International Journal of Management Science
Vol 10, No 1, May 2004

On Finding an Optimal Departure Time in Time-Dependent Networks*

Chan-Kyoo Park**
Dept. of IT Audit & Supervision, National Computerization Agency,
NCA Bldg., 77, Mugyo-dong, Jung-gu, Seoul, 100-170, Korea

Sangwook Lee***
Dept. of Industrial Engineering,
Seoul National University, Seoul, Korea

Soondal Park****
Dept. of Industrial Engineering,
Seoul National University, Seoul, Korea

(Received Dec. 2003; Revised Apr. 2004; Accepted Apr. 2004)

ABSTRACT

Most existing studies on time-dependent networks have been focused on finding a minimum delay path given a departure time at the origin. There, however, frequently happens a situation where users can select any departure time in a certain time interval and want to spend as little time as possible on traveling the networks. In that case, the delay spent on traveling networks depends on not only paths but also the actual departure time at the origin. In this paper, we propose a new problem in time-dependent networks whose objective is to find an optimal departure time given possible departure time interval at the origin. From the optimal departure time, we can obtain a path with minimum delay among all paths for possible departure times at the origin. In addition, we present an algorithm for finding an optimal departure time by enumerating trees which remain shortest path tree for a certain time interval.

Keywords: Minimum delay, Shortest path, Departure time

---

* This work was supported by Grant No. R01-2002-000-00168-0 from the Basic Research Program of the Korea Science and Engineering Foundation(KOSEF).

** Corresponding author, Email: parkck@nca.or.kr
*** Email: leesw@orlab.snu.ac.kr
**** Email: sdpark@orlab.snu.ac.kr

1. INTRODUCTION

A time-dependent network denotes the network where the delays of edges change with time. The fluctuation of arc delays is normally caused by time-varying congestion, which happens in most real road networks and communication networks. The shortest path problem in time dependent network has recently been one of the main issues in route planning problem since the development of intelligent transportation systems([14]). In this paper, the delay of arcs has the same meaning with the length of arcs, and thus shortest path and minimum delay path will be used interchangeably.

Cooke and Hasley [13] first dealt with a shortest path algorithm in time-dependent networks, and proposed a label correcting algorithm which was extended from Bellman's algorithm [2]. Orda and Rom [7] extended Dijkstra's algorithm when unrestricted waiting at any node is allowed, and suggested a label correcting algorithm when no waiting is allowed at any node. Ziliaskopoulos and Mahmassani [12] introduced DEQUE list for a label correcting algorithm for fining a minimum delay path. Kaufman and Smith [4] presented a condition under which FIFO rule holds in time-dependent networks, and showed that if a certain condition is satisfied at all nodes, label setting or label correcting algorithm can handle time-dependent shortest route problem without additional computation. Lee [5] presented a branch and bound algorithm to solve time-dependent shortest route problem where FIFO rules does not hold. Sung et al. [9] proposed a new time-dependent network model where the flow speed of each link depends on the time interval, and presented label setting algorithm modified from Dijkstra's algorithm.

Most existing researches on time-dependent networks have been concerned with finding a shortest path given a departure time at the origin or an arrival time at the destination. However, there frequently happens the situation where users can select any departure time in a certain time interval and want to spend as little time as possible on traveling the networks. Since there exists a shortest path for each departure time at the origin, users need to find a shortest path which has minimum length among all shortest paths for possible departure times at the origin. That is, both an optimal departure time and the corresponding shortest path should be determined. For example, a truck driver can start from the origin at 9 o'clock through 14 o'clock. He wants to start from the origin at the time which leads to minimum delay spent on traveling the network. Not only shortest paths but also an optimal departure time are important to the truck driver. Also, similar situations happen in communication network where users want to send a bulk of data in a certain time interval. In this case, user will try to know when they start to transmit the data in order that transmission time can be

minimized. To solve these problems, an intuitive approach is to solve shortest paths at every available departure time in a time interval and select the optimal departure time and the corresponding shortest path. For a large real network and reasonably precise departure time increment, this approach runs much too long to be useful in practice. Therefore, some dynamic approach for determining minimum delay path is needed.

In this paper, we propose a new problem which is to find an optimal departure time in a time-dependent network given departure time intervals. Throughout the paper, we assume that unrestricted waiting at every node including the origin is allowed. Without the assumption, finding a shortest path given a departure time at the origin is known to be NP-hard([7]). In addition, we assume that the delay function of each arc is piecewise linear within each time interval. This implies that the slope of the delay function is constant within an time interval. The assumption does not cause the great loss of generality because in practical environments the information on the delay of an arc is obtained and updated at every unit time. In fact, due to the modern technologies, intelligent transportation systems have one center that monitors and controls the overall road network traffic status, thereby being able to retrieve traffic data for a certain time interval.

A simple approach which one might devise to solve the proposed problem is as follows: First, find shortest paths for all departure time which can be enumerated by increasing some number of time units to the earliest departure time at the origin, and then select the shortest path which has minimum length among all shortest paths. One can expect presumably that the departure time of the selected shortest path is an optimal departure time. However, this simple approach might fail to find an optimal departure time. For an illustrative example, we consider a simple network composed of three nodes and two arcs as displayed in Figure 1, and the delays of arc (1, 2) and (2, 3) are given in Table 1.

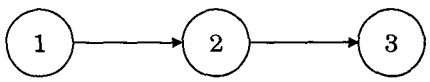

Figure 1. An example network

Table 1. Delay of arcs

arc	delay of arc for each time interval						
	[0, 1)	[1, 2)	[2, 3)	[3, 4)	[4, 5)	[5, 6)	[6, 7)
(1, 2)	3.5	4.5	5	5	5	4	4
(2, 3)	4	4	4	3	2.8	3	3

Suppose that departure at the origin can be made between time 0 and time 2. For departure time $t = 0, 1, 2$ at the origin, the length of shortest paths is shown in Table 2.

Table 2. The length of shortest paths for each departure time at the origin

arc (1, 2)			arc (2, 3)			Arrival time	Total delay
Available departure time	Actual departure time	Delay	Available departure time	Actual departure time	Delay		
0	0	3.5	3.5	3.5	3	6.5	6.5
1	1	4.5	5.5	5.5	3	8.5	7.5
2	2	5	7	7	3	10	8
0.6	**0.6**	**3.5**	**4.1**	**4.1**	**2.8**	**6.9**	**6.3**

In Table 2, the first column, 'available departure time' means the earliest time when departing from the origin is possible. But available departure time may differ from the actual departure time when departure from a node is actually made because waiting at a node can delay the departure. From Table 2, we know that minimum delay is achieved when the departure time at the origin is 0.6. This implies that we might fail to find an optimal departure time by solving shortest path problems defined for each departure time $t = 0, 1, 2\cdots$ at the origin, and consequently the simple approach might fail to solve the proposed problem. To find an optimal departure time, we need to solve shortest path problems for all possible departure times at the origin. In this paper, we propose an efficient algorithm for finding shortest paths for all possible departure time at the origin. The proposed algorithm is based on a parametric shortest path algorithm which can solve the shortest path problem in time-dependent networks where the delay of arcs is piecewise linear function of time.

The organization of the paper is as follows: In the next section, a parametric shortest path algorithm for time-dependent networks is developed. In Section 3, using the parametric shortest path algorithm, we propose an algorithm for finding an optimal departure time. An example problem is presented in Section 4, and finally some conclusions are presented in Section 5.

2. PARAMETRIC SHORTEST PATH PROBLEM

We consider the parametric shortest path problem in a directed network $G = (N, E)$, with $N = \{1, 2, \cdots, n\}$ being the set of nodes, $E \subseteq N \times N$ the set of arcs. Let $c(e)$ denote the length of arc e and $\delta(e)$ be the rate of change in the length of arc e. Throughout the paper, we assume that $c(e)$ has a positive value for all $e \in E$. For nonnegative real parameter λ, let $c_\lambda(e)$ denote the parameterized length of arc e where $c_\lambda(e) = c(e) + \lambda\delta(e)$. Let node 1 and node n be the origin and

the destination of the shortest path problem, respectively. Let $c_\lambda(p)$ denote the length of path where $c_\lambda(p) = \sum_{e\in p}(c(e)+\lambda\delta(e))$. In particular, let $c(p)$ denote the length of path when $\lambda = 0$, that is, $c(p) = \sum_{e\in p} c(e)$. Let $\delta(p)$ denote the rate of change in the length of path p where $\delta(p) = \sum_{e\in p}\delta(e)$. Let P_i denote the set of paths from node 1 to node i. For a given fixed λ, a path p^* from 1 to i is called a *shortest path* if $c_\lambda(p^*) = \min_{p\in P_i} c_\lambda(p)$. A shortest path p^* from 1 to i is called an *effective shortest path* from 1 to i if $\delta(p^*) \le \delta(p)$ where p is any shortest path from 1 to i. Given a set E of arcs, let $F(i)$ denote the set of arcs emanating from node i. A simple and nice property of effective shortest path is that every subpath of an effective shortest path is also an effective shortest path. From this property, we can develop an algorithm for finding an effective shortest path tree T such that every path from 1 to i in T is an effective shortest path when $\lambda = 0$. The algorithm can be obtained by a modified Dijkstra's algorithm.

Algorithm 1. Find_EST(N, E, c, δ)

```
0   begin
1     S = φ, S̄ = N
2     d(i) = ∞, γ(i) = ∞ for each node i ∈ N
3     d(1) = 0, γ(1) = 0 and pred(1) = 0
4     while |S| < n do
5     begin
6       let i ∈ S̄ be a node for which d(i) = min{d(j) : j ∈ S̄}
7       S = S ∪ {i}, S̄ = S̄ − {i}
8       for each arc (i,j) ∈ F(i) do
9         if d(j) > d(i) + c(i,j) then
10          d(j) = d(i) + c(i,j)
11          γ(j) = γ(i) + δ(i,j)
12          pred(j) = i
13        else if d(j) = d(i) + c(i,j) and γ(j) > γ(i) + δ(i,j) then
14          γ(j) = γ(i) + δ(i,j)
15          pred(j) = i
16        end if
17      end for
18    end
19  end
```

Lemma 2.1. *Suppose that* $c(i,j) > 0$ *for all* $(i,j) \in E$ *and* T *is a shortest path tree constructed by Algorithm 1. Then, every path in* T *is an effective shortest path for* $\lambda = 0$.

Proof. Algorithm 1 maintains distance label $d(i)$ and selects the next permanent nodes in the same manner as Dijkstra's algorithm, which implies that T obtained by Algorithm 1 is also a shortest path tree. Therefore, we only show that for every path p from 1 to i in T, the rate of change, $\delta(p)$, is not greater than those of the other shortest paths from 1 to i. Suppose there exists an effective shortest path q from 1 to i which has at least one non-tree arc. Let (u,v) be the first non-tree arc of q. Since the length of each arc is positive, node u must be selected as permanently labeled node before node v is selected. When u is selected, every arc emanating from u is considered by line 8~17. Thus, the permanent value of $\delta(v)$ is less than or equal to $\delta(q_v)$ where q_v denotes the subpath of q from 1 to v. Let p_v be the path from 1 to v in T. This implies that q', which is obtained by replacing the subpath q_v with p_v, is also an effective shortest path. By repeating the same argument, any path in T must be an effective shortest path. ■

Given an effective shortest path tree T for $\lambda = 0$, the range of λ, with which T remains an effective shortest path tree is calculated by the following:

Theorem 2.1. *Let* T *denote an effective shortest path tree when* $\lambda = 0$. *Let also* p_i *denote a unique path from 1 to i in* T. *Let* $\underline{\lambda}$ *and* $\overline{\lambda}$ *denote*

$$\underline{\lambda} = \max_{(u,v)\in C_1} \frac{c(p_u) + c(u,v) - c(p_v)}{\delta(p_v) - \delta(p_u) - \delta(u,v)}$$

$$\overline{\lambda} = \min_{(u,v)\in C_2} \frac{c(p_u) + c(u,v) - c(p_v)}{\delta(p_v) - \delta(p_u) - \delta(u,v)}$$

where

$$C_1 = \{(u,v) : (u,v) \in E - T \text{ and } \delta(p_v) < \delta(p_u) + \delta(u,v)\}$$

$$C_2 = \{(u,v) : (u,v) \in E - T \text{ and } \delta(p_v) > \delta(p_u) + \delta(u,v)\}$$

If $\underline{\lambda} < 0$, *the range* λ *within which* T *remains effective shortest path tree is* $\underline{\lambda} < \lambda < \overline{\lambda}$. *Otherwise, the range* λ *is* $0 \le \lambda < \overline{\lambda}$. *In addition,* $\overline{\lambda}$ *is positive.*

Proof. Let C be the set of non-tree arcs, that is, $C = E - T$. Let (u,v) be an arc in C, and let p_u and p_v be the shortest path in T from 1 to u and from 1 to v. The necessary and sufficient condition for T to be a shortest path tree is that $c(p_v) \le c(p_u) + c(u,v)$ for all $(u,v) \in C$. Also, since $\delta(e)$ is constant for all $e \in E$ and T is a shortest path tree for $\lambda = 0$, T remains a shortest path tree if and only if $c_\lambda(p_v) \le c_\lambda(p_u) + c_\lambda(u,v)$ for each $(u,v) \in C$ and $\lambda \in [\underline{\lambda}, \overline{\lambda}]$. Consequently, the following inequality must hold for all $(u,v) \in C$:

$$\begin{aligned} & c_\lambda(u,v) + c_\lambda(p_u) - c_\lambda(p_v) \ge 0 \\ \Rightarrow\ & c(u,v) + \lambda\delta(u,v) + c(p_u) + \lambda\delta(p_u) - (c(p_v) + \lambda\delta(p_v)) \ge 0 \\ \Rightarrow\ & c(u,v) + c(p_u) - c(p_v) - \lambda(\delta(p_v) - \delta(p_u) - \delta(u,v)) \ge 0 \end{aligned} \tag{2.1}$$

Since T is an effective shortest path tree for $\lambda = 0$, $c(u,v) + c(p_u) - c(p_v) \ge 0$ for each $(u,v) \in C$. The range of λ can be found as follows:

(a) In case that $\delta(p_u) - \delta(p_u) - \delta(u,v) = 0$,
Since $c(u,v) + c(p_u) - c(p_v) \ge 0$ by assumption, inequality (2.1) holds trivially for any value of λ.

(b) In case that $(u,v) \in C_1$, we obtain

$$\lambda \ge \max_{(u,v) \in C_1} \frac{c(p_u) + c(u,v) - c(p_v)}{\delta(p_v) - \delta(p_u) - \delta(u,v)} = \underline{\lambda} \tag{2.2}$$

(c) In case that $(u,v) \in C_2$, we have

$$\lambda \le \min_{(u,v) \in C_2} \frac{c(p_u) + c(u,v) - c(p_v)}{\delta(p_v) - \delta(p_u) - \delta(u,v)} = \overline{\lambda} \tag{2.3}$$

Therefore, the range of λ, within which T remains an shortest path tree is $\underline{\lambda} \le \lambda \le \overline{\lambda}$. In addition, if no new shortest path from the origin to any other node is formed when λ changes, T remains an effective shortest path tree. If a new shortest path from 1 to i is formed for a certain λ, then the rate of change in the length of a new shortest path is always less than that of the shortest path from 1 to i in T. This implies that the range of λ within which T remains an effective shortest path tree is $\underline{\lambda} < \lambda < \overline{\lambda}$ when $\underline{\lambda} < 0$. If $\lambda = 0$, the range of T is $0 \le \lambda < \overline{\lambda}$.

Next, we show that $\bar{\lambda}$ has a positive value. If $c(u,v)+c(p_u)-c(p_v)>0$ for all $(u,v)\in C$, the positivity of $\bar{\lambda}$ trivially holds. Suppose that $c(u,v)+c(p_u)-c(p_v)$ is zero. Then, by the definition of effective shortest path tree, it follows that $\delta(u,v)+\delta(p_u)\geq\delta(p_v)$. If $\delta(u,v)+\delta(p_u)=\delta(p_v)$, it corresponds to case (a). If $\delta(u,v)+\delta(p_u)>\delta(p_v)$, then $(u,v)\in C_1$. Therefore, for every $\text{arc}(u,v)\in C_2$, $c(u,v)+c(p_u)<c(p_v)$, which means that $\bar{\lambda}$ is positive. ■

Note that if $C_1=\phi$ then $\underline{\lambda}=-\infty$. The range of λ obtained by Theorem 2.1 might include both positive and negative values. In our proposed algorithm, however, we are only concerned about the range of time interval within which a given tree T remains an effective shortest path when time λ elapses. Therefore, only the nonnegative part of the range of λ given by Theorem 2.1 will be used in this paper.

When λ has a value equal to or greater than $\bar{\lambda}$, T does not remain an effective shortest path tree any more. To obtain another effective shortest path tree T' for $\lambda\geq\bar{\lambda}$, some arcs in T have to be replaced by some other arcs in $E-T$. The candidate arcs to enter T' are ones whose ratio in inequality (2.3) equals $\bar{\lambda}$. That is, T' can be constructed by considering only the arcs in T and C^* where

$$C^*=\{(u,v)\in C_2 \mid \frac{c(p_u)+c(u,v)-c(p_v)}{\delta(p_v)-\delta(p_u)-\delta(u,v)}=\bar{\lambda}\} \tag{2.4}$$

Let $E'=T\cup C^*$ and $c'(e)=c(e)+\bar{\lambda}\delta(e)$. Note that we use T to denote the set of arcs included in itself, which will not be confusing in the context although it is an abuse of notations. Then, T' is found by using Find_EST() in Algorithm 1 whose input parameter is (N,E',c',δ). In most practical situations, $|E'|$ is expected to be much less than $|E|$. This means that updating T into T' can be performed with far less computational effort than finding an effective shortest path tree T from scratch. In addition, any path constructed by the arcs in E' with its length being given by c' is a shortest path. In fact, the reduced cost of all the arcs E' is zero, and thus every path formed by E' is a shortest path. Using the results, we can avoid the calculation of distances d(i) in Algorithm Find_EST(), and only consider the rate of change, $\gamma(i)$.

3. FINDING A MINIMUM DELAY PATH

Consider a time-dependent directed network $G=(N, E)$ where the delay function of arc $(i,j)\in E$ is given by $d_{ij}(t)$. We assume that $d_{ij}(t)$ is a positive piecewise linear and right continuous function of time $t\geq 0$. If $[\underline{t},\bar{t}]$ is one of time intervals where $d_{ij}(t)$ is linear, $\underline{t}$ and $\bar{t}$ will be called the left and right breakpoint of the interval.

Consider another function $D_{ij}(t)$ which is derived from $d_{ij}(t)$ as follows:

$$D_{ij}(t)=\inf_{\tau\geq 0}(\tau+d_{ij}(t+\tau)) \tag{3.1}$$

That is, $D_{ij}(t)$ represents the minimum delay time to go through arc (i,j) at time t. Note that since unrestricted waiting at any node is allowed, $D_{ij}(t)$ might include waiting time at node i when the minimum delay in equation (3.1) is achieved for $\tau>0$. Also, since $d_{ij}(t)$ is assumed to be piecewise linear and right-continuous, $D_{ij}(t)$ is well defined for every $t\geq 0$ ([7]), and can be easily determined. Another feature we have to be cautious about is that $D_{ij}(t)$ has many τ^* or no τ^* such that $\tau^*+d_{ij}(t+\tau^*)=D_{ij}(t)$ for some t. For example, suppose $d_{ij}(t)=a+bt$ for $t\in[\underline{t},\bar{t})$. If the slope b of $d_{ij}(t)$ is equal to -1, then any $\tau\in[\underline{t},\bar{t})$ leads to the minimum delay of arc (i,j). Moreover, if the slope of $d_{ij}(t)$ is less than -1, then no $\tau\in[\underline{t},\bar{t})$ leads to the minimum delay of arc (i,j). As $t+\tau$ approaches to the right breakpoint of the interval, $(\tau+d_{ij}(t+\tau))$ converges to $(\bar{t}-\tau)+a+(t+(\bar{t}-\tau))b$. Since $d_{ij}(t)$ is not continuous at $\bar{t}$, the minimum delay can not be achieved at any time in $[\underline{t},\bar{t})$. If we assume that the slope of $d_{ij}(t)$ is greater than -1 for all $(i,j)\in E$ and $t\geq 0$, we can avoid this problem. Unfortunately, the assumption will make our algorithm applicable to more restrictive environment. Thus, we don't adopt the assumption, and circumvent the above problem by introducing a sufficiently small positive constant ε. If the minimum delay of equation (3.1) is achieved at the right breakpoint $\bar{t}$ of $D_{ij}(t)$ where $D_{ij}(t)$ is discontinuous, then $\bar{t}-\varepsilon$ will denote the time which is infinitely close to

$\bar{t}$. Also, $\bar{t}-\varepsilon$ will be used for denoting an optimal departure time which is infinitely close to the right breakpoint of a time interval.

Despite of the intricacy mentioned above, $D_{ij}(t)$ has a property similar to $d_{ij}(t)$ as follows:

Lemma 3.1. *For each* $(i,j)\in E$, $D_{ij}(t)$ *is a positive piecewise linear and right-continuous function.*

Proof. The positivity of $D_{ij}(t)$ directly follows from the definition. If we show that $D_{ij}(t)$ is piecewise linear for some interval $[t,t+\Delta]$, the right-continuity of $D_{ij}(t)$ is also directly obtained. Thus, it is sufficient to show that $D_{ij}(t)$ is a piecewise linear function. Let t_0 be an arbitrary time such that $D_{ij}(t_0)=\inf_{\tau\geq 0}\{\tau+d_{ij}(t_0+\tau)\}$. Since $D_{ij}(t_0)$ is the sum of waiting time at node i and traveling time through arc (i,j), let τ^* denote the waiting time of $D_{ij}(t_0)$. In case of many τ^* which leads to $D_{ij}(t_0)=\tau^*+d_{ij}(t_0+\tau^*)$, τ^* is uniquely chosen such that τ^* is the minimum among all τ^*.

(a) In case that $\tau^*>0$.

Let Δ be a positive number such that $\Delta\leq\tau^*$. It follows that

$$\begin{aligned}
D_{ij}(t_0+\Delta) &= \inf_{\tau\geq 0}(\tau+d_{ij}(t_0+\Delta+\tau)) \\
&= \min\{\inf_{\tau\leq\tau^*-\Delta}(\tau+d_{ij}(t_0+\Delta+\tau)),\ \inf_{\tau\geq\tau^*-\Delta}(\tau+d_{ij}(t_0+\Delta+\tau))\} \\
&= \min\{\inf_{\Delta\leq\tau\leq\tau^*}(\tau+d_{ij}(t_0+\tau))-\Delta,\ \inf_{\tau\geq\tau^*}(\tau+d_{ij}(t_0+\tau))-\Delta\} \\
&= \min\{D_{ij}(t_0)-\Delta, D_{ij}(t_0)-\Delta\} = D_{ij}(t_0)-\Delta.
\end{aligned}$$

Therefore, $D_{ij}(t)$ is linear in time interval $[t_0,t_0+\tau^*]$.

(b) In case that $\tau^*=0$.

Then, it follows that for any $\Delta\tau\geq 0$,

$$\begin{aligned}
&\Delta\tau+d_{ij}(t_0+\Delta\tau)\geq d_{ij}(t_0) \\
\Leftrightarrow\ &\frac{d_{ij}(t_0+\Delta\tau)-d_{ij}(t_0)}{\Delta\tau}\geq -1
\end{aligned} \tag{3.2}$$

This implies that any waiting at node i from time t_0 cannot reduce the delay of arc (i,j). Let $\overline{\Delta\tau} > 0$ be a constant such that $d_{ij}(t)$ is linear in time interval $[t_0, t_0 + \overline{\Delta\tau}]$. Therefore, we have that for any $\Delta t \in [0, \overline{\Delta\tau}]$

$$D_{ij}(t_0 + \Delta t) = \inf_{\tau \geq 0}(\tau + d_{ij}(t_0 + \Delta t + \tau)) = d_{ij}(t_0 + \Delta t) \tag{3.3}$$

In addition, we obtain from inequality (3.2) that

$$d_{ij}(t_0 + \Delta t) = d_{ij}(t_0) + \theta\Delta t, \text{ for any } \Delta t \in [0, \overline{\Delta\tau}]$$

where $\theta \geq -1$. Therefore, the equation (3.3) can be rewritten as

$$D_{ij}(t_0 + \Delta t) = d_{ij}(t_0) + \theta\Delta t \text{, for any } \Delta t \in [0, \overline{\Delta\tau}]$$

which implies that $D_{ij}(t)$ is linear in $[t_0, t_0 + \overline{\Delta t}]$. ■

By using $D_{ij}(t)$ instead of $d_{ij}(t)$, we can be free from any nuisance caused by considering unrestricted waiting time at nodes. In addition, the use of $D_{ij}(t)$ over $d_{ij}(t)$ makes FIFO rule hold automatically in time-dependent networks. Hence, we are only concerned with $D_{ij}(t)$ from now on.

Let p be a path from node i to j with $p = (i = i_0, \cdots, i_l = j)$. Note that l is greater than or equal to 1. Let p_{i_r} denote the subpath of p from i to i_r where $r = 1, \cdots, l$. For each subpath p_{i_r} of p, we define $A_t(p_{i_r})$, $D_t(p_{i_r})$ and $\delta_t(p_{i_r})$ recursively as follows:

$$A_t(p_{i_r}) = \begin{cases} t, & \text{if } r = 0 \\ A_t(p_{i_{r-1}}) + D_{i_{r-1},i_r}(A_t(p_{i_{r-1}})), & \text{if } r = 1, \cdots, l \end{cases} \tag{3.4}$$

$$D_t(p_{i_r}) = \begin{cases} 0, & \text{if } r = 0 \\ D_t(p_{i_{r-1}}) + D_{i_{r-1},i_r}(A_t(p_{i_{r-1}})), & \text{if } r = 1, \cdots, l \end{cases} \tag{3.5}$$

$$\delta_t(p_{i_r}) = \begin{cases} D'_{i_0,i_1}(t), & \text{if } r = 0 \\ \delta_t(p_{i_{r-1}}) + D'_{i_{r-1},i_r}(A_t(p_{i_{r-1}})) \cdot \delta_t(p_{i_{r-1}}), & \text{if } r = 2, \cdots, l \end{cases} \tag{3.6}$$

$A_t(p_{i_r})$ represents the arrival time to node i_r when the available departure

time at node i is t. $D_t(p_{i_r})$ represents the total sum of delay through all arcs in p_{i_r} when the available departure time at node i is t. Also, $\delta_t(p_{i_r})$ represents the rate of change in the delay of path p_{i_r}. We can know that by the definition $D_t(p_{i_r})$ is a minimum delay time of a path $p=(i_0,\cdots,i_r)$ given an available starting time t at node i, and $\delta_t(p_{i_r})$ is the derivative of $D_t(p_{i_r})$. By the definition, $A_t(p_{i_r})=t+D_t(p_{i_r})$ for $r=1,\cdots,l$. Note that $D'_{ij}(t)$ means the right-derivative of $D_{ij}(t)$. Since $D_{ij}(t)$ is piecewise linear right-continuous, $D_t(p_{i_r})$, $A_t(p_{i_r})$ and $\delta_t(p_{i_r})$ are well-defined.

In the parametric shortest path problem in Section 2 where arcs' length is constant, the length of a shortest path is expressed as the sum of parameterized length of arcs. We can derive a similar result for time-dependent networks as follows:

Theorem 3.1. *Let $p=(i=i_0,\cdots,i_l=j)$ be a path from node i to j. Let t_0 be the available starting time at node i. Suppose that for $\Delta t>0$, each delay function $D_{i_r i_{r+1}}(t)$ is linear in time interval $[A_{t_0}(p_{i_r}),A_{t_0+\Delta t}(p_{i_r})]$ where $r=0,\cdots,l-1$. Then,*

$$D_{t_0+\Delta t}(p)=D_{t_0}(p)+\Delta t\cdot\delta_{t_0}(p),\qquad A_{t_0+\Delta t}(p)=A_{t_0}(p)+\Delta t\cdot\delta_{t_0}(p).$$

Proof. We show that the following equations hold for each subpath of p:

$$D_{t_0+\Delta t}(p_{i_r})=D_{t_0}(p_{i_r})+\Delta t\cdot\delta_{t_0}(p_{i_r})\quad r=1,\cdots,l \tag{3.7}$$

$$A_{t_0+\Delta t}(p_{i_r})=A_{t_0}(p_{i_r})+\Delta t\cdot\delta_{t_0}(p_{i_r}),\quad r=1,\cdots,l \tag{3.8}$$

If $r=1$, equation (3.7) holds obviously. Suppose that equation (3.7) holds for all $r\le k-1$ where $k\ge 2$. This means that equation (3.8) also holds for all $r\le k-1$ since $A_t(p_r)=t+D_t(p_r)$. Then, we get

$$\begin{aligned}D_{t_0+\Delta t}(p_{i_k}) &= D_{t_0+\Delta t}(p_{i_{k-1}})+D_{i_{k-1},i_k}(A_{t_0+\Delta t}(p_{i_{k-1}}))\\ &= D_{t_0}(p_{i_{k-1}})+\Delta t\delta_{t_0}(p_{i_{k-1}})+D_{i_{k-1},i_k}(A_{t_0}(p_{i_{k-1}}))+\Delta t D'_{i_{k-1},i_k}(A_{t_0}(p_{i_{k-1}}))\delta_{t_0}(p_{i_{k-1}})\\ &= D_{t_0}(p_{i_k})+\Delta t\delta_{t_0}(p_{i_k}).\end{aligned}$$

Since $A_{t_0}(p_r)=t_0+D_{t_0}(p_r)$, equation (3.8) also holds for $r=k$. ■

Theorem 3.1 holds in time interval where each arc's minimum delay function is linear. The interval where Theorem 3.1 holds can be calculated by the following lemma:

Lemma 3.2. *Let* $p=(i=i_0,\cdots,i_l=j)$ *be a path from node i to j. Let* t_0 *be the available starting time at node i. Then,* $D_{t_0+\Delta t}(p)$ *is linear for* $0\le\Delta t<\bar{t}_{i_l}$ *where* $\bar{t}_{i_l}$ *is defined recursively as follows:*

$$\bar{t}_{i_1}=f_1-t_0$$
$$\bar{t}_{i_r}=\min\left\{\bar{t}_{i_{r-1}},\frac{f_r-A_{t_0}(p_{r-1})}{\delta_{t_0}(p_{i_{r-1}})}\right\},r=2,\cdots,l$$

where f_r *represents the smallest breakpoint of* $D_{i_{r-1},i_r}(t)$ *such that* $f_r\ge A_{t_0}(p_r)$.

Proof. We show the lemma by mathematical induction. If r = 1, we obtain the following equation:

$$D_{t_0+\Delta t}(p_1)=D_{i_0,i_1}(t_0+\Delta t) \tag{3.9}$$

Since $D_{i_0,i_1}(t_0+\Delta t)$ is linear in time interval $[t_0,f_1]$ where f_1 is the smallest breakpoint of $D_{i_0,i_1}(t)$ with $t\ge t_0$. Hence, $D_{t_0+\Delta t}(p_1)$ is linear in time interval $[t_0,t_0+\Delta t]$ where $\Delta t=f_1-t_0=\bar{t}_{i_1}$. If r = 2, we obtain that

$$D_{t_0+\Delta t}(p_2)=D_{t_0+\Delta t}(p_1)+D_{i_1,i_2}(A_{t_0+\Delta t}(p_1)) \tag{3.10}$$

The first term of equation (3.10) is linear for $0\le\Delta t\le\bar{t}_{i_1}$. For its second term to be linear, the following inequality should hold:

$$A_{t_0+\Delta t}(p_1)\le f_2 \tag{3.11}$$

where f_2 is the smallest breakpoint of $D_{i_1,i_2}(t)$ such that $t\ge A_{t_0+\Delta t}(p_1)$. Since $A_{t_0+\Delta t}(p_1)=A_{t_0}(p_1)+\Delta t\delta_{t_0}(p_1)$ for $0\le\Delta t\le\bar{t}_{i_1}$, equality (3.11) can be rewritten into

$$\Delta t\le\frac{f_2-A_{t_0}(p_1)}{\delta_{t_0}(p_1)}. \tag{3.12}$$

Therefore, $D_{t_0+\Delta t}(p_2)$ is linear for $0 \le \Delta t \le \bar{t}_{i_2}$. Suppose that the lemma holds for all $r \le k$. By the definition of $D_t(p_r)$,

$$D_{t_0+\Delta t}(p_{k+1}) = D_{t_0+\Delta t}(p_k) + D_{i_k,i_{k+1}}(A_{t_0+\Delta t}(p_k)) \tag{3.13}$$

By the similar arguments, the following equality should hold for the second term of equation (3.13) to be linear: for $\Delta t \le \bar{t}_{i_k}$,

$$\begin{aligned} & A_{t_0+\Delta t}(p_k) \le f_{k+1} \\ \Leftrightarrow\ & A_{t_0}(p_k) + \Delta t \delta_{t_0}(p_k)) \le f_{k+1} \\ \Leftrightarrow\ & \Delta t \le \frac{f_{k+1} - A_{t_0}(p_k)}{\delta_{t_0}(p_k)}, \end{aligned}$$

where f_{k+1} is the smallest breakpoint of $D_{i_k,i_{k+1}}(t)$ such that $t \ge A_{t_0+\Delta t}(p_k)$. Therefore, $D_{t_0+\Delta t}(p_{k+1})$ is linear for $0 \le \Delta t \le \bar{t}_{i_{k+1}}$. ■

We now discuss how to apply the parametric shortest path algorithm to find shortest paths for all departure time at the origin. First, the method to find an effective shortest path tree in time-dependent networks is needed. For this, we use Algorithm 1 in Section 2 with some modifications. Suppose that the available departure time at the origin is t_0. To find an effective shortest path tree in time dependent networks, the calculations of $d(i)+c(i,j)$ and $\gamma(i)+\delta(i,j)$ need to be modified as follows:

$$d(i)+c(i,j) \ \rightarrow \ d(i) + D_{i,j}(t_0 + d(i)) \tag{3.14}$$

$$\gamma(i)+\delta(i,j) \ \rightarrow \ \begin{cases} D'_{i,j}(t_0 + d(i)) & , \ \textit{if } i = 1 \\ \gamma(i) + D'_{i,j}(t_0 + d(i))\gamma(i) & , \ \textit{otherwise} \end{cases} \tag{3.15}$$

The modified *Find_EST()* will be referred to as *Find_EST_TD()*. Label $d(i)$ stores the minimum delay time of a path from the origin to node i, and $\gamma(i)$ contains the rate of change in the minimum delay time of the corresponding path. The arrival time to node i is easily calculated by adding t_0 to $d(i)$. Note that $D_{i,j}(d(i))$ and $D'_{i,j}(d(i))$ depend on $d(i)$, and thus their values are not determined until $d(i)$ is known.

Next, we propose the way how to calculate the range of t within which a given

effective shortest path tree remains as it is. The range of t can be calculated by modifying Theorem 2.1.

Theorem 3.2. *Let T_{t_0} be an effective shortest path tree given available departure time t_0 at the origin. Suppose that the minimum delay function of every path from the origin to the other nodes is linear for $t_0 \in [t_0, t_0 + \bar{t})$. For any $(u,v) \in E$, p_u and p_v denote the shortest paths in T_{t_0}. Then, T_{t_0} remains an effective shortest path tree for time interval $[t_0, t_0 + \overline{\Delta t})$ where $\overline{\Delta t}$ is defined as follows:*

$$\overline{\Delta t} = \min\left\{ \bar{t}, \min_{(u,v)\in C} \frac{D_{t_0}(p_u) + D_{u,v}(A_{t_0}(p_u)) - D_{t_0}(p_v)}{\delta_{t_0}(p_v) - \delta_{t_0}(p_u) - D'_{u,v}(A_{t_0}(p_u))\delta_{t_0}(p_u)} \right\}, \tag{3.16}$$

where $C = \{(u,v) \mid (u,v) \in E - T_{t_0} \text{ and } \delta_{t_0}(p_v) > \delta_{t_0}(p_u) + D'_{u,v}(A_{t_0}(p_u))\delta_{t_0}(p_u)\}$. *In addition, $\overline{\Delta t}$ is strictly positive.*

Proof. By the definition of A_t, D_t and δ_t in equations (3.4)~(3.6) and Theorem 3.1, $D_{t_0}(p_u)$, $D_{t_0}(p_v)$ and $D_{u,v}(A_{t_0}(p_u))$ correspond to $c(p_u)$,$c(p_v)$ and $c(u,v)$, respectively. Similarly, $\delta_{t_0}(p_u)$, $\delta_{t_0}(p_v)$ and $D'_{u,v}(A_{t_0}(p_u))\delta_{t_0}(p_u)$ correspond to $\delta(p_u)$, $\delta(p_v)$ and $\delta(u,v)$, respectively. Therefore, the theorem follows directly from Theorem 2.1. ■

In Theorem 3.2, we assume that the minimum delay function of every path from the origin to the other nodes is linear when available departure time t_0 at the origin belongs to time interval $[t_0, t_0 + \bar{t})$. To calculate $\bar{t}$ using Lemma 3.2, Lemma 3.2 must be applied to every path in T_{t_0} and also every path constructed by concatenating a path in T_{t_0} and one non-tree arc. A path in T_{t_0} will be called a tree path and a path which is formed by concatenating a tree path and one non-tree arc will be called a extended path.

Finally, updating T into another effective shortest path tree T' is similar to updating T in Section 2. The candidate arcs to enter T' are ones whose ratio in equation (3.16) equals $\bar{\lambda}$. If Δt in equation (3.16) is equal to $\bar{t}$, it means that a new effective shortest path tree need to be found so that the length of all paths can be expressed as linear functions for another time interval. On the other hand,

if $\overline{\Delta t}$ in equation (3.16) is less than $\bar{t}$, the candidate arcs to enter T' belongs to the following set of arcs:

$$C^* = \left\{ (u,v) \in E - T \mid \frac{D_{t_0}(p_u) + D_{u,v}(A_{t_0}(p_u)) - D_{t_0}(p_v)}{\delta_{t_0}(p_v) - \delta_{t_0}(p_u) - D'_{u,v}(A_{t_0}(p_u))\delta_{t_0}(p_u)} = \overline{\Delta t} \right\}$$

That is, T' can be constructed by considering only the arcs in T and C^*. T' is also found by using *Find_EST_TD()*.

We present an algorithm for finding an optimal departure time and a minimum delay path given available departure time interval $[\underline{\tau}, \bar{\tau}]$.

Algorithm 2. Finding an optimal departure time

```
begin
1   d* ← ∞;
2   Let ε be a sufficiently small positive number.
3   Find an effective shortest path tree T with time t0 = τ_.
4   t0 ← τ_;
5   while (t0 ≤ τ̄)
6   begin
7       Find a time interval [t0, t̄) where the lengths of all paths in P are linear.
8       while (t0 < t̄)
9       begin
10          compute Δt̄ using equation (3.16).
11          Δt ← min(t̄, Δt̄);
12          d = min_{t0 ≤ τ ≤ t0+Δt} (D_{t0}(pn) + τδ_{t0}(pn))
13          if d < d*, then
14              d* ← d
15              if Δt = t̄ and d* = (D_{t0}(pn) + (t0 + Δt)δ_{t0}(pn))
16                  t* ← (t0 + Δt − ε);
17              else
18                  t* ← τ*;
19              end if
```

```
20            p* ← p_n ;
21          end if
22          t_0 ← t_0 + Δt ;
23          Find T' using Find_EST_TD() with time t_0 .
24          T ← T';
25       end
26    end
end
```

In Algorithm 2, P means the set of all tree paths and extended paths. In line 18, τ^* is from the equation $d^* = D_{t_0}(p_n) + \tau^* \delta_{t_0}(p_n)$. After Algorithm 2 terminates, t^* has an optimal departure time, p^* stores the minimum delay path, and d^* has the delay of p^*. The computational amount of Algorithm 2 is proportional with the number of effective shortest path tree produced by Algorithm 2. If the number of effective shortest path tree found by Algorithm 2 is F, then the complexity of Algorithm 2 is $O(F \times S(|N|,|E|))$ where $S(|N|,|E|)$ represents the complexity of a shortest path algorithm for $G = (N, E)$.

4. EXAMPLES

For the illustrative purpose, let us consider the following network. We assume that the origin is 1 and the destination is 6.

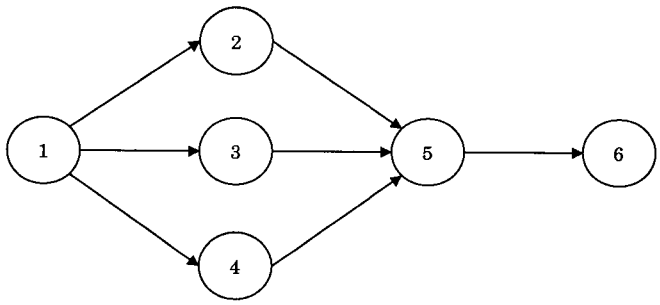

Figure 2. An example network

As given by Table 3, the travel time of each arc is piecewise linear. The fig-

ures in parentheses of Table 3 represent the rate of change in each arc's delay. The available starting time interval is [0, 4).

Table 3. Arc travel time

Arc	Delay of arc for each time interval				
	[0, 1)	[1, 2)	[2, 3)	[3, 4)	[4, 5)
(1, 2)	0.2(0.5)	0.7(0.2)	0.9(-0.7)	0.2(-0.2)	0.4(0.1)
(1, 3)	0.4(0.1)	0.5(-0.4)	0.1(0.2)	0.3(-0.1)	0.2(0.2)
(1, 4)	0.5(0.1)	0.6(0.1)	0.7(0.9)	1.6(-0.5)	1.1(0.1)
(2, 5)	0.4(0.3)	0.7(0.5)	1.2(-0.8)	0.4(0.1)	0.5(0.3)
(3, 5)	0.3(0.1)	0.4(-0.2)	0.2(0.4)	0.6(-0.2)	0.4(0.2)
(4, 5)	0.4(0.1)	0.5(0.1)	0.6(0.1)	0.7(0.4)	1.1(0.2)
(5, 6)	0.2(0.2)	0.4(-0.3)	0.1(0.1)	0.2(0.4)	0.6(0.4)

The arrival time, rate of change in delay, the total delay, and linear interval of the paths in the effective shortest path tree and the extended paths are presented in Table 4.

Table 4. Iteration 1

Path	Departure time	Arrival time	$\delta_{t_0}(p_i)$	$D_{t_0}(p_i)$	Linear interval
1-2	0	0.2	0.5	0.2	[0,1)
1-2-5	0	0.6	0.65	0.6	[0,1)
1-2-5-6	0	0.8	0.78	0.8	[0,1)
1-3	0	0.4	0.1	0.4	[0,1)
1-3-5	0	0.7	0.11	0.7	[0,1)
1-3-5-6	0	0.9	0.132	0.9	[0,1)
1-4	0	0.5	0.1	0.5	[0,1)
1-4-5	0	0.9	0.11	0.9	[0,1)
1-4-5-6	0	1.1	0.132	1.1	[0,1)

The effective shortest path at time $t_0 = 0$ is $1-2-5-6$ and its delay is 0.8. Algorithm 2 sets $t^* = 0$.

For the sake of simplicity, let π_{uv} denote

$$\pi_{uv} = \frac{D_{t_0}(p_u) + D_{u,v}(A_{t_0}(p_u)) - D_{t_0}(p_v)}{\delta_{t_0}(p_v) - \delta_{t_0}(p_u) - D'_{u,v}(A_{t_0}(p_u))\delta_{t_0}(p_u)}$$

where (u,v) is a non-tree arc. To find the time interval where T_1 remains effective shortest path tree, $\overline{\Delta t}$ is calculated by using equation (3.16) as follows.

$$\overline{\Delta t} = \min\{4, \pi_{35}(= 0.185185185), \pi_{45}(= 0.555555)\} = 0.185185185$$

Note that the non-tree arcs of T_1 are (3,5) and (4,5). Hence, the next departure time to be considered is $t_0 = 0.185185185$, and arc (3,5) comes into the effective shortest path tree.

The 'departure time', 'arrival time', $\delta_{t_0}(\cdot)$, $D_{t_0}(\cdot)$, and 'linear time interval' at $t_0 = 0.185185185$ are given by Table 5.

Table 5. Iteration 2

Path	Departure time	Arrival time	$\delta_{t_0}(p_i)$	$D_{t_0}(p_i)$	Linear interval
1-2	0.185185185	0.477777778	0.5	0.292592593	[0.185185185,1)
1-2-5	0.185185185	1.416666666	0.65	1.231481481	[0.185185185,1)
1-2-5-6	0.185185185	1.691666666	0.555	1.506481481	[0.185185185,1)
1-3	0.185185185	0.603703704	0.1	0.418518519	[0.185185185,1)
1-3-5	0.185185185	0.882962963	0.11	0.697777778	[0.185185185,1)
1-3-5-6	0.185185185	1.018074074	0.143	0.832888889	[0.185185185,1)
1-4	0.185185185	0.703703704	0.1	0.518518519	[0.185185185,1)
1-4-5	0.185185185	1.274074074	0.11	1.088888889	[0.185185185,1)
1-4-5-6	0.185185185	1.591851852	0.077	1.406666667	[0.185185185,1)

The effective shortest path at time $t_0 = 0.185185185$ is $1-3-5-6$, and its delay is 0.832889. The non-tree arcs are (2,5) and (4,5), and thus $\overline{\Delta t}$ is calculated as follows:

$$\overline{\Delta t} = \min\{4, \pi_{45}(= 1.28102 \times 10^{17})\} = 4$$

In the above equation, we do not consider π_{25} because $\pi_{25} = -0.255829904$ means the waiting time earlier than 0.185185185. Although the minimum value in the above equation is 4, path $1-3-5-6$ continues to be effective shortest path until time $t_0 = 1$ since the linear interval is [0,1). Therefore, the next departure time to be considered is $t_0 = 1$ and the 'departure time', 'arrival time', $\delta_{t_0}(\cdot)$, $D_{t_0}(\cdot)$, and 'linear time interval' at $t_0 = 1$ are given by Table 6.

Table 6. Iteration 3

Path	Departure time	Arrival time	$\delta_{t_0}(p_i)$	$D_{t_0}(p_i)$	Linear interval
1-2	1	1.7	0.7	0.7	[1,2)
1-2-5	1	2.75	1.05	1.75	[1,2)
1-2-5-6	1	2.925	1.155	1.925	[1,2)
1-3	1	1.5	0.5	0.5	[1,2)
1-3-5	1	1.8	0.4	0.8	[1,2)
1-3-5-6	1	1.96	0.28	0.96	[1,2)
1-4	1	1.6	0.6	0.6	[1,2)
1-4-5	1	2.16	0.66	1.16	[1,2)
1-4-5-6	1	2.276	0.726	1.276	[1,2)

The effective shortest path at time $t_0 = 1$ is $1-3-5-6$, and its delay is 0.96. The non-tree arcs are (2,5) and (4,5), and thus $\overline{\Delta t}$ is calculated as follows:

$$\overline{\Delta t} = \min\{4\} = 4$$

Although the minimum value in the above equation is 4, the tree continues to be effective shortest path tree until $t_0 = 2$ by linearity of the path. Therefore, the next departure time to be considered is $t_0 = 2$ and the 'departure time', 'arrival time', $\delta_{t_0}(\cdot)$, $D_{t_0}(\cdot)$, and 'linear time interval' at $t_0 = 2$ are shown in Table 7. The effective shortest path at time $t_0 = 2$ is $1-3-5-6$ and its delay is 0.474. Consequently, t^* is set to 2.

Table 7. Iteration 4

Path	Departure time	Arrival time	$\delta_{t_0}(p_i)$	$D_{t_0}(p_i)$	Linear interval
1-2	2	2.9	-0.7	0.9	[2,3)
1-2-5	2	4.82	-0.14	2.82	[2,3)
1-2-5-6	2	5.748	-0.196	3.748	[2,3)
1-3	2	2.1	0.2	0.1	[2,3)
1-3-5	2	2.34	0.28	0.34	[2,3)
1-3-5-6	2	2.474	0.308	0.474	[2,3)
1-4	2	2.7	0.9	0.7	[2,3)
1-4-5	2	3.37	0.99	1.37	[2,3)
1-4-5-6	2	3.718	1.386	1.718	[2,3)

The non-tree arcs are (2,5) and (4,5), and thus $\overline{\Delta t}$ is calculated as follows:

$$\overline{\Delta t} = \min\{4, 3.285714286\} = 3.285714286$$

Although the minimum value in the above equation is 3.285714286, the current tree continues to be effective shortest tree until $t_0 = 3$ by linearity of the path. Therefore, the next departure time to be considered is $t_0 = 3$, and the 'departure time', 'arrival time', $\delta_{t_0}(\cdot)$, $D_{t_0}(\cdot)$, and 'linear time interval' at $t_0 = 3$ are as Table 8.

Table 8. Iteration 5

Path	Departure time	Arrival time	$\delta_{t_0}(p_i)$	$D_{t_0}(p_i)$	Linear interval
1-2	3	3.2	-0.7	0.2	[3,4)
1-2-5	3	3.62	-0.77	0.62	[3,4)
1-2-5-6	3	4.068	-1.078	1.068	[3,4)
1-3	3	3.3	0.2	0.3	[3,4)
1-3-5	3	3.84	0.16	0.84	[3,4)
1-3-5-6	3	4.376	0.224	1.376	[3,4)
1-4	3	4.6	0.9	1.6	[3,4)
1-4-5	3	5.82	1.08	2.82	[3,4)
1-4-5-6	3	7.148	1.512	4.148	[3,4)

The effective shortest path at time $t_0 = 3$ is $1-2-5-6$ and its delay time is 1.068. The non-tree arcs are (3,5) and (4,5), and thus $\overline{\Delta t}$ is calculated as follows:

$$\overline{\Delta t} = \min\{4\} = 4$$

Therefore, Algorithm 2 will stop. The optimal departure time is $t^* = 2$, and the corresponding minimum delay path is $1-3-5-6$ with minimum delay of 0.474. The effective shortest path tree for each iteration are shown as following Figure 3. The effective shortest path trees for 'iteration 1', 'iteration 2~4' and 'iteration 5' are (*a*), (*b*) and (*c*), respectively.

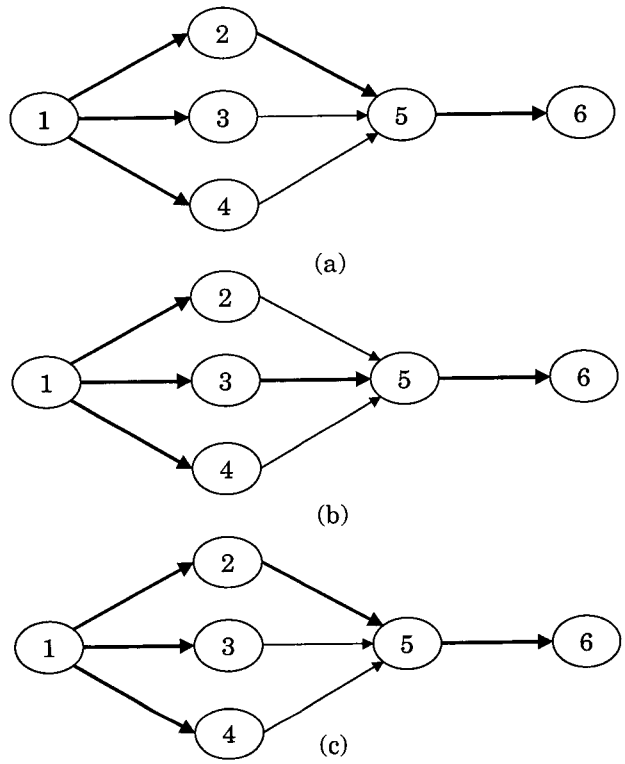

Figure 3. Effective shortest path trees

5. CONCLUSIONS

In this paper, we proposed a new problem in time dependent networks, which is to find an optimal departure time given available departure time interval at the origin. An optimal departure time leads to the minimum delay spent on traveling the networks. We also presented an algorithm for finding an optimal departure time when the delay function of each arc is piecewise linear. The proposed algorithm updates a sequence of trees which remains an effective shortest path tree for some time interval. The algorithm can find a minimum delay path for each available departure time, and determine the minimum delay path whose departure time is an optimal departure time. As a subject of further research, it will be interesting to apply the proposed algorithm to real time-dependent networks and to examine its performance or usefulness.

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, *Network Flows: Theory, Algorithms and Applications*, Prentice-Hall, 1993.

[2] Bellman, R., "On a routing problem," *Quart. Appl. Mathematics* 16 (1958), 87-90.

[3] M. I. Henig, "The shortest path problem with two objective functions," *European Journal of Operational Research* 25 (1985), 281-291.

[4] Kaufman, D. E., R. L. Smith, "Minimum travel time paths in dynamic networks with application to intelligent vehicle highway systems," *IVHS Journal*, To appear.

[5] Myung-Seok Lee, Soondal Park, "A study on shortest problem between specified nodes with multiple travel time," *Korean Management Science Review* 7, 2 (1990), 51-57, 17.

[6] E. D. Miller, H. S. Mahmassani, A. Ziliaskopoulos, "Path search techniques for transportation networks with time-dependent, stochastic arc costs," *Proc. of 1994 IEEE International Conference on Systems*, Men and Cybernetics, Part 2, 1716-1721.

[7] A. Orda and R. Rom, "Shortest-path and minimum-delay algorithms in Networks with Time-Dependent Edge-Length," *Journal of the ACM* 37, 3 (1997), 607-625.

[8] D. R. Shier and Christoph Witzgall, "Arc tolerances in shortest path and network flow problems," *Networks* 10 (1980), 277-291.

[9] Sung, K., M.G.H. Bell, M. Seong, S. Park, "Shortest paths in a network with time-dependent flow speeds," *European Journal of Operational Research* 121 (2000), 32-39.

[10] N. Ravi and R. E. Wendell, "The tolerance approach to sensitivity analysis in network linear programming," *Networks* 18 (1988), 159-171.

[11] N. E. Young, R. E. Tarjan and J. B. Orlin, "Faster parametric shortest path and minimum-balance algorithms," *Networks* 21 (1991), 205-221.

[12] A. K. Ziliaskopoulos, H. S. Mahmassani, "Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications," *Tranportations Science Record* 1408 (1993), 94-100.

[13] Cooke, K. L., E. Halsey, "The shortest route through a network with time-dependent internodal transit times", *Journal of Math.* Anal. Appl., 14 (1966), 492-498.

[14] Garcia, A., Amin, S. M.,Wooton, J. R., "Intelligent Transportation Systems - Enabling Technologies", *Mathl. Comput. Modelling.*, 22, 4-7 (1995), 11-81.