International Journal of Management Science
Vol 4, No 2, November 1998

MODELS AND SOLUTION METHODS FOR SHORTEST PATHS IN A NETWORK WITH TIME-DEPENDENT FLOW SPEEDS*

Ki Seok Sung
Industrial Engineering Department, Kangnung National University
Kangnung, 210-702, Korea
Sung@knusun.kangnung.ac.kr

Michael G. H. Bell
Transport Operations Research Group, University of Newcastle Upon Tyne
Newcastle Upon Tyne, NE1 7RU, United Kingdom
M.G.H.Bell@newcastle.ac.uk

(Received March 1998; revision received July 1998)

ABSTRACT

The Shortest Path Problem in Time-dependent Networks, where the travel time of each link depends on the time interval, is not realistic since the model and its solution violate the Non-passing Property (NPP:often referred to as FIFO) of real phenomena. Furthermore, solving the problem needs much more computational and memory complexity than the general shortest path problem. A new model for Time-dependent Networks where the flow speeds of each link depend on time interval, is suggested. The model is more realistic since its solution maintains the NPP. Solving the problem needs just a little more computational complexity, and the same memory complexity, as the general shortest path problem. A solution algorithm modified from Dijkstra's label setting algorithm is presented. We extend this model to the problem of Minimum Expected Time Path in Time-dependent Stochastic Networks where flow speeds of each link change statistically on each time interval. A solution method using the Kth-shortest Path algorithm is presented.

1. INTRODUCTION

Most of dynamic traffic assignment(DTA) algorithms require massive amount of iterative computation of shortest paths on time-dependent networks(SPTDN).

* This research was supported by Korea Research Foundation, 1996.

For an instance, the DTA algorithm built by Mahmassani *et al.*[8] spends more than seventy percent of CPU time on path computation. That means the computation of SPTDN is the bottleneck to the real-time implementation of his algorithm. Besides DTA, many other areas in telecommunications and logistics require shortest path computation that accounts for time-dependency of networks.

Many models and algorithms to find SPTDN have been developed. All those are based on the model of time-dependent network where link length or link travel time is depending on time interval. In such models, a shortest path does not guarantee non-passing property(NPP) which is often referred to as FIFO. In real transportation network, NPP implies that two vehicles travelling on same link arrive at the end of the link in the same order as that in which they start, even when some congestion occurs during the travel. For more realistic and useful models on transportation systems, establishing models and solution methods for SPTDN satisfying NPP has been an unsolved problem for a long time.

Cooke and Halsey[3] first proposed a Shortest Path Problem in networks where inter-nodal time requirements are included. The algorithm they suggest is a modified form of Bellman's[2] label correcting shortest path algorithm. The computational complexity of their algorithm is $O(n^3M)$, where n is the number of nodes in the network and M is the longest travel time of any link at any time interval. Their algorithm is pseudo-polynomial because M is included in the complexity. The optimal solution itself can not guarantee the NPP.

Orda and Rom[10] suggested a formulation of the Shortest Path Problem in networks in which link delays are functions of time. They stated two cases, one in which waiting at the nodes is prohibited and the other in which it is allowed. In the latter case, the problem can be solved efficiently by modifying Dijkstra's algorithm. Their model is appropriate for communication networks, but it is not realistic in transportation networks because it may violate the NPP, and unrestricted waiting at road junctions is prohibited.

Ziliaskopoulos and Mahmassani[12] tackled again the problem of Shortest Paths in time-dependent networks, and suggested an algorithm that is based on the general Bellman's Principle of Optimality. They proposed a label correcting shortest path algorithm. The algorithm uses dequeue list technique to reduce the number of label corrections. Their algorithm is very efficient in practice but unattractive in worst-cases, and the optimal solution also does not guarantee the NPP.

Kaufmann and Smith[7] suggested a condition that a link travel time satisfies the NPP in a time-dependent network. They showed that if all the link travel times satisfy the NPP, then the Principle of Optimality is also satisfied

and any shortest path algorithm based on label-setting or label-correcting can be applied without additional computational complexity. They tried to modify the link travel time data to enforce the NPP with his suggested condition. When the modification is successfully applied, the resulting solution possesses the NPP, but the data modification process itself may cause more computational burden than the shortest path algorithm.

Here we suggest a model for SPTDN where flow speeds of each link depend on time interval. In the model, the flow speed of each link in a network changes as time interval changes. Then the travel time of each link is calculated according to the flow speed at the time of passing the link. The model is more realistic and the optimal solution maintains the NPP. Furthermore, calculating optimum SPTDN based on the flow speed model is much easier than that of the link travel time model. We present a solution algorithm modified from Dijkstra's label setting algorithm.

Hall[5] showed that standard shortest path algorithms do not find the minimum expected travel time path on a network with travel times that are random and time-dependent. We extend our flow speed model to consider the minimum expected time path in time-dependent stochastic networks, where the flow speeds of each link change statistically in each time interval. And we also comment on the possibility of a solution method using the *K*th-shortest path algorithm.

2. THE MODEL

We describe a model for SPTDN where the flow speed of each link depends on time interval. Consider a network $G=(N, A)$ with node set $N=\{1, 2, ..., n\}$ and link set $A\subseteq\{(i, j)\in N\times N\}$. Let $l_{(i, j)}$ be the non-negative length of link (i, j). Let $[f_k, f_{k+1})$, $k=0, 1, 2,..,V\text{-}1$ be the time intervals divided by the time fences $f_0 < f_1 < f_2 \ldots\ldots < f_{V\text{-}1} < f_V$ defined on the time horizon. Let $v_{k(i,j)}$ be the non-negative flow speed in the time interval $[f_k, f_{k+1})$ on link (i, j). For each node $j\in N$, $j\neq 1$, define a value $d(j)=\mathrm{Min}_{j\neq i}\{T(d(i), (i, j))\}$ where $T(d(i), (i, j))$ is the arrival time at node j starting from node i at time $d(i)$. The arrival time $T(d(i), (i, j))$ will be explained in the next section. Then, the objective of the model is to find a value $d(j)$ and the link set $P(j)\subseteq A$ that gives the value $d(j)$ for all node $j\in N$, $j\neq 1$, when the starting time at origin node is given as $d(1)=s$.

3. NON-PASSING TRAVEL TIME

Consider a link (i, j) of length l (in this section, we will drop the script (i, j)) with the vehicle speed $v_k \geq 0$ changing according to the time intervals $[f_k, f_{k+1})$, $k=0, 1, 2,..,V\text{-}1$. A vehicle starting at node i at time s will arrive at node j at time $T(s)$ where $T(s)\text{-}s$ is the travel time between node i and j on link (i, j). Assuming that $f_0 \leq s \leq f_1$, the arrival time $T(s)$ is calculated as follows.

$$
\begin{aligned}
& T(s) = l/v_0 + s, && \text{if } l/v_0 < f_1 - s \\
\text{else}\quad & T(s) = (l - l_0)/v_1 + f_1 && \text{if } (l - l_0)/v_1 < f_2 - f_1 \\
\text{else}\quad & T(s) = (l - l_1)/v_2 + f_2 && \text{if } (l - l_1)/v_2 < f_3 - f_2 \\
& \quad \cdots \\
\text{else}\quad & T(s) = (l - l_{k-1})/ v_k + f_k && \text{if } (l - l_{k-1})/ v_k < f_{k+1} - f_k \\
& \quad \cdots
\end{aligned}
$$

$$
\begin{aligned}
\text{where}\quad & l_0 = v_0(f_1 - s), \\
& l_1 = l_0 + v_1(f_2 - f_1), \\
& l_2 = l_1 + v_2(f_3 - f_2), \\
& \quad \cdots \\
& l_k = l_{k-1} + v_k(f_{k+1} - f_k), \\
& \quad \cdots
\end{aligned}
$$

Two vehicles travelling on a link arrive at the end of the link in the same order as they start, even when some congestion occurs during the travel. Here we will show that the travel time calculated above is consistent to NPP. Assume that two vehicles starting at node i at time s and t are arriving at node j at time $T(s)$ and $T(t)$ respectively. Then we can show the NPP as follows:

Theorem (Non-passing Property): $T(s) \leq T(t)$, if $s \leq t$.

(Proof)

Assume $s \in [f_i, f_{i+1})$, $t \in [f_j, f_{j+1})$, $T(s) \in [f_k, f_{k+1})$, $T(t) \in [f_m, f_{m+1})$,
where $i \leq j \leq k$, m, and let the length of the link be l.

Then,

$$
\begin{aligned}
l &= v_m(T(t) - f_m) + v_{m-1}(f_m - f_{m-1}) + \cdots + v_j(f_{j+1} - t) \\
&= v_k(T(s) - f_k) + v_{k-1}(f_k - f_{k-1}) + \cdots + v_i(f_{i+1} - s) \\
&\geq v_k(T(s) - f_k) + v_{k-1}(f_k - f_{k-1}) + \cdots + v_j(f_{j+1} - t) \text{ since } i \leq j \text{ and } s \leq t.
\end{aligned}
$$

This means,

$v_m(T(t) - f_m) \geq v_k(T(s) - f_k)$, where $k = m$, or

$v_m(T(t) - f_m) + v_{m-1}(f_m - f_{m-1}) + \ldots + v_k(f_{k+1} - f_k)\ v_k(T(s) - f_k)$, where $k < m$, or
$v_m\ (T(t) - f_m) \geq v_k(T(s) - f_k) + v_{k-1}(f_k - f_{k-1}) + \ldots + v_m(f_{m+1} - f_m)$, where $k > m$.
The third inequality violates the assumption $v_k \geq 0$ since $T(t)\ f_{m+1}$, so $k \leq m$.
Hence, $T(t) \geq T(s)$. □

Figure 1 and Figure 2 show the simple comparison of arrival times in the link travel time model and the flow speed model. The link travel time in Figure 1 and the flow speed in Figure 2 can be converted from each other with the given length of links. In Figure 1, if the link travel time decrement in the next of the time intervals is large enough, a vehicle can pass the other vehicle started earlier. While in Figure 2, no matter how large the flow speed increment would be in the next time intervals, a vehicle cannot come closer to the other vehicle started earlier.

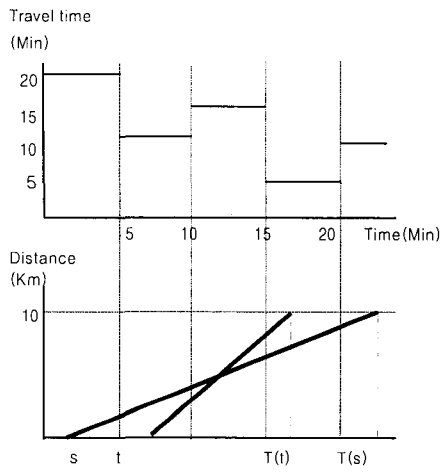

Figure 1. Arrival time in the link travel time model.

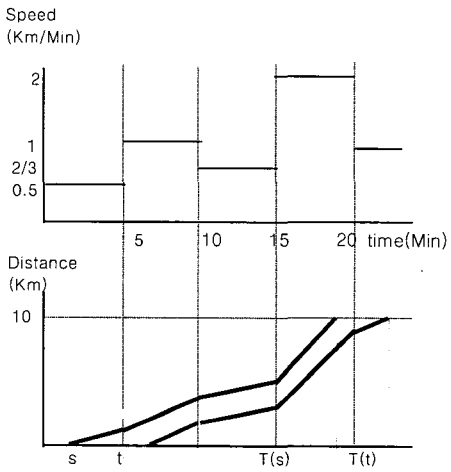

Figure 2. Arrival time in the flow speed model

We can extend the flow speed as being continuous function depending on time, instead of time intervals. That is, in a link (i, j) of length l, the vehicle speed changes as $v(t) \geq 0$ for any time t in time horizon. Then, a vehicle start ing at node i at time s will arrive at node j at time $T(s)$, where $T(s)$ is the value satisfying the equation $\int_s^{T(s)} v(t)dt = 1$. We can see that this arrival time also satisfies the NPP.

Kaufmann and Smith[7] showed that if all the link travel time satisfy the NPP in a time-dependent network, then the Principle of Optimality is also satisfied. Once the Principle of Optimality is satisfied, we can apply any short-

est path algorithm based on label-setting or label-correcting without additional computational complexity to the algorithm. So, we will apply a label-setting algorithm without further verification.

4. THE ALGORITHM

As one of several instances, we use Dijkstra's label-setting shortest path algorithm. In the algorithm, the travel time of each link is calculated according to the flow speed at the time of passing the link. We can easily combine the procedure of label-setting and link travel time calculation into one. It does not increase the computational complexity so much, and it does not need any additional memory storage. We only have to insert a function calculating the arrival time at the next connecting node of current node.

Let *Arr_time*$(d(i), (i, j))$ be the arrival time at node j starting from node i at time $d(i)$. Let o be the origin node, *pred*(i) be the predecessor node of node i. Let *A(i)* be the set of all links connected to node i. Then the combined Dijkstra's label-setting algorithm is described as follows.

Algorithm *Modified Dijkstra;*
Begin
$S:=\varnothing$; $U:=N$;
$d(i):=\infty$ for each node $i\in N$;
$d(o):=0$ and *pred*$(o):=0$;
while $|S| < n$ **do**
begin
let $i\in U$ be a node for which $d(i)=\min\{d(j): j\in U\}$;
$S:=S \cup\{i\}$;
$U:=U \{I\}$;
for each $(i, j)\in A(i)$ **do**
if $d(j) >$ *Arr_time*$(d(i), (i, j))$ then $d(j):=$ *Arr_time*$(d(i), (i, j))$ and *pred*$(j):=i$;
end;
end;

Function *Arr_time*$(d(i), (i, j))$;
Res_length:= $l_{(i,j)}$;
let $k\in\{0, 1, 2, ..., V\}$ be an index for which $f_{k(i,j)}\le d(i)\le f_{k+1(i,j)}$
Res_length:= *Res_length* - $v_{k(i,j)}\times(f_{k+1(i,j)}- d(i))$;

while *Res_length* > 0 **do**
begin
  k:=k+1;
  Res_length:= *Res_length* - $v_{k(i,j)}(f_{k+1(i,j)} - f_{k(i,j)})$;
end;
Arr_time:= $f_{k+1(i,j)}$ + *Res_length* / $v_{k(i,j)}$;
Return;

The computational complexity of the original Dijkstra's algorithm is $O(n^2+m)$ where n and m are the number of node and link in the network respectively. That of the modified algorithm is $O(n^2+mv)$ where v is the maximum number of time fences scanned in the function *Arr_time*. We can rebuild the function *Arr_time* so that each node remembers the index of the time fence, after which the node is arrived at, to reduce the fence scanning. Then the computational complexity of the modified algorithm is reduced to $O(n^2+nV)$ where V is the number of time fences defined on the time horizon.

The appropriate number of the time fences can be determined according to the situation of the real transportation network and the whole length of the time horizon we are concerning. When the fluctuation of the link travel time is very sever and frequent, the time interval should be short enough for the accurate observation and approximation of the link travel time. The shorter the time interval and the longer the whole time horizon, the more time fences are needed. In the application of the practical transportation networks, the typical time interval length is 5 to 10 minutes and the typical whole time horizon length is about 5 to 6 hours, including the rush-hours of the day. Hence the number of the time fences will not exceed one hundred in the practical applications.

5. NUMERICAL EXAMPLE

A simplified urban network is presented in Figure 3 as a numerical example. Table 1 represents the flow speed of each link in the example network in each time interval. Table 2 shows the solutions of the example network obtained by the algorithm suggested. According to the table, we can see that the NPP is satisfied, although the travel time from the origin to each node changes as the starting time from the origin changes.

Table 1. Flow speed of each link at each time fence
(Speed: Km/hour, Time: Min)

Links	Time intervals								
	0 ~	10~	20~	30~	40~	50~	60~	70~	80~
(o,a)	40	40	40	40	40	40	40	40	40
(o,b)	60	30	10	20	40	60	60	60	60
(a,b)	60	40	30	40	60	60	60	60	60
(a,c)	40	40	40	40	40	40	40	40	40
(b,c)	60	60	60	60	40	30	60	60	60
(b,d)	60	60	60	60	60	30	10	30	60
(c,d)	40	40	40	40	40	40	40	40	40

Table 2. Solutions to the example problem by the FSM

Start time	Node b			Node c			Node d		
	Path	Arrival time	Travel time	Path	Arrival time	Travel time	Path	Arrival time	Travel time
0	o-b	10	10	o-b-c	20	20	o-b-d	20	20
5	o-b	20	15	o-b-c	30	25	o-b-d	30	25
10	o-b	40	30	o-a-c	40	30	o-b-d	50	40
15	o-a-b	43 $\frac{1}{3}$	28 $\frac{1}{3}$	o-a-c	45	30	o-a-b-d	56 $\frac{2}{3}$	41 $\frac{2}{3}$
20	o-a-b	46 $\frac{2}{3}$	26 $\frac{2}{3}$	o-a-c	50	30	o-a-c-d	65	45
25	o-b	48 $\frac{3}{4}$	23 $\frac{3}{4}$	o-a-c	55	30	o-a-c-d	70	45
30	o-b	50	20	o-a-c	60	30	o-a-c-d	75	45
35	o-b	51 $\frac{2}{3}$	16 $\frac{2}{3}$	o-a-c	65	30	o-b-d	80	45
40	o-b	53 $\frac{1}{3}$	13 $\frac{1}{3}$	o-b-c	66 $\frac{2}{3}$	26 $\frac{2}{3}$	o-b-c-d	81 $\frac{2}{3}$	41 $\frac{2}{3}$
45	o-b	56 $\frac{2}{3}$	11 $\frac{2}{3}$	o-b-c	68 $\frac{1}{3}$	23 $\frac{1}{3}$	o-b-c-d	83 $\frac{1}{3}$	38 $\frac{1}{3}$

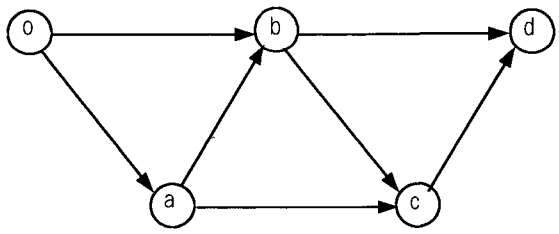

Figure 3. Example network 1 (Length of all links : 10Km)

For the comparison, we solved the problem after converting the problem into the travel time model. Table 3 and Table 4 show the converted travel time and the solution respectively. By Table 4, we can see that the NPP is violated.

Table 3. Travel time of each link at each time fence (Time: Min)

Links	Time intervals								
	0 ~	10~	20~	30~	40~	50~	60~	70~	80~
(o,a)	15	15	15	15	15	15	15	15	15
(o,b)	10	20	60	30	15	10	10	10	10
(a,b)	10	15	20	15	10	10	10	10	10
(a,c)	15	15	15	15	15	15	15	15	15
(b,c)	10	10	10	10	15	20	10	10	10
(b,d)	10	10	10	10	10	20	60	20	10
(c,d)	15	15	15	15	15	15	15	15	15

Table 4. Solutions to the example problem by the LTM

Start time	Node b			Node c			Node d		
	Path	Arrival time	Travel time	Path	Arrival time	Travel time	Path	Arrival time	Travel time
0	o-b	10	10	o-b-c	20	20	o-b-d	20	20
5	o-b	15	10	o-b-c	25	20	o-b-d	25	20
10	o-b	30	20	o-b-c	40	30	o-b-d	40	30
15	o-b	35	20	o-b-c	45	30	o-b-d	45	30
20	o-a-b	50	30	o-a-c	50	30	o-a-c-d	65	45
25	o-a-b	50	25	o-a-c	55	30	o-a-c-d	70	45
30	o-a-b	55	25	o-a-c	60	30	o-a-c-d	75	45
35	o-a-b	60*	25	o-a-c	65	30	o-a-c-d	80*	45
40	o-b	55*	15	o-a-c	70	30	o-b-d	75*	35
45	o-b	60	15	o-b-c	70	25	o-b-c-d	85	40

* The case of the violated NPP

6. Stochastic Time-dependent Model.

Here, we extend our flow speed model to the problem of the minimum expected time path in the time-dependent stochastic networks where the flow speeds at each link change statistically at each time interval. The stochastic model is defined as follows.

Consider a network $G=(N, A)$ with the node set $N=\{1, 2, ..., n\}$ and the link set $A\subseteq\{(i, j)\subseteq N\times N\}$. Let $l_{(i,j)}$ be the non-negative length of the link (i, j). Let $F=\{[f_k, f_{k+1}), k=0, 1, 2, .., V-1\}$ be the set of time intervals divided by the time fences $f_0<f_1<f_2 .....<f_{v-1}<f_v$ defined on the time horizon. Let $v_{k(i,j)}$ be the randomly changing non-negative flow speed at the time interval $[f_k, f_{k+1})$ on

link (i, j) and υ be a $|A| \times |F|$ vector of speeds $v_{k(i,j)}$. Let V be the set of all the possible states of υ, and $Prob(\upsilon)$ be the probability of the state υ, such that $\sum_{\upsilon \in V} Prob(\upsilon)=1$. Let P_j be a set of paths from node 1 to node j. For a state $\upsilon \in V$ and a path $P \in P_j$, define a value $d(j \mid \upsilon, P)=T(d(i \mid \upsilon, P), (i, j) \mid \upsilon, P)$, where $T(d(i \mid \upsilon, P), (i, j) \mid \upsilon, P)$ is the arrival time at node j starting from node i at time $d(i \mid \upsilon, P)$, with speed υ. The node i is a predecessor of node j on path P. The arrival time $T(d(i), (i, j))$ is same with the one mentioned in Section 4. Then, as a result of recursive calculation, $d(j \mid \upsilon, P)$ is the arrival time at node j, starting from node 1 at time $d(1)$, going through path P with speed υ. The objective of the model is to find a path P_j^* for all node $j \in N$, $j \neq 1$, such that $E[d(j \mid P_j^*)]=\text{Min}_{P \in Pj}\{E[d(j \mid P)]\}$, where $E[d(j \mid P)]=\sum_{\upsilon \in V} d(j \mid \upsilon, P) Prob(\upsilon)$. The value $E[d(j \mid P)]$ means the expected arrival time at node j going through a path P starting from node 1 at time $d(1)$.

As mentioned in the introduction, Hall[5] showed that the standard shortest path algorithms do not find the minimum expected travel time path. In the stochastic time-dependent travel time model, each link travel time is affected by the travel time of the previous links. That means the travel times of each link at each time interval are not independent to each other, even if they are statistically independent. We cannot use the expected value of a link travel time to calculate that of a path travel time, since $E[T(i, j)+T(j, k)] \neq E[T(i, j)]+E[T(j, k)]$, where $T(i, j)$ is the stochastic travel time of link (i, j). This makes it difficult to check the optimality of a candidate sub-path, and eventually to find the optimal path.

This dependency problem arises with both the link travel time model and the flow speed model. The network in Figure 4 shows the problem of dependency. Table 5 and Table 6 represent stochastic time-dependent link travel time and flow speed of the example network respectively. The link travel time and flow speed are equivalent to each other. In the Table 5, the expected travel time of path *a-b-c* is $(10+10) \times 0.5 + (30+30) \times 0.5 = 40$ minutes, while the expected value calculated with expected travel time of links is $(10 \times 0.5 + 30 \times 0.5) + 30 = 50$ minutes. In the Table 6, the expected travel time of path *a-b-c* is 40 minutes, while the expected value calculated with the expected flow speeds of each link is 35 minutes.

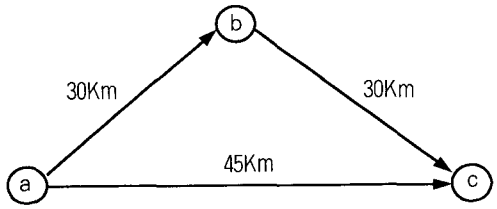

Figure 4. Example network 2

Table 5. Stochastic time-dependent link travel time (Minute)

Links	Time intervals	
	0~	20~
(a,b)	Prob(10)=0.5 Prob(30)=0.5	Prob(10)=0.5 Prob(30)=0.5
(b,c)	Prob(10)=1	Prob(30)=1
(a,c)	Prob(45)=1	Prob(45)=1

Table 6. Stochastic time-dependent flow speed (Km/Min)

Links	Time intervals	
	0~	20~
(a,b)	Prob(3)=0.5 Prob(1)=0.5	Prob(3)=0.5 Prob(1)=0.5
(b,c)	Prob(3)=1	Prob(1)=1
(a,c)	Prob(1)=1	Prob(1)=1

Here, we suggest a solution method using the K shortest path algorithm. For a link (i, j), let $V_{k(i,j)}$ be the set of all possible states of $v_{k(i,j)}$ and $Prob(v_{k(i,j)})$ be the probability of the state $v_{k(i,j)}$ such that $\sum_{v_{k(i,j)} \in V_{k(i,j)}} Prob(v_{k(i,j)})=1$, and let a vector $\boldsymbol{v}_{max}=(v'_{k(i,j)})$ where $v'_{k(i,j)}=\text{Max}\{v_{k(i,j)} \mid Prob(v_{k(i,j)})>0\}$. $\boldsymbol{v}_{max}$ is a vector of speeds that are the maximum possible value of the speed in each link in all time intervals. So it is impossible to access node j spending less time than $d(j \mid \boldsymbol{v}_{max}, P)$ for any $P \in \boldsymbol{P}_j$. Then, for any $P \in \boldsymbol{P}_j$, $d(j \mid \boldsymbol{v}_{max}, P)$ is a lower bound of $d(j \mid \boldsymbol{v}, P)$, for all $\boldsymbol{v} \in \boldsymbol{V}$, and hence that of $E[d(j \mid P)]$ as well.

Let $P^k \in \boldsymbol{P}_j$ be the kth path when paths are selected one by one in increasing order of $d(j \mid \boldsymbol{v}_{max}, P)$, then $d(j \mid \boldsymbol{v}_{max}, P^k)$ is the lower bound of $E[d(j \mid P^l)]$ for all paths $P^l \in \boldsymbol{P}j$, $l > k$, these are not yet selected. By definition, $E[d(j \mid P)]$ of any paths $P \in \boldsymbol{P}_j$ is an upper bound of $E[d(j \mid P^*)]$ where $P^* \in \boldsymbol{P}_j$ is the minimum expected time path in $\boldsymbol{P}_j$. So, we can establish a solution procedure for the minimum expected time path as follows.

Procedure for *Minimum Expected Time Path*

Step *0* : Let $k=0$, $P=\varnothing$, $U=\infty$, $L=0$.

Step *1* : $k=k+1$.
Find the kth shortest path P^k with speeds given $\boldsymbol{v}_{max}$.

Step *2* : If $d(j \mid \boldsymbol{v}_{max}, P^k) > L$, then $L = d(j \mid \boldsymbol{v}_{max}, P^k)$.
If $E[d(j \mid P^k)] < U$, then $U = E[d(j \mid P^k)]$, $P^*=P^k$.

Step *3* : If $L \geq U$ then let P^* is the optimal path and terminate.
Else go to step *1*.

The iterative procedure above is graphically presented in Figure 5. The lower bound of the expected travel time of the paths not yet selected increases as the number of paths selected increases. On the contrary, the upper bound of the minimum expected travel time decreases as the number of the selected paths increases. When the two bounds meet, the path corresponding to the least current upper bound is the minimum expected time path.

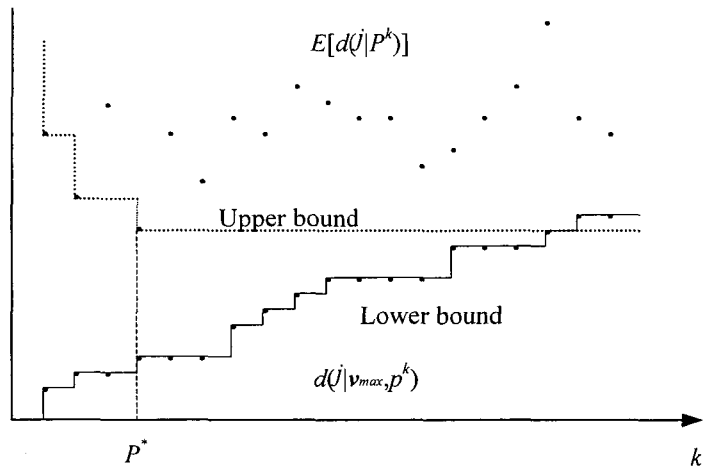

Figure 5. Graphical representation of the procedure for minimum expected time path

There are several algorithms to find the kth shortest path such as Azevedo[1], Dreyfus[4], Katoh[6], Martins[9] and Yen[11]. The computational complexities of Dreyfus' and Yen's algorithm are $O(kn\log n)$ and $O(kn^3)$ respectively. Those algorithms rank the shortest paths with and without loops respectively. In the procedure above, we can use any kind of ranking algorithm since the lengths of the kth ranked paths give only the kth stepwise lower bounds. But the problem is that we do not know how many paths should be searched to obtain an optimum solution. Theoretically, in the worst case, the number of paths increases exponentially as that of nodes in the network increases. But in the real transportation networks, it will not show the case theoretically worst. So, the experimental investigation about the trend of the parameter k in real transportation networks is needed as a further research.

7. CONCLUSIONS.

In this paper, we suggested a model of shortest paths in time-dependent networks, where flow speeds of each link depend on time interval. The model is realistic and useful since it maintains the NPP. Solving the problem needs just a little more computational complexity, and the same memory complexity, as the general shortest path problem. A solution algorithm that is modified from Dijkstra's label setting algorithm is presented.

We extended this model to the problem of finding the minimum expected time paths in time-dependent stochastic networks, where the flow speeds of each link change statistically on each time interval. Also we commented a solution method using the kth shortest path algorithm.

REFERENCES

[1] Azevedo J.A., Madeira J.J.E.R.S. and Martins E. Q. V., "A computational improvement for a shortest paths ranking algorithm", *European Journal of Operational Research* 73 (1994), 188-191

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

[3] Cooke K. L. and Halsey E., "The Shortest Route Through a Network with Time-Dependent Inter-nodal Transit Times", *J. of Math. Anal. Appl.* 14 (1966), 493-498.

[4] Dreyfus S. E., "An Apprisal of Some Shortest-Path Algorithms", *Operations research* 17 (1969), 395-412.

[5] Hall R. W., "The Fastest Path through a Network with Random Time-Dependent Travel Times", *Transportation Science* 20 (1986), 182-188.

[6] Katoh N., Ibaraki T. and Mine H., "An Efficient Algorithm for K shortest Simple Paths", Networks 12 (1982), 411-427

[7] Kaufman D. E. and Smith R. L., "Fastest Paths in Time-Dependent Networks for IVHS Application", *IVHS Journal* 1 (1993), 1-12.

[8] Mahmassani H.S., Hu T., Peeta S. and Ziliaskopoulos A., "Development and Testing of Dynamic Traffic Assignment and Simulation Procedures for ATIS/ATMS Applications". Technical Report DTFH61-90-C-00074-FG. CTR, The University of Texas at Austin 1994.

[9] Martins E. Q. V., "An Algorithm for Ranking Paths that may contain Cycles", *European Journal of Operational Research* 18 (1984), 123-130

[10] Orda A. and Rom R., "Shortest-Path and Minimum-Delay Algorithms in Networks with Time-Dependent Edge-Length", *J. of the Assoc. Comp. Mach.* 37 (1990), 607-625.

[11] Yen J.Y., "Finding the K shortest Loopless Paths in a Network", *Management Science* 17 (1971), 712-716.

[12] Ziliaskopoulos A. K. and Mahmassani H. S., "Time-Dependent, Shortest-Path Algorithm for Real-time Intelligent Vehicle Highway System Applications", *Transportation Res. Rec.* 1408 (1993), 94-100.