this paper our approach for solving EAP and MNTP is only compared to the original formulations of the time-expanded model and the shortest-path algorithms. The comparison to the newest improvements of the time-expanded and time-dependent model should be made in future research.

4 Our Approach: The Edge-Converted Model

In this section we will describe a new model for timetable information. Similar to the time-expanded approach, we use a transformation of the initial network to obtain a static structure supporting well known algorithms, such as DIJKSTRA or breadth-first search. The core idea of our approach is to convert the initial elementary connections to nodes. Therefore we call it edge-converted approach. The whole transformation routine is listed below:

Step 1. At first we take all the stations of the initial network as new nodes. We call these nodes *type A nodes*.

Step 2. Then for every elementary connection $e = (s, t, d, a)$, a new node that gets all four parameters of the edge e will be created. We call these nodes *type B nodes* (see Fig. 2).

Step 3. Now we connect type A nodes to type B nodes according to the next two rules.

a) There is an outgoing edge from a type A node u to a type B node $v = (s, t, a, d)$ if $u = s$.
b) There is an outgoing edge from a type B node $v = (s, t, a, d)$ to a type A node u if $t = u$ (see Fig. 2).

Step 4. Next, we add several edges connecting type B nodes with each other. There are four conditions for the existence of an edge between two type B nodes $u = (s_u, t_u, d_u, a_u)$ and $v = (s_v, t_v, d_v, a_v)$:

a) $t_u = s_v$
b) $a_u \leq d_v$
c) For all type B nodes $w = (s_w, t_w, d_w, a_w)$, if $s_w = s_u$, $t_w = t_u$, and $a_w \leq d_v$, then $d_u \geq d_w$.

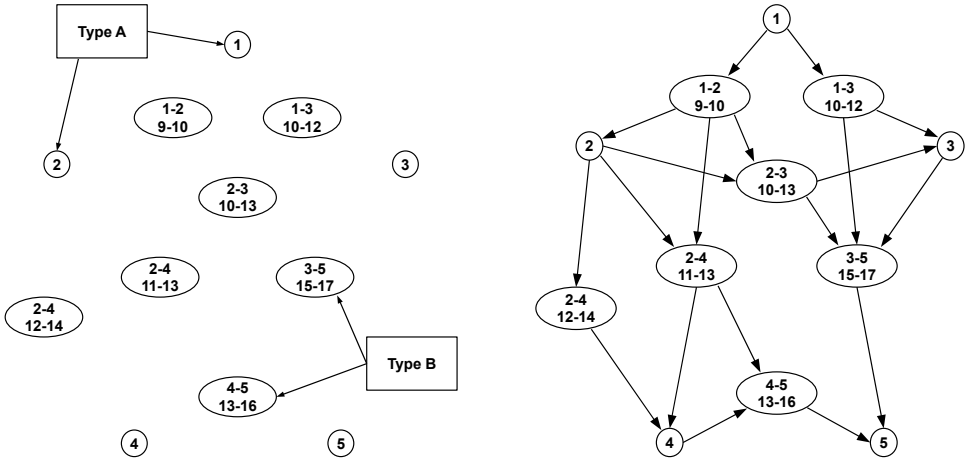

Fig. 2. Generation of nodes in the edge-converted model (left) and the complete edge-converted graph for the initial network from Fig. 1 (right)

d) For all type B nodes $w = (s_w, t_w, d_w, a_w)$, if $s_w = s_v$, $t_w = t_v$, and $a_u \leq d_w$, then $a_w \geq a_u$.

The complete edge-converted graph from the example in Fig. 1 is depicted in Fig. 2.

A route between two nodes (independent of their type) is defined as a usual path in the edge-converted graph. We start with some initial observations on the edge-converted graph.

Observation 2

1. *The connections between two type B nodes represent a transfer possibility between two elementary connections in the initial timetable.*
2. *If there exists a route between two type A nodes u and v, then there exists a route which only contains type B nodes as intermediate nodes, i.e., the only type A nodes are the source u and the target v. Thus, the length of a route is not dependent on the network size, but only on the number of the necessary transfers (compare with Observation 1).*
3. *The edge-converted graph has no cycles consisting only of type B nodes.*

We will use these observations in the applications below where we search some path between two type A nodes only via type B nodes.

Now we want to estimate the size of the new edge-converted model. Each node has been induced either by an initial station (type A) or by an initial elementary connection (type B). So the number of new nodes can be calculated as the sum of the initial nodes and edges. The number of new edges cannot be provided in an explicit form and does not only depend on the number of the initial edges or nodes but also on the connections' time stamps. Rules c) and d) from Step 4 in our construction filter out the "bad" transfer possibilities from the set of all possible transfers. The remaining edges between type B nodes represent the "good" transfer possibilities. Thus, the total number of edges in the edge-converted graph equals $2 \cdot \#\text{initial edges} + \#\text{good transfers}$.

Let us calculate an estimate for this number. Given a timetable with n stations, each station can be connected at most to $n-1$ stations in the original network. If we assume that there are at most k elementary connections between each pair of the initial stations, then, in the worst case, the edge-converted model contains $O(kn^3)$ edges connecting type B nodes with each other.

This, however, does not happen in realistic networks. Towards a better analysis, let us assume that the original network is of *constant degree* of at most d, i.e., every station has at most d ingoing and d outgoing connections to other stations. In this case we get $\leq d^2 n$ edges for connecting type B nodes and $\leq 2dn$ edges for connecting type A nodes to the type B nodes. Thus, the total size of the edge-converted graph is linear in the size of the original network. This is depicted in Table 2. As the table shows, regarding realistic networks, our model contains fewer nodes, but more edges than in the time-expanded model.

Table 2. A comparison of the size of the time-expanded and edge-converted models

Initial graph	Time-expanded model	Edge-converted model
Very dense networks		
#stations $= n$	$\leq 2kn^2$	$\leq n + kn^2$
#elementary connections $\leq kn^2$	$\leq 3kn^2 - n$	$\leq kn^3 + 2kn^2$
Constant-degree networks		
#stations $= n$	$\leq 2dn$	$\leq (d+1)n$
#elementary connections $\leq dn$	$\leq (3d-1)n$	$\leq (d^2+2d)n$

A possible drawback of our construction is that, unlike in the time-expanded approach, we can only incorporate a fixed time horizon into the edge-converted model. Thus for practical purposes, one has to define a fixed maximal travel time and adjust the time horizon accordingly to one or several days.

5 EAP and MNTP with the Edge-Converted Model

The common approach to solve EAP or MNTP in the time-expanded approach is to use the Dijkstra algorithm which consumes more than linear running time. For the edge-converted model we will describe below two algorithms for EAP and MNTP with only linear run-time. Moreover, our algorithms have the advantage of great simplicity as they implement variants of depth-first search and breadth-first search, respectively.

Our algorithms include a pre-processing step that has to be done only once. Let (s, t, d) be an EAP query. We need to find a route connecting the stations s and t, starting not earlier than at the given time d and providing the earliest arrival time at t. The main idea of our algorithm below is to use a usual depth-first search but starting from the target node t and moving backwards to the source s. This algorithm solves the EAP if we execute the next pre-processing routine on the edge-converted model:

1. First we delete all the edges constructed in step 3.a) in the section above. They are redundant for solving the EAP using the next algorithm.

2. Next, given some node v of type A or type B in an edge-converted graph, it has a set of ingoing edges $\{e_1, \ldots, e_k\}$. Every edge $e_i = (u_i, v)$ in this list has a start node u_i of type B, because there are no edges starting in type A nodes according to the previous step. We sort the set of ingoing edges for each node v in descending order by the arrival time stamps of their start nodes u_i.

5.1 EAP with the Edge-Converted Model

Algorithm 1 implements an inverse depth-first search on an edge-converted network constructed and pre-processed according to the above rules. The algorithm uses a stack S supporting the operations push(S, u) and pop(S, u) which push and pop a node u from the top of S. During the computation the algorithm maintains an array route$[u]$ which for each type B node u points towards a subsequent connection. At the end, the fastest route from s to t can be read off by following the pointers in the array, starting with route$[s]$.

Algorithm 1 EAP in the edge-converted model

Require: an EAP query (G, s, t, d_0)
where $G = (V, E)$ is an edge-converted network, $s, t \in V$ are the start and target node, and d_0 is the earliest departure time

1: **for all** $v \in V$ **do**
2: route$[v] \leftarrow$ nil
3: visited$[v] \leftarrow$ false
4: **end for**
5: push(S, t)
6: **while** S is not empty **do**
7: $u \leftarrow$ pop(S)
8: visited$[u] \leftarrow$ true
9: **if** u is a type A node **then** {this only happens if $u = t$}
10: $s_u \leftarrow u$
11: **else**
12: $u = (s_u, t_u, d_u, a_u)$ is a type B node
13: **end if**
14: **if** $s_u = s$ **then**
15: route$[s] \leftarrow u$
16: **return** route
17: **end if**
18: **for all** edges $e = (v, u)$ (in descending order according to the arrival time a of v) **do**
19: $v = (s_v, t_v, d_v, a_v)$ is a type B node
20: **if** visited$[v]$ = false and $d_v \geq d_0$ **then**
21: route$[v] \leftarrow u$
22: push(S, v)
23: **end if**
24: **end for**
25: **end while**
26: **return** there is no connection between s and t starting after time d_0

We state the correctness of the algorithm in the following theorem.

Theorem 3. *Algorithm 1 solves the EAP in the edge-converted model in linear time.*

Proof. Let G be an edge-converted network and let (s, t, d_0) be an EAP query. Let $u_1, \ldots, u_k$ be the set of predecessors of t, ordered according to the arrival time stamps of the type B nodes u_i (in ascending order). Each node u_i is the root of a depth-first search tree T_i consisting of all nodes which are visited from u_i in Algorithm 1. If the EAP instance (s, t, d_0) has a solution, then there exists a type B node $v_s = (s, v, d, a)$ such that $d \geq d_0$ and v_s is contained in one of the trees T_i for some $1 \leq i \leq k$.

We prove the correctness of Algorithm 1 by induction on the number i. First note that if s is reached in line 14, then

$$(v_s = \text{route}[s], \text{route}[\text{route}[s]], \ldots, u_i, t)$$

describes the unique path from v_s to t in T_i. In the base case $i = 1$, we have $v_s \in T_1$. But then we have found a route from s to t which arrives at t by the earliest possible connection in the network, and hence this route is optimal.

Let now $v_s \in T_i$ with $i \geq 2$. Aiming towards a contradiction, we assume that Algorithm 1 returns the route via the connections $(v_s, \ldots, u_i)$, but this is not the optimal solution. This means that there exists some node $v'_s = (s, v', d', a')$ such that $d' \geq d_0$ and there exists a route $(s, v'_s, \ldots, u_j, t)$ which leads to an earlier arrival at t. As the connections $u_1, \ldots, u_k$ have been ordered according to their arrival times, we have $j < i$. But then $v'_s \in T_j$ and Algorithm 1 would have returned the route $(s, v'_s, \ldots, u_j, t)$ by the induction hypothesis.

Therefore, Algorithm 1 is correct. It runs in linear time, because every type B node is visited at most once. □

In Theorem 3 the time is measured in terms of the input, i.e., in terms of the edge-converted network. As the size of the edge-converted graph is linear for constant-degree graphs (cf. Table 2), we immediately get:

Corollary 4. *For constant-degree graphs, Algorithm 1 solves the EAP in linear time measured in the size of the initial network.*

In comparison, using Dijkstra on constant-degree graphs only yields algorithms with running time $O(n \log n)$. In real networks, each station only has a limited number of connections per time interval. Therefore, real networks will usually be close to regular graphs.

5.2 MNTP with the Edge-Converted Model

To solve MNTP with the edge-converted model we can use breadth-first search (see Algorithm 2). Starting at the source node s, we find the minimum number of transfers route by reaching the target node t. Instead of a stack, Algorithm 2 uses a queue Q. The correctness of the algorithm can be shown by induction on the number of transfers in the optimal route from s to t. Thus we get:

Theorem 5. *Algorithm 2 solves the MNTP in the edge-converted model in linear time.*

Again, for regular networks we obtain a linear-time bound in terms of the original network:

Corollary 6. *For constant-degree graphs, Algorithm 2 solves the MNTP in linear time measured in the size of the initial network.*

Algorithm 2 MNTP in the edge-converted model

Require: an MNTP query (G, s, t)
where $G = (V, E)$ is an edge-converted network and $s, t \in V$ are the start and target node

```
 1: for all v ∈ V do
 2:     route[v] ← nil
 3:     visited[v] ← false
 4: end for
 5: if s = t then
 6:     return route
 7: end if
 8: enqueue(Q, s)
 9: while Q is not empty do
10:     u ← dequeue(Q)
11:     visited[u] ← true
12:     for all edges e = (u, v) do
13:         if visited[v] = false and v = (s_v, t_v, d, a) is a type B node then
14:             route[v] ← u
15:             if t_v = t then
16:                 route[t] ← v
17:                 return route
18:             end if
19:             enqueue(Q, v)
20:         end if
21:     end for
22: end while
23: return there is no connection between s and t
```

6 Experiments

To test the performance of the algorithms for EAP and MNTP in our model we implemented the time-expanded and edge-converted model. To solve EAP and MNTP in the time-expanded model we used Dijkstra with a priority queue, yielding time complexity $O(n \log n)$. These algorithms were tested against Algorithms 1 and 2 in the edge-converted model on randomly generated data.

The experiments were run on a PC with an Intel Core2Duo processor at 1.6 GHz and 2 GB RAM running Windows Vista. The algorithms were implemented in `C++` compiled with a VC8 compiler on the maximum optimization level. We used the Boost Graph Library [14] for all the graph, node, edge, and iterator classes.

6.1 Test Data Generation

We use a rectangle area to distribute a set of stations. The stations are randomly chosen in the area by assigning some x and y coordinates. Each station u gets some priority $p(u)$ in the interval $[0, 1]$. The priorities are uniformly distributed among all nodes. The distance $d(u, v)$ between two stations u and v is defined as the Euclidean distance between u and v in the plane.

For each pair of stations (u, v) we introduce elementary connections between u and v if $\frac{p(u) \cdot p(v)}{d(u,v)}$ is greater than some chosen threshold. We choose the number of these elementary connections proportional to $\frac{1}{d(u,v)}$. The time horizon is defined as $[0, 1439]$. For an elementary connection between u and v, we define

the travel time proportional to $d(u, v)$. The departure time at u is uniformly distributed over the time horizon taking into account that the arrival time must also fall within the time horizon.

6.2 Performance Analysis

We ran experiments with 20, 30, 40, 50, 60, and 70 stations. As the preprocessing time increases rapidly with the number of nodes, we could not perform experiments with many stations, for lack of hardware. For each experiment we generated the test data and counted the number of nodes and elementary connections in the initial network as well as in the time-expanded and edge-converted models. Then we solved EAP and MNTP by both approaches and measured the time. The results are shown in Table 3.

Table 3. Experimental comparison of EAP and MNTP in the time-expanded model (using DIJKSTRA with priority queue) and in the edge-converted model (Algorithms 1 and 2)

Initial graph		Time-expanded model				Edge-converted model			
#nodes	#edges	#nodes	#edges	EAP in sec.	MNTP in sec.	#nodes	#edges	EAP in sec.	MNTP in sec.
20	1048	2019	7020	11	15	1068	11434	4	34
30	2854	5336	18743	20	28	2884	52763	9	140
40	4141	7676	27016	48	64	4186	89643	15	213
50	7332	13035	46241	126	162	7382	221402	27	250
60	9140	16179	57438	143	180	9200	295835	36	321
70	10296	18108	64346	325	421	10366	351010	67	325

The results clearly show that Algorithm 1 solves EAP considerably faster than using DIJKSTRA in the time-expanded model, wheras for MNTP we obtain similar running times. Comparing the size of the two models it is apparent that the edge-converted approach reduces the number of nodes by a factor of 2 whereas the number of edges drastically increases. Instead of using an explicit stack, we implemented Algorithm 1 recursively which explains the better running time in comparison to Algorithm 2 which uses a queue.

7 Conclusion and Future Work

Our theoretical results as well as our practical evaluations show that using the edge-converted model might be an interesting alternative to the known algorithmic techniques for timetable information. This is mainly due to the very easy algorithms based on depth-first and breadth-first search. Particularly Algorithm 1 for EAP allows for a very simple and efficient recursive implementation.

However, our results here only provide a first basic study of this model and further investigation seems to be necessary. In particular, we would like to compare the edge-converted model with more sophisticated versions of the time-expanded approach which use a range of speed-up techniques for DIJKSTRA [12, 13, 15, 16]. An interesting question for further research is whether similar

speed-up techniques are applicable in the edge-converted model. It also appears interesting to compare our model with the time-dependent approach (cf. [11] for an extensive comparison of the time-dependent and time-expanded models). Finally, in future work we would like to test the edge-converted model on larger and preferably real networks.

References

1. G. S. Brodal and R. Jacob. Time-dependent networks as models to achieve fast exact time-table queries. *Electr. Notes Theor. Comput. Sci.*, 92:3–15, 2004.
2. D. Delling, T. Pajor, and D. Wagner. Engineering time-expanded graphs for faster timetable information. In *Proc. 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS)*, 2008.
3. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
4. L. Fleischer and M. Skutella. The quickest multicommodity flow problem. In *Proc. 9th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 36–53, 2002.
5. R. Geisberger, P. Sanders, D. Schultes, and D. Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *Proc. 7th International Workshop on Experimental and Efficient Algorithms (WEA)*, pages 319–333, 2008.
6. E. Köhler, K. Langkau, and M. Skutella. Time-expanded graphs for flow-dependent transit times. In *Proc. 10th Annual European Symposium on Algorithms (ESA)*, pages 599–611, 2002.
7. E. Köhler, R. H. Möhring, and H. Schilling. Acceleration of shortest path and constrained shortest path computation. In *Proc. 4th International Workshop on Experimental and Efficient Algorithms (WEA)*, pages 126–138, 2005.
8. M. Müller-Hannemann, F. Schulz, D. Wagner, and C. D. Zaroliagis. Timetable information: Models and algorithms. In *Proc. 4th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS)*, pages 67–90, 2004.
9. A. Orda and R. Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM*, 37(3):607–625, 1990.
10. E. Pyrga, F. Schulz, D. Wagner, and C. D. Zaroliagis. Experimental comparison of shortest path approaches for timetable information. In *Proc. 6th Workshop on Algorithm Engineering and Experiments and 1st Workshop on Analytic Algorithmics and Combinatorics (ALENEX/ANALC)*, pages 88–99, 2004.
11. E. Pyrga, F. Schulz, D. Wagner, and C. D. Zaroliagis. Efficient models for timetable information in public transportation systems. *ACM Journal of Experimental Algorithmics*, 12:1–39, 2008.
12. F. Schulz, D. Wagner, and K. Weihe. Dijkstra's algorithm on-line: An empirical case study from public railroad transport. *ACM Journal of Experimental Algorithmics*, 5:12, 2000.
13. F. Schulz, D. Wagner, and C. D. Zaroliagis. Using multi-level graphs for timetable information in railway systems. In *4th International Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 43–59, 2002.
14. The Boost Graph Library. Available from http://www.boost.org.
15. D. Wagner and T. Willhalm. Speed-up techniques for shortest-path computations. In *Proc. 24th Symposium on Theoretical Aspects of Computer Science*, pages 23–36, 2007.
16. D. Wagner, T. Willhalm, and C. D. Zaroliagis. Geometric containers for efficient shortest-path computation. *ACM Journal of Experimental Algorithmics*, 10:1–30, 2006.