TRANSPORT
ISSN 1648-4142 / eISSN 1648-3480
2020 Volume 35 Issue 1: 26–36
https://doi.org/10.3846/transport.2020.12106

A COMPUTATION METHOD ON TIME-DEPENDENT ACCESSIBILITY OF URBAN RAIL TRANSIT NETWORKS FOR THE LAST SERVICE

Yao CHEN, Baohua MAO*, Yun BAI, Zhujun LI, Jimeng TANG

Key Laboratory of Transport Industry of Big Data Application Technologies for Comprehensive Transport, Beijing Jiaotong University, Beijing, China

Received 18 December 2016; revised 24 April 2017; accepted 16 May 2017

Abstract. Urban rail transit networks seldom provide 24-hour service. The last train is the latest chance for passengers. If passengers arrive too late to catch the last train, the path becomes inaccessible. The network accessibility thus varies depending on the departure time of passenger trips. This paper focuses on the computation method on the time-dependent accessibility of urban rail transit networks in order to facilitate the itinerary planning of passengers. A label setting algorithm is first designed to calculate the latest possible times for Origin–Destination (O–D) pairs, which is the latest departure times of passengers from the origins such that the destinations can be reach successfully. A searching approach is then developed to find the shortest accessible path at any possible departure times. The method is applied in a real-world metro network. The results show that the method is a powerful tool in solving the service accessibility problem. It has the ability to allow passengers to plan an optimal itinerary. Comparison analysis indicates that the proposed method can provide exact solutions in much shorter time, compared with a path enumeration method. Extensive tests on a set of random networks indicate that the method is efficient enough in practical applications. The execution time for an O–D pair on a personal computer with 2.8 GHZ CPU and 4GB of RAM is only 1.2 s for urban rail transit networks with 100 transfer stations.

Keywords: urban rail transit network, accessibility, itinerary planning, last train, timetable, label setting algorithm.

Introduction

Accessibility in transportation refers to the ease to reach destinations from origins on a network. It reflects the service level of the transportation systems to various locations. People in the locations with high accessibility can reach many other destinations quickly; while people in inaccessible places can reach fewer places within the same time. Therefore, accessibility can be defined as a binary indicator of whether a destination can be reached. At a deeper level, it is also an indicator of how convenient a destination to be reached, which can be measured by the travel cost of the shortest path to reach the destination.

In urban rail transit systems, accessibility depends on network configurations and service schedule of rail transit lines. There are always several travel paths between an Origin–Destination (O–D) pair on urban rail transit networks. Different paths indicate different travel times. Passengers prefer to choose the shortest path, which reflects the accessibility of O–D pairs.

Urban rail transit systems seldom provide 24-hour services. The service usually closes for system checking and repairing at night time. Before service closure, the last train is usually the latest chance for passengers. If passengers arrive at the platform of the boarding or transfer station too late to catch the last train, the path becomes inaccessible. So, the accessibility of a path is determined by the schedule of last trains and the departure time of trips. The increasing number of paths will become inaccessible when approaching the closure time. As a result, the shortest path between an O–D pair may vary with the departure time. If all paths are inaccessible, the O–D pair thus fails to be connected. Therefore, the network accessibility will be time-dependent before service closure time.

To illustrate the time-dependent service accessibility, Figure 1 depicts an example network, where the departure and arrival times at stations of the last several trains are indicated. There is an O–D pair from p to q with three paths. For a trip with a departure time 22:52, the path a is inaccessible since the last train has departed from the origin station. As a result, the path b becomes the shortest accessible path. When departing at 22:57, passenger

*Corresponding author. E-mail: *bhmao@bjtu.edu.cn*

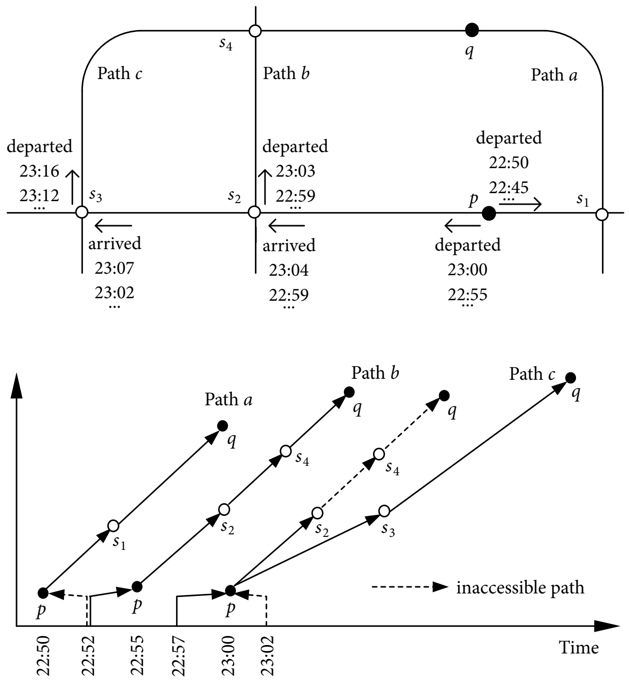

Figure 1. Illustration of time-dependent service accessibility

arrives too late to transfer at the station s_2 and only the path *c* is accessible. While for a trip with departure time 23:02, the destination turns to inaccessible eventually.

Passengers might to miss the last train if they do not know the time-dependent accessibility. They have to change their itinerary or even the travel mode on their trip. It severely brings inconvenience to passengers. In order to improve the service quality, it is vitally important to provide the accessibility information, i.e., how the shortest path varies according to the departure time and when the destinations will become inaccessible. At transfer stations, a delay of the arrival train or an early departure of the next connecting train may result in the inaccessibility of a path. The accessibility information therefore needs to be updated on-line. It requires a computation efficiency method.

The previous researches on service accessibility of transit network mainly concentrate on the path finding. The path finding algorithms on transit network can be divided into headway-based and schedule-based algorithms. The headway-based algorithms simplify that passenger waiting time at boarding or transfer station is an average value that depends on the service headway. The classic algorithms for road networks are applicable for headway-based services, including Dijkstra's algorithm (Dijkstra 1959), Bellman–Ford algorithm (Bellman 1958) and other algorithms (Zhan, Noon 1998; Sanders, Schultes 2006) for shortest path problem and Yen's algorithm (Yen 1971), Eppstein's algorithm (Eppstein 1998) and others (Martins *et al.* 1999; Martins, Pascoal 2003; Van der Zijpp, Catalano 2005) for *K*-shortest path problem. They are suitable for high-frequency transit networks, on which passengers do not care about schedule but only headway. However, because of the simplification on passenger waiting time, the headway-based algorithms are applied in static networks. The inaccessibility of a path due to the closure of services does not be taken into account. The time-dependent network accessibility cannot be computed by these algorithms.

The schedule-based algorithms are different from the headway-based algorithms due to the consideration of service schedules exactly. Passenger waiting time is determined by line schedules and the arrival time of passengers at the station. So, the schedule-based algorithms require a dynamic network description (Brodal, Jacob 2004). Different kinds of schedule-based algorithms are proposed on dynamic networks to find the most efficient paths at planned departure or expected arrival times, including dynamic programming algorithms (Cooke, Halsey 1966; Ziliaskopoulos, Mahmassani 1993), label setting and correcting algorithms (Tong, Richardson 1984; Ziliaskopoulos, Wardell 2000; Huang, Peng 2002; Huang, 2007; Zografos, Androutsopoulos 2008; Xu *et al.* 2012), revised Martins and Santos algorithm (Wang *et al.* 2016) and others (Friedrich *et al.* 2001). These schedule-based algorithms are necessary for transit systems with long headways, since passengers' path choice is not only affected by network configurations but also by schedules. The schedule-based algorithms can effectively verify the O–D accessibility at given departure times. However, these algorithms do not pay enough attention to the closure of services. The time-dependent accessibility before closure time is still not clear. More specifically, two problems below cannot be completely solved by the schedule-based algorithms. The last service for an O–D pair provided by the urban rail system cannot be computed directly. The relationship between the shortest accessible path and the departure time is still unknown.

With the consideration of service closure, Luo *et al.* (2010) proposes a computation method on the dynamic accessibility based on path enumeration. The accessibility of paths is examined according to train timetable. But paths between an O–D pair cannot be completely enumerated, especially in large-scale networks. An inaccurate result of O–D accessibility may be led if any paths are left out. Guo *et al.* (2015) set up a model to formulate the dynamic accessibility and design an analytical algorithm based on recursion equations. But the algorithm is not graph based and, therefore, it does not support to path finding. The algorithm is used for schedule planning but not for travel guidance.

Therefore, this paper aims to propose a novel method to compute the accessibility information of urban rail transit networks before service closure time in order to facilitate passenger itinerary planning. A label setting algorithm is first designed to calculate the latest possible time for O–D pairs, which is the latest departure time from the origin such that the destination can be reach successfully, and to find the accessible path at the latest possible time. An approach is built to search for the shortest accessible path at any possible departure times. The method can be used to generate trip information for passengers. The latest possible times inform passengers of the last service between O–D pairs and enables passengers to reach their destination successfully. The shortest path at different departure times not only facilitates passenger path planning and also departure time choices.

The remainder of this paper is organized as follows. In Section 1, problems of searching for the latest possible time and the time-dependent shortest path are presented. A label setting algorithm to calculate the latest possible time and an approach to find the shortest accessible path at any possible departure times are proposed in Section 2. In Section 3 a real case on Shenzhen metro network is conducted to investigate the performance of the proposed method. In latest section conclusions are presented.

1. Problem

1.1. Urban rail network description

An urban rail transit network usually consists of a number of bidirectional lines. Each line has a number of stations, which could be denoted by a node in the graph theory. An edge is a unidirectional connection between two adjacent nodes and each edge belongs to a transit rail line. Let $N\left(V_1 \cup V_2, E\right)$ denote an urban rail network, where: V_1 denotes the set of transfer nodes; V_2 the set of other nodes and E the set of edges; $e\left(v,u\right) \in E$ denotes the directed edge starting from node v and ending at node u.

To model the relationship of nodes and edges in a same transit line, $P\left(v\right)$ denotes the set of the edges pointing to the node v. All edges which are located in the same transit line with node v and with directions toward node v are included in $P\left(v\right)$. It means that the set $P\left(v\right)$ contains not only the edges whose ending nodes are node v, but also the edges which are not adjacent to node v. Similarly, $P\left(v,u\right)$ denotes the set of the edges pointing to the edge $e\left(v,u\right)$. A particular case is stated that the edge $e\left(v,u\right)$ is included in $P\left(v,u\right)$. Taken Figure 2 as example, $P\left(v\right)=\left\{e\left(p,w\right), e\left(w,v\right), e\left(u,v\right), e\left(q,u\right)\right\}$ and $P\left(w,v\right)=\left\{e\left(p,w\right), e\left(w,v\right)\right\}$.

There are timetables showing the departure and arrival times of trains at each edge. The time information is recorded:

– let $dep\left(j,v,u\right)$ denote the departure time of the jth-to-the-last train from edge $e\left(v,u\right)$, i.e. the departure time from the starting node v;
– let $arr\left(j,w,v\right)$ denote the arrival time of the jth-to-the-last train at edge $e\left(w,v\right)$, i.e. the arrival time at the ending node v;
– let $R\left(v,u\right)$ denote the running time of trains at edge $e\left(v,u\right)$;

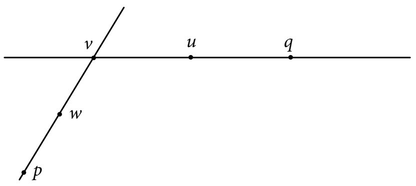

Figure 2. An example network

– let $H\left(w,v\right)$ denote the dwell time of trains at node v whose running direction is from node w to node v;
– let $Tr\left(w,v,u\right)$ denote the transfer walking time of passengers at transfer node v where the transfer direction is from $e\left(w,v\right)$ to $e\left(v,u\right)$.

1.2. Problem formulation

Given an O–D pair $\left(p,q\right)$, a path with one transfer is denoted by a sequence of nodes $g=\left\{v_0 \equiv p, v_1, ..., v_k, v_{k+1}, ..., v_t, ..., v_m \equiv q\right\}$, where: v_t denotes the transfer station. The accessibility of a path is indicated by whether the travel time of the path is infinite. A finite travel time represents the path is accessible while an infinite value represents not. Given a departure time t_p from the origin station p, the travel time of the path g is denoted as $T_g\left(t_p\right)$. It can be calculated by:

$$T_g\left(t_p\right)=W\left(v_0\right)+ \sum_{k=0}^{t-1}\left(R\left(v_k,v_{k+1}\right)+H\left(v_k,v_{k+1}\right)\right)+ Tr\left(v_{t-1},v_t,v_{t+1}\right)+W\left(v_t\right)+ \sum_{k=t}^{m-1}\left(R\left(v_k,v_{k+1}\right)+H\left(v_k,v_{k+1}\right)\right), \quad (1)$$

where: $W\left(v\right)$ denotes the waiting time of passengers at the platform of station v.

The waiting time at the origin station is determined by the departure time of the trip. If passengers depart from origin too late to catch up with the last train, there is no train for passengers to ride and the waiting time is set to infinite. Otherwise, passengers ride the first arrival train. The waiting time is the difference between the departure time of the first arrival train and the departure time of the trip, which can be expressed as:

$$W\left(v_0\right)=\begin{cases}\infty, & \text{if } dep\left(1,v_0,v_1\right)<t_p; \\ a, & \text{overwise;}\end{cases} \quad (2)$$

$$a=\min\left(dep\left(j,v_0,v_1\right)-t_p | dep\left(j,v_0,v_1\right)\geq t_p\right),$$

where: $dep\left(1,v_0,v_1\right)$ is the departure time of the last train from the origin station v_0.

Similarly, the waiting time at the transfer station is determined by the arrival time of the riding train, the transfer walking time to the platform of connecting line and the departure time of trains on the connecting line. If passengers arrive the platform later than the departure time of the last connecting train, the waiting time at the transfer station is set to infinite.

$$W\left(v_t\right)=\infty, \text{ if } dep\left(1,v_t,v_{t+1}\right)- Tr\left(v_{t\text{-}1},v_t,v_{t+1}\right)<W\left(v_0\right)+ \sum_{k=0}^{t-1}\left(R\left(v_k,v_{k+1}\right)+H\left(v_k,v_{k+1}\right)\right). \quad (3)$$

Therefore, the travel time of the path g depends on the departure time t_p. When the departure time reaches a high level, the travel time of the path becomes infinite. And then, the path becomes inaccessible. The latest departure time such that the path g is accessible, denoted by $t^L\left(g\right)$, can be expressed by:

$$t^L\left(g\right)=\max_{\forall T_g\left(t_p\right)\neq\infty} t_p. \tag{4}$$

Let G denote the set of paths g between $\left(p,q\right)$. If the departure time t_p is later than $\max t^L\left(g\right)$, all paths become inaccessible and the destination cannot be reached. So, "the latest possible time", which is defined as the latest departure time from origin station such that the destination station can be reached successfully, can be expressed by:

$$t_{pq}^L=\max_{\forall g\in G} t^L\left(g\right), \tag{5}$$

where: t_{pq}^L represents the latest possible time of the O–D pair $\left(p,q\right)$. Corresponding, the path, which is accessible at the latest possible time is named "the most effective path". The most effective path can be expressed by:

$$g=g^*|t^L\left(g^*\right)=\max_{\forall g\in G} t^L\left(g\right). \tag{6}$$

Due to service closure, the previous shortest path may turn to inaccessible and thus the shortest path is time-dependent. The time-dependent shortest path problem is to find the accessible path with minimum travel time at any possible departure times. For the time-dependent shortest path problem, the waiting time at station platform of passengers is set to an average value before the path turn to inaccessible. Indeed, urban rail systems always provide the high frequent of train services even before closure. The simplification is reasonable here and it is significantly effective in reducing the complexity of the model.

2. Solution method

In this section, a solution method is developed to compute the service accessibility of urban rail network. A label setting algorithm is first designed to calculate the latest possible time and to find the most effective path. Based on the most effective path, a searching approach is built to find the shortest accessible path at any possible departure time. The method hence includes two parts:

- a label setting algorithm for the latest possible time;
- a searching approach for the time-dependent shortest path.

2.1. Label setting algorithm

The label setting algorithm is modified according to the notion of the classic Dijkstra's algorithm (Dijkstra 1959). The Dijkstra's algorithm finds the shortest path from an origin node to any other nodes by a number of iteration steps. In the first step, the origin node is labelled, and its label value is the arrival time and set to zero. During the iterations, all edges, which start from the labelled node are considered. The arrival time at their end nodes is equal to that of the labelled node plus the travel time of the edges. The unlabelled node with the minimum arrival time will be chosen and labelled. Only one node will be labelled at every iteration step. Once all nodes are labelled, the algorithm is stopped.

With some modifications, the proposed algorithm calculates the latest possible times from all nodes to a single destination. The required modifications are described:

- *first*, instead of starting at the origin node, this algorithm moves backward from the destination node. The latest possible time will be recorded as the label value instead of the arrival time;
- *second*, in Dijkstra's algorithm, the label value is set at nodes. However, the node in the urban rail network requires the representation of the transfer time or the dwell time at stations. So, the label value cannot be set at nodes directly in urban rail networks. Some researches transform nodes to additional edges, namely, foot edges at transfer station and stay edges representing train waiting at a station, and use edge length to represent the transfer/dwell time (Wang *et al.* 2016). To avoid the expansion of the network, the latest possible time is set at directed edges instead of nodes. The latest possible time of an edge $e\left(v,u\right)$ represents the latest departure time from node v through edge $e\left(v,u\right)$ such that the destination node can be reached. In the first step, the edge is labelled whose end node is the destination node. And the label value is set to the departure time of the last train from the edge;
- *third*, at every iteration step, given a labelled edge (called e_1) and its adjacent edge (called e_2), where "adjacent" means that the ending node of e_2 is the starting node of e_1, the label value of the adjacent edge e_2 need to be determined. We first find the latest train whose arrival time at edge e_2 is less than the latest possible time of the labelled edge e_1. Then, the departure time of the train from edge e_2 is set as the label value. When a transfer is made between the two edges, the transfer walking time is added to the arrival time at edge e_2. All edges adjacent to edge e_1 are considered. And the edge with the maximum label value will be labelled;
- *fourth*, the accessibility of a path depends on the transfer accessibility (Kang *et al.* 2015). In order to reduce the iteration steps, only the edges starting from transfer nodes will be labelled during the iterations. And at every iteration step, all the edges pointing to the labelled edge are considered instead of the adjacent edges only.

Based on the modification, the algorithm is presented. The necessary parameters are first defined:

- $d\left(v,u\right)$ denotes the label value of the edge $e\left(v,u\right)$, which is the latest departure time from node v through $e\left(v,u\right)$ such that the destination can be reached;

- $d(v)$ denotes the label value of node v, which is the latest departure time from node v such that the destination can be reached; the equation $d(v) = \max d(v,u)$ $\forall e(v,u) \in E$ is followed;
- HP denotes the set of edges to be labelled; only the edges that start from transfer nodes will be labelled;
- $e(v^*, u^*)$ denotes the labelled edge at every iteration step, which is the edge with the maximum label value in HP;
- j^*th-to-the-last train is the latest train whose arrival time at transfer node v^* is earlier than the label value $d(v^*, u^*)$ minus transfer walking time; the train is the latest available train for passengers to reach the destination;
- $F(v,u)$ denotes the parent edge of $e(v,u)$.

The algorithm to calculate the latest possible times from all nodes to the destination node q is then described as follows.

The algorithm process is depicted in Figure 3.

***Step 1*:** set $d(v,u) = dep(1,v,u)$, $\forall e(v,u) \in P(q)$ and $d(v,u) = 0$, $\forall e(v,u) \notin P(q)$.

Insert all the edges starting from transfer nodes into the set HP, where $HP = \{e(v,u) \mid v \in V_1\}$.

***Step 2*:** if $d(w) \geq d(v,u)$, $w \in V_1 \cup V_2$, $\forall e(v,u) \in HP$, then go to Step 6.

***Step 3*:** search the edge with the maximum departure time from HP, and delete it, i.e.:

$$d(v^*, u^*) = \max_{\forall e(v,u) \in HP} d(v,u) \text{ and}$$

$$HP = HP - e(v^*, u^*).$$

***Step 4*:** for each edge $e(w, v^*)$, find the j^* th-to-the-last train such that:

$$j^* = \min\left(j \middle| arr(j,w,v^*) \leq d(v^*,u^*) - Tr(w,v^*,u^*)\right).$$

***Step 5*:** for each edge $e(v,u)$ such that $e(v,u) \in P(w, v^*)$, do

if $dep(j^*, v, u) > d(v,u)$,

then $d(v,u) = dep(j^*, v, u)$ and $F(v,u) = e(v^*, u^*)$.

Go to Step 2.

***Step 6*:** Output $d(w)$ as the latest possible time at nodes w.

The most effective path can be found from $F(v,u)$.

When the algorithm calculates the latest possible time for an O–D pair instead of all origin nodes, it is easy to know that the calculation can be completed once $d(p) \geq d(v,u)$, $\forall e(v,u) \in HP$, where p is the origin node. Moreover, the non-transfer nodes can be ignored in Step 1 and Step 5 to save the computation time.

Taken an urban rail network as an example, it is shown in Figure 4. There are three rail lines and three transfer nodes. The timetable of the last several trains is set. On all lines, the departure times from the terminal stations of the last train are all set to 23:00 and the headways between successive trains are all 4 min. The running times between two adjacent stations, the dwell times at stations and transfer walking times at all transfer directions are set to 2, 1 and 3 min, respectively. We calculate the latest possible time from the origin node p to the destination node q.

The iterations process of the algorithm is shown in Figure 5, where the time along edges is the value of $d(v,u)$:
- initially, the label values of edges that point to the node q are set to the departure time of the last train from the edges. The label values of other edges are set to 0. The edges starting from transfer nodes are inserted into HP:
- at the first iteration, as shown in Figure 5b, since the label values $d(v_1, q)$ is greater than $d(v_2, q)$, the edge $e(v_1, q)$ is labelled and then deleted from HP;
- then we search the latest available train, whose arrival time at node v_1 is less than $d(v_1, q)$ minus the transfer walking time. The 5th-to-the-last train and the last train at two directions on Line 2 are obtained;
- based on the timetable of the two trains, the label values of the edges on Line 2 are updated. The updated values are shown in Figure 5b;
- at the second iteration, $e(v_2, q)$ with greater value than $e(v_3, q)$ is newly labelled and deleted from HP;
- we repeat the iteration steps. The results of the second and third iteration are shown in Figure 5c and Figure 5d;

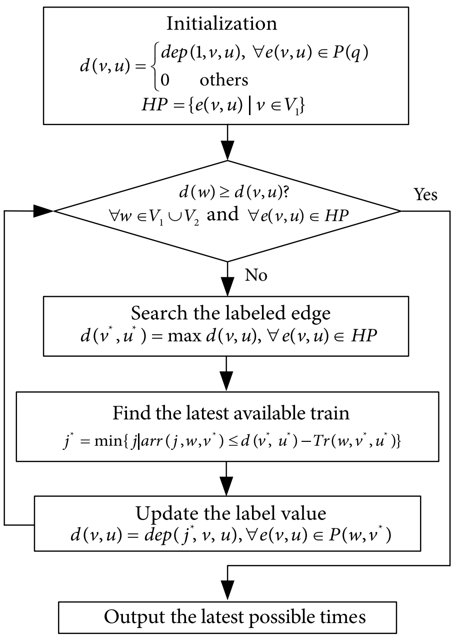
Initialization
$d(v,u) = \begin{cases} dep(1,v,u), \forall e(v,u) \in P(q) \\ 0 \quad \text{others} \end{cases}$
$HP = \{e(v,u) \mid v \in V_1\}$
$d(w) \geq d(v,u)$?
$\forall w \in V_1 \cup V_2$ and $\forall e(v,u) \in HP$
Yes
No
Search the labeled edge
$d(v^*,u^*) = \max d(v,u), \forall e(v,u) \in HP$
Find the latest available train
$j^* = \min\{j \mid arr(j,w,v^*) \leq d(v^*,u^*) - Tr(w,v^*,u^*)\}$
Update the label value
$d(v,u) = dep(j^*,v,u), \forall e(v,u) \in P(w,v^*)$
Output the latest possible times

Figure 3. Flow chart of the algorithm

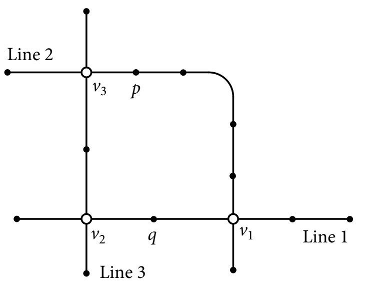

Figure 4. Structure of the example transit network

– as shown by Figure 5d, the edge with the maximum label value in *HP* is $e(v_3,p)$ and its label value is $d(v_3,p)=22{:}47$. The value is less than the label values of node p_1 22:50. Hence, the algorithm is stopped.

Theorem 1. The label setting algorithm is correct for calculating the latest possible times.

Proof. The algorithm is correct, really for the same reason as Dijkstra's algorithm (Dijkstra 1959). The validity of the algorithm is proved by the induction. The induction hypotheses are founded on the premises that: (1) the departure time $d(p)$ of node *p* is latest after the algorithm is stopped; (2) the departure time $d(v,u)$ is the possible departure time such that the destination can be reach from node *v* through edge $e(v,u)$.

To prove the hypothesis (1), recall that the algorithm initially set the last train as the latest available train, and at each iteration step the algorithm find the latest train that can catch the labelled edge at transfer node. According to Bellman's principle of optimality, it shows that the departure time $d(p)$ is the latest to the destination node at all iteration steps.

Moreover, the algorithm is stopped when $d(p)\geq d(v,u)$, $\forall e(v,u)\in HP$. The departure time of node *p* cannot be more than $d(p)$ any more since the departure time of the node will be less than $d(v,u)$, $\forall e(v,u)\in HP$ if the algorithm continues. This result indicates that the departure time $d(p)$ of node *p* is latest after the algorithm is stopped.

The hypothesis (2) is then proved. For each edge $e(v,u)$, $d(v,u)=dep(j^*,v,u)$ and $e(v,u)\in P(w,v^*)$. So, passengers starting from edge $e(v,u)$ at the departure time $d(v,u)$, the j^*th-to-the-last train is caught and the train can get to the transfer node v^*. Since $arr(j^*,w,v^*)\leq d(v^*,u^*)-Tr(w,v^*,u^*)$, passengers are able to transfer into the connecting train and then get to the node u^*. Notice that $e(v^*,u^*)$ is the parent edge of $e(v,u)$. It means that the parent edge can be reached after making one transfer. Therefore, after the update of the edge $e(v,u)$, its parent edge also can be reached until the destination node *q* is reached.

Taken together, the departure time $d(p)$ of node *p* is the latest possible time such that the destination node can be reach from node *p*.

Theorem 2. If the Fibonacci heap data structure is used to store $d(v,u)$ for every edge (v,u) in *HP*, then the time complexity of the algorithm for an O–D pair is $(O(m_1+P)\cdot T)$, where m_1 is the number of the edges starting from transfer nodes, *T* the number of the transfer directions in the network, *P* the maximum times of finding the latest available train for each transfer directions.

Proof. In order to obtain the time complexity of the algorithm, recall that the set *HP* involves Step 1 that insert edges into *HP*, Step 3 that search for the labelled edge and Step 5 that update the value $d(v,u)$. If Fibonacci heap data is used, the amortized times of insertion and update operation are both $O(1)$, and search operation can be done in amortized time $O(\log_2(m_1))$ (Fredman, Tarjan 1984). Since *HP* contains m_1 edges, it is obvious that the total time for Step 1 is $O(m_1)$, and Step 3 is $O(m_1\cdot\log_2(m_1))$.

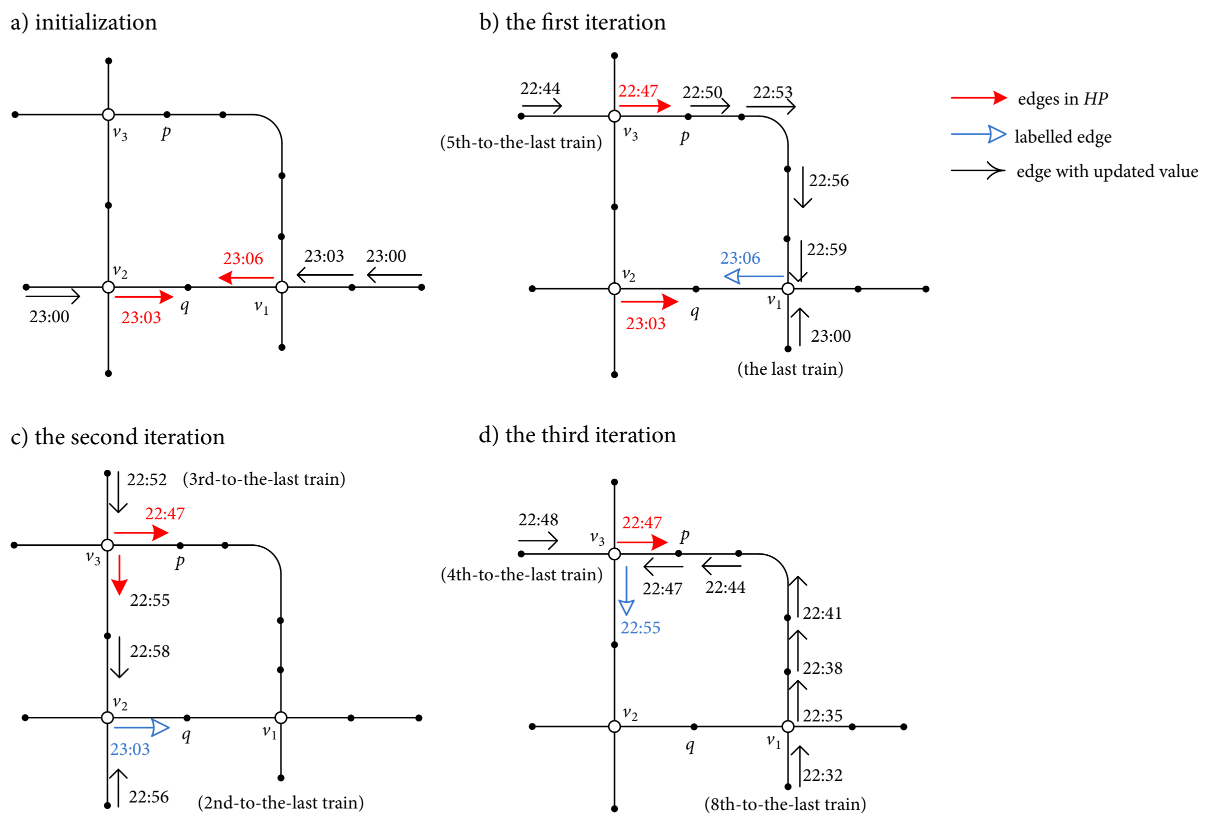

Figure 5. Results of the iterations

The total time for Step 5 is $O(m_1 \cdot T)$ since we at most need to examine m_1 edges for each transfer and the number of transfers involved is at most T.

As for Step 4 that find the latest available train, the time complexity is at most $O(P)$ for each transfer. The total time is $O(P \cdot T)$. Therefore, the total time for the algorithm is:

$$O\left(m_1 + m_1 \cdot \log_2(m_1) + m_1 \cdot T + P \cdot T\right) = O\left((m_1 + P) \cdot T\right).$$

2.2. Searching approach

A searching approach is proposed to find the shortest accessible path at any possible departure times. Since the inaccessibility of paths causes the time-dependent shortest path, the approach requires the K-shortest paths and examines the accessibility of the paths.

It should be noted that the waiting time at origin and transfer stations are simplified to an average value. As a result, the travel time of paths is nonnegative and constant. The urban rail transit network is considered as a static network. The K-shortest paths thus can be found by the classic algorithms (Yen 1971; Martins, Pascoal 2003; Van der Zijpp, Catalano 2005). Moreover, since the K-shortest paths do not vary in static networks, they often can be found and stored offline. In this paper, the algorithm developed by Martins and Pascoal (2003) is applied to find the K-shortest paths.

To examine the accessibility of the K-shortest paths, the latest possible times for paths are calculated. The label setting algorithm is used by treating paths as networks. When applied for a path, the algorithm degenerates into a recursion algorithm in effect.

Before presenting the approach, a lemma is introduced first. Let g_i denote the ith shortest path in the static network. Without loss of generality, the most effective path is represented by the kth shortest path. Let $t^L(g_i)$ denote the latest possible time of the ith shortest path.

Lemma. For paths g_i and g_j, if $i < j$ and $t^L(g_i) < t^L(g_j)$, the travel time of the path g_j is only shorter than that of the path g_i at departure time $t^L(g_i) < t_p \le t^L(g_j)$. If $i < j$ and $t^L(g_i) \ge t^L(g_j)$, the travel time of the path g_j is longer than the path g_i at any departure times.

According to the lemma, it is obvious that the path g_{k+1} cannot become the shortest path at any departure times. The paths with longer travel time than the most effective path can be ignored. Therefore, the searching approach, based on the most effective path, is proposed as follow:

***Step 1*:** if $k = 1$, then the shortest path is path g_k when $t_p \le t^L_{pq}$, go to Step 7;

***Step 2*:** calculate the latest possible time of path g_1. The shortest path is path g_1, when $t_p \le t^L(g_1)$;

***Step 3*:** set $i = 1$ and $j = 2$;

***Step 4*:** if $k = j$, then the shortest path is path g_k when $t^L(g_i) < t_p \le t^L_{pq}$, go to Step 7;

***Step 5*:** calculate the latest possible time of path g_j; if $t^L(g_j) > t^L(g_i)$, then the shortest path is path g_j, when $t^L(g_i) < t_p \le t^L(g_j)$ and set $i = j$;

***Step 6*:** set $j = j + 1$. Go to Step 4.

***Step 7*:** output the shortest path at different departure times.

3. Case studies

A practical application on the Shenzhen metro network is undertaken to illustrate the effectiveness of the proposed method. On the metro network, the latest possible times and the time-dependent shortest paths between O–D pairs are computed. A comparison analysis is then conducted to confirm the correctness of the method. In addition, the execution time of the method on a set of random rail transit network is tested to validate the computation efficiency.

3.1. Practical application

The Shenzhen metro network is composed of 5 bidirectional lines and 118 stations, including 13 transfer stations. The total length of Shenzhen metro network is 178 km. The network topology is illustrated in Figure 6, in which the arrows on the lines indicate up-directions.

The train timetable is provided by the operator of Shenzhen metro. The departure times of the last trains from terminal stations are 23:00 on all lines. Transfer walking time in transfer stations is assumed to be constant, which is set as 20% longer than average value to apply to most passengers. It ranges from 3 to 5.5 min varying with different transfer directions.

To illustrate the effectiveness of the proposed method on trip information generation, two O–D pairs from Changlingpi to Yangnan and to Guomao are taken as examples. The latest possible times and the accessible paths for the two O–D pairs are calculated by the label setting algorithm. It is implemented on a personal computer with 2.8 GHZ CPU and 4GB of RAM. The computational time of the algorithm is less than 0.2 s for one execution in this case. The results are shown in Figure 7.

The results indicate that the latest possible time of an O–D pair is not necessarily the departure time of the last train from the origin station. The latest possible time from Changlingpi to Yangnan is 22:57, and the latest available train for passengers is the 5th-to-the-last train. Passengers on the train have enough time to transfer into Line 4 at Shenzhen North. But passengers on the 4th-to-the-last train might to fail to transfer at the station. Hence, it is necessary to provide the latest possible times of O–D pairs for passengers. They are really useful to enable passengers using the urban rail system to reach their destination successfully.

Then the shortest paths are searched at different departure time between O–D pairs. The execution time for an O–D pair is less than 0.01 s on average. For the O–D pair from Changlingpi to Yangnan, the most effective path is the shortest path. So, the time-dependent shortest path can be concluded in Table 1.

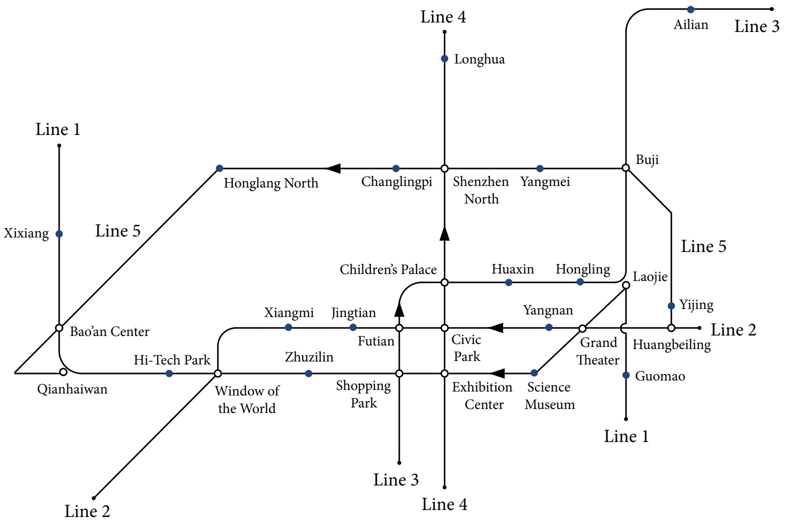

Figure 6. Network topology of Shenzhen metro

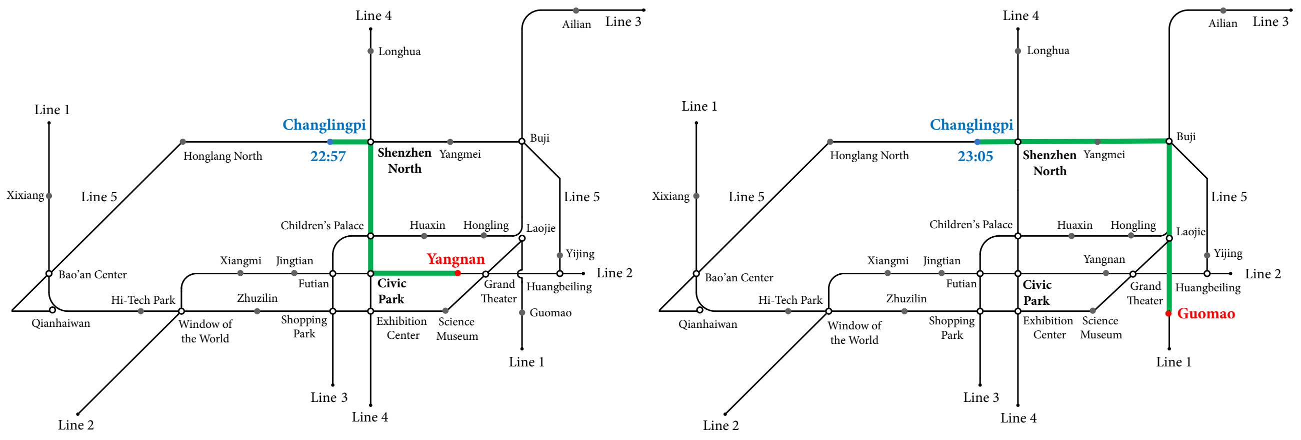

Figure 7. Results of the latest possible times and the most effective paths

Table 1. The time-dependent shortest path from Changlingpi to Yangnan

Departure time [h:min]	The shortest path	Travel time [min]
22:57	Changlingpi → Shenzhen North → Civic Park → Yangnan	38.5

Table 2. The time-dependent shortest path from Changlingpi to Guomao

Departure time [h:min]	The shortest path	Travel time [min]
22:57	Changlingpi → Shenzhen North → Exhibition Center → Guomao	42.5
22:57…23:05	Changlingpi → Buji → Laojie → Guomao	52.5

Moreover, based on the latest possible times from an origin station to all other destination stations, the accessible stations from the origin station at a given departure time can be obtained. To get the information, the algorithm is executed $n-1$ times, where n is the number of stations in the network. The computational cost is about 7 s in this network. The accessibility of Changlingpi to other stations is shown in Figure 8.

The accessible stations from Changlingpi at 23:03 are marked by green in Figure 8. It is useful for passengers to choice their trip destination, especially in the situation that the planned destination is inaccessible by the rail system. For example, Yangnan is inaccessible for passengers departing from Changlingpi at 23:03, but the alternative destination can be Laojie or Hongling, which is near from Yangnan. Moreover, the failure of transfer connection can be identified by the inaccessible stations. The missed transfer connections are illustrated in Figure 7 by curves. This information can be provided on the electronic board at each station. It is very beneficial to passengers' trip at late night.

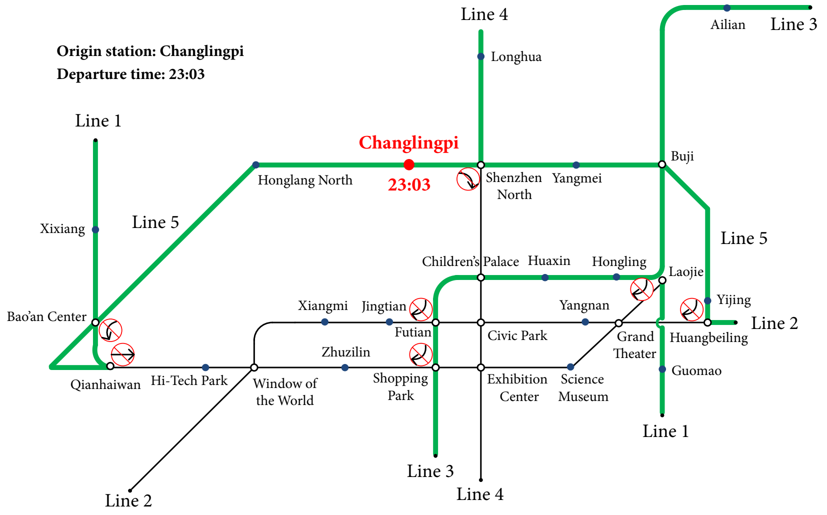

Figure 8. The accessibility of Changlingpi to other stations

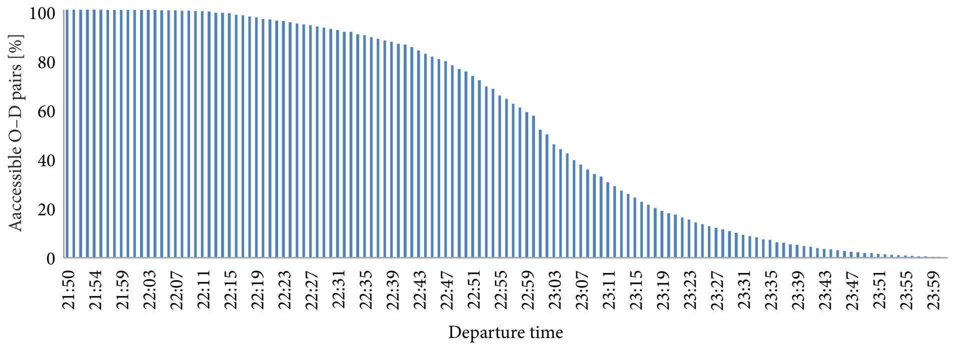

Figure 9. The percentage of accessible O–D pairs varying with departure times

In addition to passengers' trip guidance, the method also can be used for operators in the analysis of the time-dependent network accessibility. The percentage of accessible O–D pairs on the network varying with departure times is shown in Figure 9. An O–D pair firstly becomes inaccessible at 21:56 and about half of O–D pairs turn to inaccessible after 23:01. It indicates the level of service provision. And it also is the basis of the improvement of service accessibility.

For the O–D pair from Changlingpi to Guomao, the most effective path is the 4th shortest path. The time-dependent shortest path is given in Table 2. The relationship of the minimum travel time varying with the departure time is revealed. It enables passengers to balance the departure time and travel time. It is clear that if departing earlier than 22:57, passengers can spend less travel time.

3.2. Comparison analysis

To justify the label setting algorithm, a comparison analysis is conducted with the path-based method proposed by Luo *et al.* (2010). The path-based method is applied to find the latest possible time and the most effective path. Since paths between an O–D pairs cannot be completely enumerated, we search the effective paths whose travel time is less than 1.5 times of the minimal travel time between O–D pairs. And at least 10 paths are listed for each O–D pair. Then, we calculate the latest possible times of these paths.

On the Shenzhen metro network, same results are obtained by the two methods for most O–D pairs. However, results are different for about 1.13% of all O–D pairs. The O–D pairs are mainly between stations on Line 1 and Line 5. Taken the O–D pair from Hi-Tech Park to Yijing as an example, the different result is illustrated in Table 3.

It is obvious that the path found by the label setting algorithm is excluded by the path-based method since the travel time of the path is too long. Indeed, the path is still a choice for passengers before service closure time. So the path-based method may leave out accessible paths due to the incomplete enumeration.

Table 3. the result obtained by two methods

Methods	The latest possible time [h:min]	The most effective path	Travel time [min]
Path-based method	22:39	Hi-Tech Park → Laojie → Buji → Yijing	72.5
Label setting algorithm	22:46	Hi-Tech Park → Bao'an Centre → Yijing	79.5

To confirm the correctness of the proposed algorithm, the path-based method is executed again and more paths are examined. We list the first 20th paths for each O–D pair and calculate the latest possible times. In this experiment, same results are obtained for all O–D pairs. Since no path is left out by the label setting algorithm, it can be concluded that the label setting algorithm always can calculate the latest possible time and obtain the latest service exactly.

From the view of computation efficiency, the execution time of the path-based method for all O–D pairs is 68 s, while the label setting algorithm only uses 7 s to accomplish it. The label setting algorithm is more efficient than the path-based method. That is because the calculation of the latest possible times for so many paths is time-consuming.

3.3. Computational test

To further evaluate the computation efficiency, the label setting algorithm is tested on a set of random transit networks with increasing number of transfer stations: from 12 to 500 transfer stations. To better emulate actual rail transit networks, the number of stations on the tested networks is set depending on the number of transfer station, for example, a network with 30 transfer stations has at least 150 stations. The train timetable is also generated randomly. The latest possible time for a random O–D pair is computed. Each test is performed for 10 times randomly and the average computation time is recorded. The computation times on different networks with varying number of transfer stations are illustrated in Figure 10.

The results indicate that the computation times increase quadratically with the number of transfer stations. The results confirm the analysis in the time complexity of the algorithm. The algorithm is considerably efficient enough in the practical application. In more details, the computation times are less than 0.7 s for the test networks with 50 transfer stations, which is much larger than most real-world networks. Even for those with 100 transfer stations, the computation times are still about 1.2 s.

To illustrate the relationship of the computation time with the number of all stations, the computation times with the same number of transfer stations but different number of all stations are tested, which is shown in Figure 11.

The computation times are nearly the same for different networks with the same number of transfer stations. The results suggest that the computation cost is influenced by the number of transfer stations but not by the number of all stations. It is a nice characteristic in practical applications.

Conclusions

In urban rail transit systems, the closure of train services leads to the time-dependent accessibility. It may bring inconvenience to passengers. In such case, the accessibility information should be provided to passengers in order to improve the day-end service quality.

This paper proposed a method to compute the time-dependent accessibility of urban rail transit networks. First, a label setting algorithm is designed to calculate the latest possible time and the accessible path at the departure time. Then a search procedure is developed to find the shortest path at different departure time. The main contribution of the method attributes to the full consideration of service accessibility before closure time.

The proposed method has been applied on the Shenzhen metro network. The results indicate that it is a powerful tool in solving the service accessibility problem in the metro network. It provides passengers with the latest service of O–D pairs and the shortest path at different departure time. The information not only facilitates passengers the path optimization and also departure time choices. It moreover has applications in the analysis of service accessibility.

Comparison analysis indicates that the proposed method always can provide exact solutions in much shorter time, while the correctness of the path-based method is dependent on the number of listed paths. Extensive experiments on a set of random rail transit network show that the proposed method is efficient enough to calculate the accessibility information. It implies great potential of this method in a real-world network.

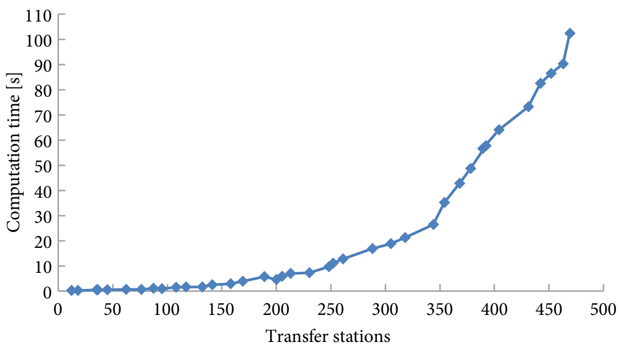

Figure 10. Relationship of the computation time with the number of transfer stations

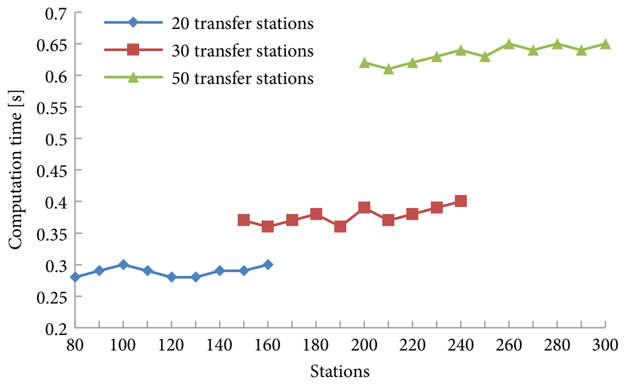

Figure 11. Relationship of the computation time with the number of all stations

Acknowledgements

The authors thank Dr. Peng Du and Dr. Qi Xu for helpful comments on drafts of this paper.

The authors also thank the referees for providing constructive comments and help in improving this paper.

Funding

This research is supported by the Fundamental Research Funds for the Central Universities (Grant No 2017YJS120) and the National Natural Science Foundation of China (Grant No 71971021).

Contribution

Yao Chen constructed the model formulation, designed the solution method, conducted the empirical case experiment, and drafted the paper.

Baohua Mao proposed the conception of the work and revised the paper.

Yun Bai thoroughly reviewed and improved this paper and polished the language.

Zhujun Li participated in the design of the label setting algorithm.

Jimeng Tang participated in the coding the algorithm in the case study.

All authors have discussed and contributed to the manuscript.

All authors have read and approved the final manuscript.

Disclosure statement

The authors of this paper confirm that we do not have any competing financial, professional or personal interests that would influence the performance or presentation of the work described in this manuscript.

References

Bellman, R. 1958. On a routing problem, *Quarterly of Applied Mathematics* 16: 87–90. https://doi.org/10.1090/qam/102435

Brodal, G. S.; Jacob, R. 2004. Time-dependent networks as models to achieve fast exact time-table queries, *Electronic Notes in Theoretical Computer Science* 92: 3–15. https://doi.org/10.1016/j.entcs.2003.12.019

Cooke, K. L.; Halsey, E. 1966. The shortest route through a network with time-dependent internodal transit times, *Journal of Mathematical Analysis and Applications* 14(3): 493–498. https://doi.org/10.1016/0022-247X(66)90009-6

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs, *Numerische Mathematik* 1(1): 269–271. https://doi.org/10.1007/BF01386390

Eppstein, D. 1998. Finding the *K* shortest paths, *SIAM Journal on Computing* 28(2): 652–673. https://doi.org/10.1137/S0097539795290477

Fredman, M. L.; Tarjan, R. E. 1984. Fibonacci heaps and their uses in improved network optimization algorithms, in *25th Annual Symposium on Foundations of Computer Science*, 24–26 October 1984, Singer Island, FL, US, 338–346. https://doi.org/10.1109/SFCS.1984.715934

Friedrich, M.; Hofsaess, I.; Wekeck, S. 2001. Timetable-based transit assignment using branch and bound techniques, *Transportation Research Record: Journal of the Transportation Research Board* 1752: 100–107. https://doi.org/10.3141/1752-14

Guo, J.-Y.; Jia, L.-M.; Qin, Y. 2015. Analyzing and computing dynamic accessibility with constraints of schedule, *Journal of Transportation Systems Engineering and Information Technology* 15(1): 118–122 (in Chinese).

Huang, R. 2007. A schedule-based pathfinding algorithm for transit networks using pattern first search, *GeoInformatica* 11(2): 269–285. https://doi.org/10.1007/s10707-006-0011-y

Huang, R.; Peng, Z.-R. 2002. Schedule-based path-finding algorithms for transit trip-planning systems, *Transportation Research Record: Journal of the Transportation Research Board* 1783: 142–148. https://doi.org/10.3141/1783-18

Kang, L.; Wu, J.; Sun, H.; Zhu, X.; Gao, Z. 2015. A case study on the coordination of last trains for the Beijing subway network, *Transportation Research Part B: Methodological* 72: 112–127. https://doi.org/10.1016/j.trb.2014.09.003

Luo, Q.; Xu, R.; Jiang, Z.; Chen, J. 2010. Dynamic accessibility of urban mass transit network based on train diagram, *Journal of Tongji University* (*Natural Science*) 38(1): 72–75. https://doi.org/10.3969/j.issn.0253-374x.2010.01.013 (in Chinese).

Martins, E. Q. V.; Pascoal, M. M. B. 2003. A new implementation of Yen's ranking loopless paths algorithm, *Quarterly Journal of the Belgian, French and Italian Operations Research Societies* 1(2): 121–133. https://doi.org/10.1007/s10288-002-0010-2

Martins, E. Q. V.; Pascoal, M. M. B.; Santos, J. L. E. 1999. Deviation algorithms for ranking shortest paths, *International Journal of Foundations of Computer Science* 10(3): 247–261. https://doi.org/10.1142/S0129054199000186

Sanders, P.; Schultes, D. 2006. Engineering highway Hierarchies, *Lecture Notes in Computer Science* 4168: 804–816. https://doi.org/10.1007/11841036_71

Tong, C. O.; Richardson, A. J. 1984. A computer model for finding the time-dependent minimum path in a transit system with fixed schedules, *Journal of Advanced Transportation* 18(2): 145–161. https://doi.org/10.1002/atr.5670180205

Van der Zijpp, N. J.; Catalano, S. F. 2005. Path enumeration by finding the constrained *K*-shortest paths, *Transportation Research Part B: Methodological* 39(6): 545–563. https://doi.org/10.1016/j.trb.2004.07.004

Wang, S.; Yang, Y.; Hu, X.; Li, J.; Xu, B. 2016. Solving the *K*-shortest paths problem in timetable-based public transportation systems, *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations* 20(5): 413–427. https://doi.org/10.1080/15472450.2015.1082911

Xu, W.; He, S.; Song, R.; Chaudhry, S. S. 2012. Finding the *K* shortest paths in a schedule-based transit network, *Computers & Operations Research* 39(8): 1812–1826. https://doi.org/10.1016/j.cor.2010.02.005

Yen, J. Y. 1971. Finding the *K* shortest loopless paths in a network, *Management Science* 17(11): 712–716. https://doi.org/10.1287/mnsc.17.11.712

Zhan, F. B.; Noon, C. E. 1998. Shortest path algorithms: an evaluation using real road networks, *Transportation Science* 32(1): 65–73. https://doi.org/10.1287/trsc.32.1.65

Ziliaskopoulos, A. K.; Mahmassani, H. S. 1993. Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications, *Transportation Research Record: Journal of the Transportation Research Board* 1408: 94–100.

Ziliaskopoulos, A.; Wardell, W. 2000. An intermodal optimum path algorithm for multimodal networks with dynamic arc travel times and switching delays, *European Journal of Operational Research* 125(3): 486–502. https://doi.org/10.1016/S0377-2217(99)00388-4

Zografos, K. G.; Androutsopoulos, K. N. 2008. Algorithms for itinerary planning in multimodal transportation networks, *IEEE Transactions on Intelligent Transportation Systems* 9(1): 175–184. http://doi.org/10.1109/TITS.2008.915650