A Modified *K*-Shortest Paths Algorithm for Solving the Earliest Arrival Problem on the Time-Dependent Model of Transportation Systems

Yang Yang, Shuai Wang, Xiaolin Hu, Jianmin Li, and Bingji Xu

***Abstract*—We are concerned with solving the *K*-earliest arrival problem on the timetable information-based public transportation systems. The problem is like this: given a departure time window at station A, find *K* best itineraries from station A to station B in terms of the earliest arrival time at station B. There are two typical models for the timetable information, the time-expanded and the time-dependent models. The *K*-earliest arrival problem can be solved on the time-expanded model by using the classical *K*-shortest paths algorithms for static networks. In this paper, we modified one of these algorithms proposed by Martins and Santos and applied it to the time-dependent model. The experimental results on the Chinese railway system show that the modified algorithm is nearly 3 times faster than its original version on the time-expanded model.**

***Index Terms*—timetable information, time-expanded model, time-dependent model, *K*-earliest arrival itineraries.**

I. Introduction

A typical problem in public transportation systems with timetable schedules is to offer appropriate itineraries to the passengers based on their preferences, such as earliest arrival, minimum number of transfers and shortest travel time and so on. The natural approach to solve these itinerary problems is to transform the timetable information into an appropriate graph, on which some shortest path algorithms can be used to find the optimal itineraries. There are two main approaches for modeling the timetable information: *the time-expanded* ([1], [3], [4]) and the *time-dependent* approaches ([1], [2], [3], [4], [5], [6]). The former approach constructs a digraph in which every node corresponds to a specific time event (departure or arrival) at a station and an edge between nodes represents either a specific train running from one station to the other or waiting within a station. The latter approach constructs a digraph in which each station is represented by a node and each train connection between two stations is represented by an edge. A dynamic link cost is assigned to each edge. The experimental results on real-world data presented in [1] show that the time-expanded approach turns out to be more robust for modeling more complex scenarios, whereas the time-dependent approach demonstrates better performance. Extended realistic time-expanded and time-dependent models with transfers between stations are discussed in [3].

However, so far, few researches have focused on solving the *K*-shortest paths problem on timetable information-based transportation systems. There are two classes of the shortest paths ranking problem: the unconstrained problem and the constrained problem. In the former no constraints are imposed on the path definition ([8], [9], [10], [11], [12], [13], [14], [17]), while in the latter the paths that satisfy specific restricts can be considered ([15], [16]). In this paper, we are concerned with the unconstrained problem. Two typical algorithms for solving the unconstrained problem refer to the Eppstein's algorithm ([8]) and the Martins and Santos' algorithm ([14], [17]). It is reported that the later outperforms the former on real-world data [14].

However, most of the *K*-shortest paths algorithm including the Eppstein's algorithm and the Martins and Santos' algorithm are designed purposely for static networks. The time-expanded model for the timetable information-based networks fits them well as its edge cost is constant and nonnegative. But the computing speed might be slow as a result of the large number of nodes and edges. It is known that the time-dependent model has fewer nodes and edges. If some *K*-shortest paths algorithm can be adapt to the time-dependent model, then the efficiency might be enhanced. In this paper, we explore this issue based on the Martins and Santos' algorithm (for short, MS algorithm hereafter), which results in a *K*-earliest arrival itineraries algorithm for the time-dependent model. The modified algorithm is compared with the original MS algorithm on the time-expanded model on real-world data. The experimental results show that the former is nearly three times faster than the latter.

The rest of the paper is organized as follows. Section II presents the time-expanded model, the time-dependent model and the original MS algorithm. Section III presents a modified version of the MS algorithm for the time-dependent

Manuscript received December 12, 2011. This work was supported in part by the National Natural Science Foundation of China under Grants 60805023, 90820305 and 61134012, National Basic Research Program (973 Program) of China under Grant 2007CB311003, Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology (TNList), and Research Foundation of the Easyway Company Limited.

Y. Yang and B. Xu are with the School of Information Technology, China University of Geosciences, Beijing, 100083 China.

S. Wang, X. Hu, and J. Li are with the State Key Laboratory of Intelligent Technology and Systems Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Computer Science and Technology, Beijing, 100084 China (e-mail: xlhu@tsinghua.edu.cn).

ISBN: 978-988-19251-9-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

network. The experimental results are reported in section IV and the conclusions are drawn in section V.

II. Preliminaries

In this section the time-expanded model and the time-dependent model for solving the earliest arrival problem (EAP) are first reviewed. The details can be found in [1], [2], and [3]. After that, an excellent *K*-shortest paths algorithm for static networks, the MS algorithm is briefly reviewed.

A. The Time-Expanded Model

In the time-expanded digraph, every node corresponds to a specific time event (departure or arrival) at a station. An edge between nodes belonging to the same station represents either transferring between trains inside a station (named *transfer edge*) or waiting within a station (named *stay edge*), and an edge between nodes belonging to different stations represents either transferring between stations (named *foot edge*) or a specific train running from one station to the other (named *train edge*). The cost of an edge (p,q) is $t_q - t_p$, where t_p and t_q are the time values associated with nodes *p* and *q*, respectively. The nodes and edges defined above are illustrated in Fig. 1. Note that here we present the simplified realistic version of the time-expanded model in [1] and take into account the transfers between stations in [3].

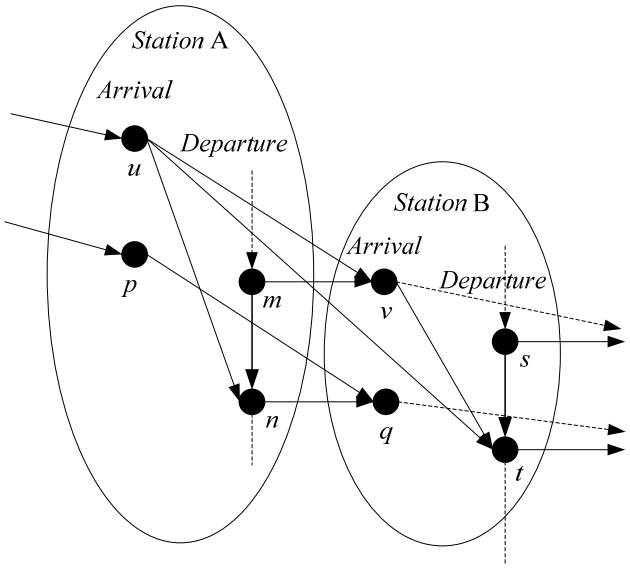

Fig.1. Illustration of the time-expanded model. Here, *u*, *p*, *v*, and *q* denote the arrival nodes; *m*, *n*, *s*, and *t* denote the departure nodes; (*u*, *v*), (*p*, *q*), (*m*, *v*), and (*n*, *q*) denote four train edges; (*m*, *n*) and (*s*, *t*) denote two stay edges; (*u*, *n*) and (*v*, *t*) denote two transfer edges; and (*u*, *t*) denotes a foot edge.

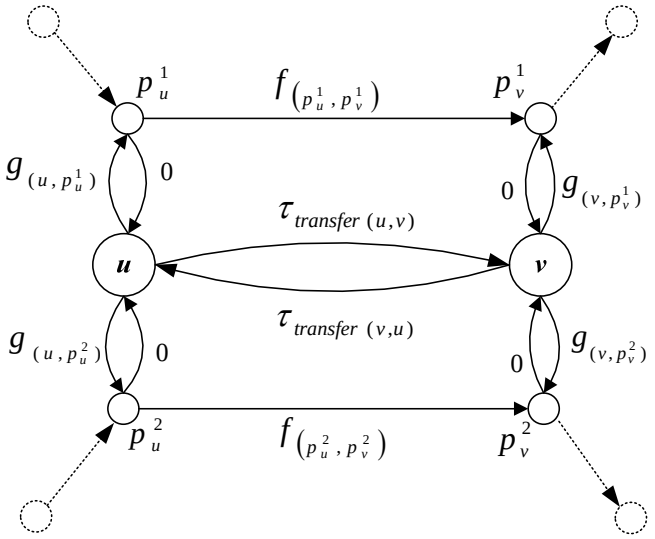

Fig.2. Illustration of the time-dependent model. Here, *u* and *v* denote two station nodes; p_u^j and p_v^j denote the route nodes, $j \in \{1, 2,...\}$; (u, p_u^1) denotes a boarding edge; (p_u^1, u) denotes a dropping edge; (p_u^1, p_v^1) denotes a route edge; and (*u*, *v*) and (*v*, *u*) denote two foot edges.

Because the edge costs in the time-expanded model are constant and nonnegative; the earliest arrival itinerary can be computed by using the Dijkstra's algorithm [7].

B. The Time-Dependent Model

The time-dependent model is based on the train-route digraph. A set of stations $S_0, S_1, ..., S_k (k > 0)$ forms a *train route* if one train departs from S_0 and passes through the consecutive stations $S_1, ..., S_k$ in turns. In the digraph each station is associated with a *station node*, denoted by *u*, *v*,…. The train routes passing through the station *u* are represented by *route nodes*, denoted by p_u^j, *j*=1, 2,…The model is illustrated in Fig. 2. There are four types of edges: *boarding edge*, *dropping edge*, *route edge*, and *foot edge*. The sets of these edges are denoted by *B*, *D*, *R*, and *F*, respectively. To solve the EAP, the edge costs are defined as follow:

1) An edge $(u, p_u^i) \in B$ has a dynamic cost specified by a function $g_{(u,p_u^i)}: T \to T$ such that $g_{(u,p_u^i)}(t)$ is the time at which p_u^i will be reached through the edge (u, p_u^i), given that *u* was reached at time *t* satisfying $t \le g_{(u,p_u^i)}(t) - \tau_{transfer}^u$, where $\tau_{transfer}^u$ is the constant transfer time in the corresponding station. Clearly, $g_{(u,p_u^i)}(t)$ also denotes the departure time at p_u^i along the route edge outgoing from p_u^i;
2) An edge $(p_u^i, u) \in D$ is assigned 0 cost;
3) An edge $(p_u^i, p_v^j) \in R$ has a dynamic cost specified by a function $f_{(p_u^i,p_v^j)}: T \to T$ such that $f_{(p_u^i,p_v^j)}(t)$ is the time at which p_v^j will be reached through the edge (p_u^i, p_v^j), given that p_u^i was reached at time *t*.
4) An edge $(u,v) \in F$ is assigned a constant cost $\tau_{transfer(u,v)}$ which denotes the transfer time needed for transferring from *u* to *v*.

An edge $e \in D \bigcup F$ is called a *static edge*. An edge $e \in B \bigcup R$ is called a *dynamic edge*.

A modified Dijkstra's algorithm ([1], [5]) can be used to solve the EAP on the time-dependent model if the following assumption holds.

Assumption 1. *Let u, v be any two station nodes and* p_u^i *and* p_v^j *be the corresponding route nodes such that* $(p_u^i, p_v^j) \in R$. *If* d_1, d_2 *are departure times from* p_u^i *and* a_1, a_2 *are the respective arrival times to* p_v^j, *then* $d_1 \le d_2 \Rightarrow a_1 \le a_2$.

ISBN: 978-988-19251-9-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE I
THE MS ALGORITHM

begin
determine a shortest tree in $G_1 = (V_1, E_1)$ and let p_1 be the shortest path from *s* to *t* in the shortest tree;
set *k*=1 and set $p = p_1$;
while (there is an alternative to *p*) and (*k*<*K*) do:
begin
determine $v_h(k)$, the first node of *p* such that $v_h(1)$ has more than one incoming edge; (A)
if $v_h'(k) \notin V_k$
begin
add $v_h'(k)$ to V_k;
update $I(v_h(1))$ as $I(v_h(1)) - \{(v_{h-1}(k), v_h(1))\}$ and $T(v_h(1))$ as $T(v_h(1)) - \{v_{h-1}(k)\}$; (B)
set $\pi_{v_h'(k)} = \min\{\pi_v + d(v, v_h(1)) \mid (v, v_h(1)) \in I(v_h(1))\}$; (C)
let $v_i(k) = v_{h+1}(k)$;
end
else let $v_i(k)$ be the first node on *p* after $v_h(k)$ such that $v_i'(k) \notin V_k$;
endif
for each $v_j(k) \in \{v_i(k), \ldots, t^{(k-1)'}\}$ do
begin
add $v_j'(k)$ to V_k;
update $I(v_j(1))$ as $I(v_j(1)) \cup \{(v_{j-1}'(k), v_j(1))\} - \{(v_{j-1}(k), v_j(1))\}$
and $T(v_j(1))$ as $T(v_j(1)) \cup \{v_{j-1}'(k)\} - \{v_{j-1}(k)\}$; (D)
set $\pi_{v_j'(k)} = \min\{\pi_v + d(v, v_j(1)) \mid (v, v_j(1)) \in I(v_j(1))\}$; (E)
end
obtain a new graph $G_{k+1} = (V_{k+1}, E_{k+1})$;
let *p* be the shortest path from *s* to t^k in G_{k+1};
set *k*=*k*+1;
end
end

This is not a strict assumption and it is easy to construct such a time-dependent network for any timetable information-based transportation systems. The correctness of the algorithm follows from that the functions *f* and *g* have nonnegative delays $(\forall t, f(t) \geq t, g(t) \geq t)$ and that the functions are nondecreasing.

C. A K-Shortest Paths Algorithm for Static Networks

We then briefly review the MS algorithm for ranking the shortest paths on static networks (see [14] for details). The algorithm constructs a sequence of growing graphs $G_1, G_2, ..., G_K$ such that the shortest path in G_k is the *k*-th shortest path in G_1, for $1 \leq k \leq K$. Given a static directed graph $G_1 = (V_1, E_1)$, an origin-destination pair (*s*, *t*) and an integer *K*, the MS algorithm for ranking the *K*-shortest paths is presented in Table I, where the notations are defined as follows:

$p_i = \{s \equiv v_0, \ldots, v_{h-1}, v_h, \ldots, v_{x-1}, v_x, \ldots, v_r \equiv t\}$: the *i*th shortest path from *s* to *t*;
$G_k = (V_k, E_k)$: the current graph, where *k*=1, 2, …, *K*;
$v_x(k)$: $v_x(k) \in V_k$, where *k*=1, 2, …, *K*-1;
$v_x'(k)$: the duplicate node of $v_x(k)$, where *k*=1, 2, …, *K*-1;
$t^{n'}$: the *n*-th duplicate node of *t*, where *n*=0, 1, …, *K*-1, $t^{0'} \equiv t$;
$v_{x-1}(k)$: the tail node of the edge on path *p* whose head node is $v_x(k)$, where *k*=1, 2, …, *K*-1;
$d(v_i, v_j)$: the cost of edge (v_i, v_j);
π_v: the distance label of node *v*, i.e., the shortest distance from *s* to *v*;
$I(v)$: the set of the incoming edges to *v*;
$T(v)$: the set of the tail nodes for edges belonging to $I(v)$.

III. THE K-EARLIEST ARRIVAL ITINERARIES ALGORITHMS

In the paper we are concerned with solving the *K*-earliest arrival problem (*K*-EAP): given a departure time window at station A, find *K* best itineraries from station A to station B in terms of the earliest arrival time at station B. The MS algorithm can be adapted for this purpose. In this section, we first introduce some specifics in implementing the MS algorithm on the time-expanded model for solving the *K*-EAP, and then present a modified version for the time-dependent model.

A. The K-Earliest Arrival Itineraries Algorithm on the Time-Expanded Model

As the edge costs in the time-expanded model are constant and nonnegative, the *K*-EAP on the time-expanded model can be solved by using the MS algorithm directly. However, due to the characteristics of the timetable information, some specific tricks are required.

By using the algorithm to rank the earliest arrival itineraries with given departure time window $[t_{d0}, t_{d1}]$, there is more than one origin node whose departure times are in

ISBN: 978-988-19251-9-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

$[t_{d0}, t_{d1}]$, as well as more than one target node. Let *S* denote the set of all the origin nodes and *T* denote the set of all the target nodes. Two virtual nodes are added to the digraph: *virtual origin node* $\tilde{S}$ and *virtual target node* $\tilde{T}$, which are connected to the origin nodes and the target nodes, respectively. $\tilde{S}$ has outgoing edges only, denoted by $(\tilde{S}, s)$, and the cost of the edge is $t_s - t_{d0}$, for each $s \in S$. $\tilde{T}$ has incoming edges only, denoted by $(t, \tilde{T})$, and the cost of the edge is 0, for each $t \in T$. Then the problem is transformed to find the *K*-earliest arrival itineraries from $\tilde{S}$ to $\tilde{T}$ and the MS algorithm can be applied now.

B. The K-Earliest Arrival Itineraries Algorithm on the Time-Dependent Model

As there exist dynamic edges on the time-dependent model, the MS algorithm cannot be applied directly. In the following, we modified the algorithm for the time-dependent model.

Let e_n denote the *n*-th elementary connection of a dynamic edge, $t_d(e_n)$ the departure time of e_n and $t_a(e_n)$ the arrival time of e_n. Let $a(v_x)$ denote the earliest arrival time label at node v_x.

The fundamental process of the modified algorithm is the same as that of the original MS algorithm. But because there are two types of dynamic edges, the route edges and the boarding edges, some steps need to be modified appropriately. First of all, the definition of the in-degree of a node is different from that in static networks. Second, when updating the incoming edges as well as the predecessor nodes of the node, which is in the given network and corresponds to the head node of a route edge on the current path, we remove the route edge and its tail node completely, as if the route edge was a static edge. Third, when updating the incoming edges of the node, which is in the given network and corresponds to the head node of a boarding edge on the current path, we remove part of the elementary connections of the boarding edge only: the specific elementary connection on the path and the other elementary connections whose arrival time is earlier than its arrival time. When updating the predecessor nodes of the node, we do not remove the tail node of the boarding edge unless all of the elementary connections of the edge have been removed. In addition, the algorithm to compute the earliest arrival time label of a node is different from the original MS algorithm. Specifically, the steps (A), (B), (C), (D), and (E) in the original MS algorithm (see Table I) should be modified as follows.

Step (A)

In calculating the in-degree of a head node of a boarding edge, the boarding edge corresponds to *n* degrees, where *n* stands for the number of the elementary connections of the edge. While in calculating the in-degree of a head node of a route edge, the route edge corresponds to 1 degree, as if the route edge was a static edge.

Step (B)

If $(v_{h-1}(k), v_h(1)) \in R$, $I(v_h(1))$ is updated as $I(v_h(1)) - \{(v_{h-1}(k), v_h(1))\}$ and $T(v_h(1))$ is updated as $T(v_h(1)) - \{v_{h-1}(k)\}$, the same as that in the original algorithm.

If $(v_{h-1}(k), v_h(1)) \in B$, different to the original MS algorithm, we first remove the elementary connection e_i on $(v_{h-1}(k), v_h(1))$, where $t_a(e_i) \le a(v_h(k))$, and leave $I(v_h(1))$ and $T(v_h(1))$ as they are unless all of the elementary connections of $(v_{h-1}(k), v_h(1))$ have been removed. If that happens, $I(v_h(1))$ is updated as

$$I(v_h(1)) - \{(v_{h-1}(k), v_h(1))\}, \quad (1)$$

and $T(v_h(1))$ is updated as

$$T(v_h(1)) - \{v_{h-1}(k)\}. \quad (2)$$

If $(v_{h-1}(k), v_h(1))$ is not a dynamic edge, this step is the same as that in the original algorithm.

Step (C)

The modified Dijkstra's algorithm in [1] and [5] is used to compute the earliest arrival time at $v_h'(k)$. If $v_h(k)$ is a route node, set

$$a(v_h'(k)) = \min\{t_a(e_i) \mid t_d(e_i) \ge a(v_x(k)), \; e_i \in (v_x(k), v_h(1)), (v_x(k), v_h(1)) \in I(v_h(1))\}. \quad (3)$$

Otherwise,

$$a(v_h'(k)) = \min\{a(v_x(k)) + d(v_x(k), v_h(1)) \mid (v_x(k), v_h(1)) \in I(v_h(1))\}. \quad (4)$$

Step (D)

If $(v_{j-1}(k), v_j(1)) \in R$, $I(v_j(1))$ is updated as

$$I(v_j(1)) \cup \{(v_{j-1}'(k), v_j(1))\} - \{(v_{j-1}(k), v_j(1))\}, \quad (5)$$

and $T(v_j(1))$ is updated as

$$T(v_j(1)) \cup \{v_{j-1}'(k)\} - \{v_{j-1}(k)\}. \quad (6)$$

If $(v_{j-1}(k), v_j(1)) \in B$, two cases need to be considered: $(v_{j-1}'(k), v_j(1))$ exists in the current graph and $(v_{j-1}'(k), v_j(1))$ does not exist. In the former case, first, we replace the cost function of the edge $(v_{j-1}'(k), v_j(1))$ with that of $(v_{j-1}(k), v_j(1))$. Second, we remove the elementary connection e_i on $(v_{j-1}(k), v_j(1))$, where $t_a(e_i) \le a(v_j(k))$. If all of the elementary connections of $(v_{j-1}(k), v_j(1))$ have been removed, then update $I(v_j(1))$ as

$$I(v_j(1)) - \{(v_{j-1}(k), v_j(1))\}, \quad (7)$$

and $T(v_j(1))$ as

$$T(v_j(1)) - \{v_{j-1}(k)\}. \quad (8)$$

In the latter case, first, we update $I(v_j(1))$ as

$$I(v_j(1)) \cup \{(v_{j-1}'(k), v_j(1))\}, \quad (9)$$

and $T(v_j(1))$ as

$$T(v_j(1)) \cup \{v_{j-1}'(k)\}. \quad (10)$$

Second, we remove the elementary connection e_i on $(v_{j-1}(k), v_j(1))$, where $t_a(e_i) \le a(v_j(k))$. If all of the elementary connections of $(v_{j-1}(k), v_j(1))$ have been removed, then update $I(v_j(1))$ as

$$I(v_j(1)) \cup \{(v_{j-1}'(k), v_j(1))\} - \{(v_{j-1}(k), v_j(1))\}, \quad (11)$$

and $T(v_j(1))$ as

ISBN: 978-988-19251-9-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

$$T(v_j(1)) \bigcup \{v'_{j-1}(k)\} - \{v_{j-1}(k)\}. \quad (12)$$

If $(v_{j-1}(k), v_j(1))$ is not a dynamic edge, this step is the same as that in the original algorithm.

Step (E)

The modified Dijkstra's algorithm in [1] and [5] is used to compute the label of the node $v'_j(k)$. If $v_j(k)$ is a route node, set:

$$a(v'_j(k)) = \min\{t_a(e_i) \mid t_d(e_i) \geq a(v_x(k)), \\ e_i \in (v_x(k), v_j(1)), (v_x(k), v_j(1)) \in I(v_j(1))\}. \quad (13)$$

Otherwise,

$$a(v'_j(k)) = \min\{a(v_x(k)) + d(v_x(k), v_j(1)) \mid \\ (v_x(k), v_j(1)) \in I(v_j(1))\}. \quad (14)$$

It is easy to see that the modifications presented above do not affect the correctness of the MS algorithm. Then we have the following theorem.

Theorem 1. *If Assumption 1 holds, then the paths determined by using the algorithm presented above on the time-dependent model are the K-earliest arrival itineraries.*

IV. EXPERIMENTS

In this section we compare the efficiencies of the two *K*-earliest arrival itineraries algorithms with real-world data of the Chinese railway system.

A. Data

We have used the timetable information of the Chinese railway system to generate the time-expanded and the time-dependent digraphs. The data is available from Feb. 23 to Mar. 16 of 2011. Not all the trains operate daily. Table II summarizes the characteristics of the graphs.

TABLE II
THE CHARACTERISTICS OF THE TWO MODELS

Model	Nodes	Edges	Average Number of Elementary Connections per Dynamic Edge
Time-Expanded	1544804	2663287	--
Time-Dependent	44461	99913	20.21

For detailed comparison, we divide the stations into *large stations* and *small stations*. A station is called a large station if on average over 50 trains pass through it every day; otherwise, it is called a small station. According to this criterion, there are 185 large stations and 2791 small stations. The queries are divided into five types, which are defined as follow：

1) L2L: from large station to large station;
2) L2S: from large station to small station;
3) S2L: from small station to large station;
4) S2S: from small station to small station;
5) Random: from any station to any station;

Each query consists of a departure station, an arrival station, the earliest departure time and the last departure time.

B. Experimental Setup

The algorithms are implemented in C++ and conducted on a server with Intel (R) Xeon (R) E5320 processor at 1.86GHz and 4GB of memory running Linux (kernel version 2.6.32).

For each query, for speeding up the algorithms, on any model we only label the nodes within 5 days after the earliest

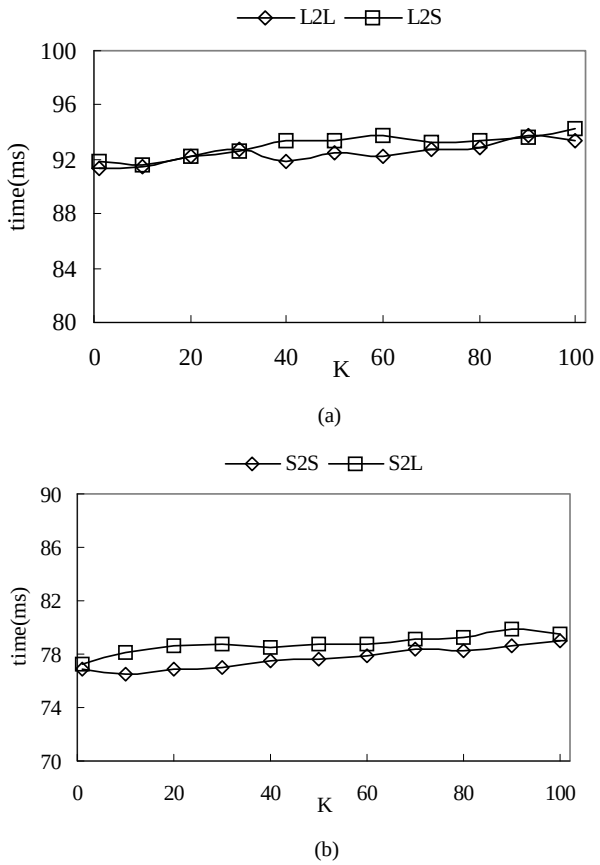

Fig. 3. Execution time of the MS algorithm on the time-expanded model for the (a) L2L, L2S and (b) S2S, S2L queries.

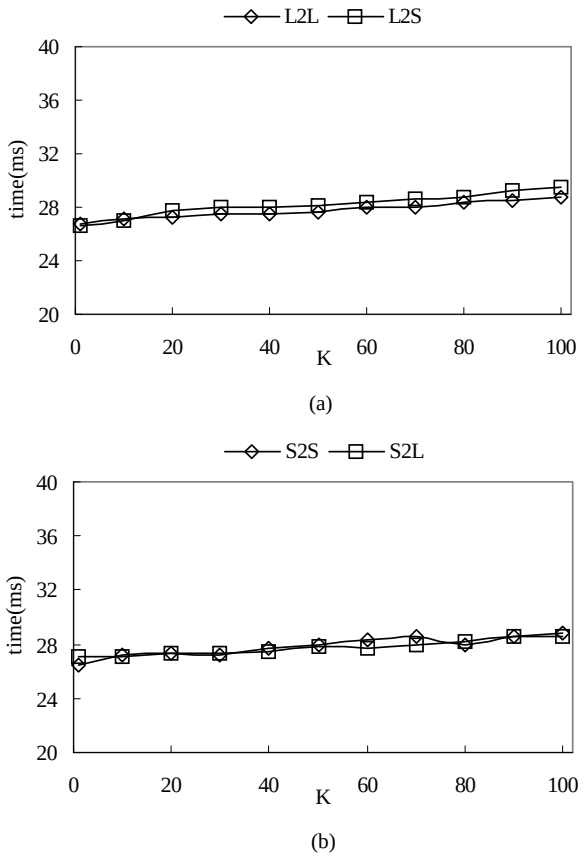

Fig. 4. Execution time of the revised MS algorithm on the time-dependent model for the (a) L2L, L2S and (b) S2S, S2L queries.

ISBN: 978-988-19251-9-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

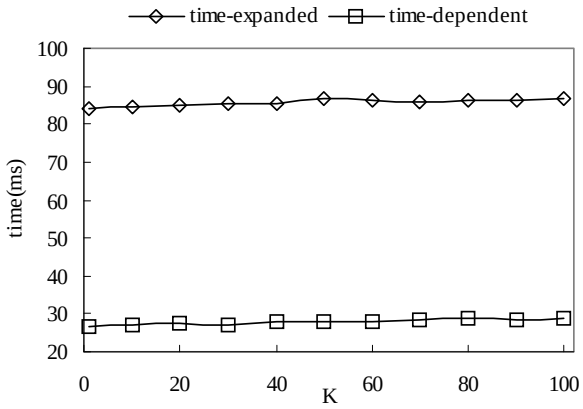

Fig. 5. Execution time of the two algorithms for the "Random" queries.

departure time. It will not affect the final results since our experimental results showed that the *K*-th earliest arrival itinerary never exceeded this period even for *K*=100.

C. Results

The five types of queries mentioned above are used to test the two algorithms, and each type includes 400 queries, for *K* = 1, 10, 20, …, 100, respectively.

Fig. 3 shows the execution time on the time-expanded model in responding to the queries of type L2L, L2S, S2S and S2L. First of all, Fig. 3(a) shows that the speed of the MS algorithm for the queries of type L2L is nearly the same as that for the queries of type L2S. Fig. 3(b) shows that the speed of the algorithm for the queries of type S2L is nearly the same as that for the queries of type S2S. By comparing Fig. 3(a) and Fig. 3(b), we can see that the MS algorithm for the queries departing from small stations runs faster than that for the queries departing from large stations. It is due to the fact that the departure nodes at a small station are usually fewer than those at a large station.

Fig. 4 shows the execution time of revised MS algorithm on the time-dependent model for the queries of type L2L, L2S, S2S and S2L. It is seen that the efficiency of the algorithm is very robust and the execution time for all types of queries are nearly the same.

From Fig. 3 (a) and Fig. 4(a), it is seen that the algorithm for the query types L2L and L2S on the time-dependent network is 3.3 times faster than that on the time-expanded network on average. From Fig. 3(b) with Fig. 4(b), it is seen that the algorithm for the query types S2S and S2L on the time-dependent network is 2.8 times faster than that on the time-expanded network on average.

Fig. 5 compares the execution time of the algorithms on the two models for the "Random" queries. It is seen that, for any model, the execution time for $0 < K \le 100$ is nearly the same as that for *K*=1. This result suggests the high efficiency of the MS algorithm. In addition, the modified *K*-earliest arrival itineraries algorithm on the time-dependent model is 3.1 times faster than the original MS algorithm on the time-expanded model on average.

V. Conclusions

To solve the *K*-earliest arrival problem for the timetable information-based transportation system, we modified an excellent *K*-shortest paths algorithm proposed by Martins and Santos for the time-dependent model. The experimental results on Chinese railway system show that the modified algorithm is nearly 3 times faster than the original algorithm on the time-expanded model.

Theoretically, both of the algorithms can be guaranteed to find the *K*-earliest arrival itineraries. However, neither algorithm can guarantee that all of the itineraries found are reasonable in reality due to the intrinsic deficiency of the time-expanded and time-dependent models. For example, an itinerary found by the algorithms may pass through a station twice though at different time. In the next step, we plan to integrate the rationality check of the results into the algorithms.

References

[1] E. Pyrga, F. Schulz, D. Wagner and C. Zaroliagis, "Efficient models for timetable information in public transportation systems," *Journal of Experimental Algorithmics*, vol.12, pp. 1-39, 2008.

[2] E. Pyrga, F. Schulz, D. Wagner and C. Zaroliagis, "Towards realistic modeling of time-table information through the time-dependent approach," in *Electronic Notes in Theoretical Computer Science*: *Proceeding of ATMOS Workshop 2003*, pp. 85-103.

[3] M. Müller-Hannemann, F. Schulz, D. Wagner and C. Zaroliagis, "Timetable information: models and algorithms," *Algorithmic Methods for Railway Optimization*, vol. 4359, pp. 67-90, 2007.

[4] D. Delling, K. Giannakopoulou, D. Wagner and C. Zaroliagis, "Timetable information updating in case of delays: modeling issues," *Technical Report 133, Arrival Technical Report*, 2008.

[5] G. Stolting Brodal and R. Jacob, "Time-dependent networks as models to achieve fast exact time-table queries," in *Electronic Notes in Theoretical Computer Science*: *Proceeding of ATMOS Workshop 2003*, 2004, pp. 3-15.

[6] A. Orda and R. Rom, "Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length," *Journal of the ACM*, vol. 37, no. 3, pp. 607-625, 1990.

[7] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269-271, 1959.

[8] D. Eppstein, "Finding the *K* shortest paths," *SIAM Journal on Computing*, vol. 28, no. 2, pp. 652-673, 1998.

[9] E. Q. V. Martins, M. M. B. Pascoal and J. L. E. Santos, "Deviation algorithms for ranking shortest paths," *International Journal of Foundations of Computer Science*, vol. 10, no. 3, pp. 247-262, 1999.

[10] E. Q. V. Martins, V. Martins, et al, "The optimal path problem," *Investigacao Operacional*, vol. 19, pp. 43-60,1999.

[11] E. Q. V. Martins, E. Queir, et al. (1999). Labeling algorithms for ranking shortest paths. *Technical Report, CISUC*. [Online]. Available: http://www.mat.uc.pt~/marta/ Publicacoes/ labeling. ps. gz.

[12] V. M. Jiménez and A. Marzal, "Computing the *K* shortest paths: a new algorithm and an experimental comparison," *Algorithm Engineering*, vol. 1668, pp. 15-29, 1999.

[13] V. M. Jiménez and A. Marzal, "A lazy version of Eppstein's *K* shortest paths algorithm," *Experimental and Efficient Algorithms*, vol. 2647, pp. 179-191, 2003.

[14] E. Q. V. Martins, J. L. E. Santos, et al, "A new shortest paths ranking algorithm," *Investigacao Operacional*, vol. 20, no. 1, pp. 47-62, 1999.

[15] J. Y. Yen, "Finding the k shortest loopless paths in a network," *management Science*, vol. 17, no. 11, pp. 712-716, 1971.

[16] E. Q. V. Martins and M. M. B. Pascoal, "A new implementation of Yen's ranking loopless paths algorithm," *4OR: A Quarterly Journal of Operations Research*, vol. 1, no. 2, pp. 121-133, 2003.

[17] J. A. Azevedo, J. J. E. R. S. Madeira, E. Q. V. Martins and F. M. A. Pires, "A shortest paths ranking algorithm," in *Proceedings of the Annual Conference AIRO'90*, Sorrento, 1990, pp. 1001-1011.

ISBN: 978-988-19251-9-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)