MINIMUM PATH SEARCH PROBLEMS IN NETWORKS UNDER UNCERTAIN TRAVEL TIME

Kenetsu UCHIDA[1] and Ryuichi TANI[2]

[1]Member of JSCE, Faculty of Engineering, Hokkaido University
(Kita 13, Nishi 8, Sapporo, Hokkaido, Japan)
E-mail: uchida@eng.hokudai.ac.jp
[2]Member of JSCE, Faculty of Engineering, Hokkaido University
(Kita 13, Nishi 8, Sapporo, Hokkaido, Japan)
E-mail:r-tani@eng.hokudai.ac.jp

It is thought that the reliable path problem that finds the path with the minimum travel cost, which is a linear combination of mean and variance of path travel time (the m-v shortest path problem) in a correlated network, can be formulated as neither an additive problem nor a linear programming problem. The path travel cost in the additive problem is defined as the sum of the link travel costs related to that path. In this study, by applying the flow conservation low proposed in Uchida (2015), it is shown that the m-v shortest path problem in a correlated network can be formulated as a linear programming problem in a hypergraph to which efficient solution algorithms are already provided. In addition, by utilizing the property of hyper-link flows in a hypergraph, it is shown that the m-v shortest path problem in this study can be formulated as a nonlinear optimization problem. The solution algorithm for the optimization problem is also presented. In this study, the reliable path problem that finds the path with minimum travel cost, a linear combination of the mean and standard deviation of path travel time (the m-s shortest path problem) in a correlated network, is formulated as a concave minimization programming problem. A global optimization algorithm for the m-s shortest path problem developed based on the linear programming problem is also presented. Numerical experiments are carried out to demonstrate the algorithm proposed in this study.

Key Words : *reliable path, stochastic travel time, hypergraph, total unimodularity*

1. INTRODUCTION

The shortest travel cost path search is carried out when we analyze transport network system. If the travel cost of a path in a network is additive in which the path travel cost is defined as the sum of the link travel costs related to that path, standard minimum path search algorithms based on node labeling can be applied for finding it. The reliable path search problems that find the path with the minimum travel cost, which is a linear combination of mean and variance/standard deviation of path travel time, are known as the non-additive and nonlinear problems. Different reliable path problems can be formulated based on a travel time reliability criterion. The reliable path can be the path with the minimum of a higher percentile value of travel time (Frank, 1969; Nie and Wu, 2009), the path with the maximum probability of on-time arrival (Fan et al., 2005a; Nie and Fan, 2006; Nie and Wu, 2009), the path with minimum mean-excess travel time (Chen and Zhou, 2010), as well as the path with minimum of travel cost that is a linear combination of travel time mean and standard deviation (or variance) of path travel time (Hall, 1983; Lo and Tung, 2003; Xing and Zhou, 2011; Khani and Boyles, 2015).

The reliable path search problem in a network in which the path travel cost that is a linear combination of mean and variance of path travel time is minimized (the m-v shortest path problem) has been studied in the literature. The reliable path problem in a network in which the path travel cost that is a linear combination of the mean and standard deviation of path travel time is minimized (the m-s shortest path problem) has also been studied in the literature. Even though the m-v shortest path problem in a non-correlated network where every link travel time is statistically independent is formulated as a mixed-integer quadratic programming problem that is difficult to solve in a large network. Sen et al. (2001) proposed a continuous relaxation method that produces a set of paths from which a solution of the m-v shortest path problem can be found. Khani and Boyles (2015) proposed an exact algorithm for the m-s shortest path problem in a non-correlated network. This algorithm finds a solution by repeatedly solving a subproblem that is formulated as the m-v shortest path problem. It is known that the m-s shortest path problem is

equivalent to the α-reliable path problem when path travel times follow normal distributions (Chen and Ji, 2005). Statistically, independent link travel times in a network are assumed in the studies that address reliable path problems (Fan et al. 2005a; Nie and Wu, 2009; Boyles and Waller, 2010; Khani and Boyles, 2015; Chen et al., 2016).

On the other hand, some studies assumed the statistically correlated link travel times in a network (Fan et al., 2005b; Nie and Fan, 2006; Sen et al., 2001; Hutson and Shier, 2009; Shahabi et al., 2013, 2014; Zockaie et al., 2013, 2014; Srinivasan et al., 2014). In these studies, however, only the correlation between two link travel times which are located within a certain distance is considered. The m-s shortest path problem in a correlated network was addressed, and several methods for solving the problem were proposed in the literature. Xing and Zhou (2011) proposed a Lagrangian relaxation method by which the problem is decomposed into three easier sub-problems to solve. Zeng et al. (2015) proposed a similar method to one proposed in Xing and Zhou (2011) that employs a Lagrangian relaxation method based on Cholesky decomposition. Zhang et al. (2017) proposed a Lagrangian relaxation method based on Cholesky decomposition by which the problem is decomposed into two easier sub-problems to solve. Rostami et al. (2018) developed a branch-and-bound algorithm by which both the upper and lower bounds of the m-s shortest path problem can be estimated. Zhang and Khani (2019) proposed a method that adopts both a Lagrangian substitution and an eigendecomposition technique.

In this study, both the m-v shortest path problem and the m-s shortest path problem in a network are studied. The m-s shortest path problem has been widely used in the literature since the standard deviation of a stochastic path travel time has the same unit as the stochastic path travel time in a network. However, since the solution algorithm of the m-v shortest path problem can be used in solving the m-s shortest path problem, we address both the m-v shortest path problem and the m-s shortest path problem in this study. We assume that every link travel time in the network is statistically correlated. This assumption is supported by the studies that address travel demand uncertainty in a framework of the network flow model (Lam et al., 2008; Uchida, 2014; Uchida, 2015). It is also assumed that the travel time covariance between two different links and the mean and variance of each link travel time in the network are given. The remainder of this paper is structured as follows. In section 2, we show the notation used in the paper. In section 3, by applying the flow conservation low proposed in Uchida (2015), it is shown that the m-v shortest path problem in a correlated network can be formulated as a linear programming problem. In section 4, it is shown the m-v shortest path problem in this study can be formulated as a problem that minimizes a linear function of hyperlink flows in a hypergraph subject to equality constraints. It is also shown that the coefficient matrix that is used in the equality constraints is a totally unimodular matrix, and therefore the m-v shortest path problem in this study has an integer solution. In addition, by utilizing the property of hyperlink flows in a hypergraph, it is shown that the m-v shortest path problem in this study can be formulated as a nonlinear optimization problem. The solution algorithm for the optimization problem is also presented. In section 6, the m-s shortest path problem in a correlated network is formulated as a concave minimization programming problem. A global optimization algorithm for the problem developed based on the linear programming problem is also presented. In section 6, numerical experiments are carried out to demonstrate the algorithm proposed in this study. In section 7, some concluding remarks are provided.

2. NOTATION AND ASSUMPTIONS

The following notations are used in this study.

A:	set of links in network
$\|A\|$:	number of links in A
N:	set of nodes in network
$\|N\|$:	number of nodes in N
o:	origin node
d:	destination node
T_j:	stochastic travel time of path j
μ_a:	mean of stochastic travel time of link a
σ_a:	standard deviation of stochastic travel time of link a
ρ_{ab}:	coefficient of correlation of travel times of links a and b
L_n^{in}:	set of links each of which has head (or destination) node of n
L_n^{out}:	set of links each of which has tail (or origin) node of n
$\omega_{a\hat{}b}$:	flow which passes through both links a and b in network ($\omega_{a\hat{}b} = \omega_{b\hat{}a}$) if $a \neq b$ (or flow of hyperlink $a\hat{}b$ in hypergraph)
$\omega_{a\hat{}a}$:	flow of link a (or flow of hyperlink $a\hat{}a$ in hypergraph)

The following assumptions are employed in this study. Only one link is directed from an origin node. In a similar way, only one link is directed to a destination node in a network. These two assumptions are always true if we add two nodes and two links, i.e.,

the origin and destination nodes and the corresponding two links, to a network to be analyzed. Hereinafter, the link directed from the origin node and the link directed to the destination node are referred to as the origin link and the destination link in this study, respectively.

3. SHORTEST PATH PROBLEM

The shortest path search problem in a directed network formulated based on linear programming is

$$\min \sum_{a\in A} \mu_a \cdot x_a \quad (1)$$

w.r.t. $x_a \ \forall a \in A$

s.t.

$$\sum_{a\in L_n^{in}} x_a - \sum_{b\in L_n^{out}} x_b = \begin{cases} -1 & \text{if } n = o \\ 1 & \text{if } n = d \\ 0 & \text{otherwise} \end{cases} \quad \forall n \in N \quad (2)$$

$$x_a \geq 0 \ \forall a \in A$$

The constraints of (2) for an example network shown in Figure1 are expressed by using a coefficient matrix $\boldsymbol{A}$ and two row vectors, $\boldsymbol{x}$ and $\boldsymbol{b}$, as follows

$$\boldsymbol{A}\boldsymbol{x}^T = \boldsymbol{b}^T \quad (3)$$

where

$$\boldsymbol{A} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 \\ 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\boldsymbol{x} = (x_1 \quad \dots \quad x_4), \boldsymbol{b} = (-1 \quad 0 \quad 0 \quad 1)$$

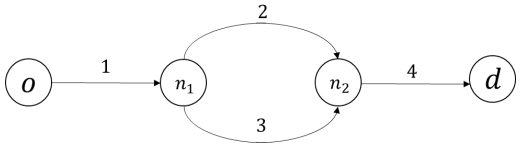

Figure1 An example network.

The objective function is of the linear form and $\boldsymbol{b}$ is a vector with integer entries. As discussed in the latter section, the node-link incident matrix of the directed network $\boldsymbol{A}$ is known as a totally unimodular matrix, the problem has an integer solution without introducing integer constraints to the problem.

4. THE M-V SHORTEST PATH PROBLEM

The m-v shortest path problem in a correlated network can be formulated as the following integer programming problem (IP):

$$\min g_1 = \sum_{a\in A} \mu_a \cdot x_a + \gamma_1 \cdot \sum_{a\in A} \sum_{b\in A} \psi_{ab} \cdot x_a \cdot x_b \quad (4)$$

w.r.t.

$$x_a = \{0, 1\} \ \forall a \in A \quad (5)$$

s.t.

$$\sum_{a\in L_n^{in}} x_a - \sum_{b\in L_n^{out}} x_a = \begin{cases} -1 & \text{if } n = o \\ 1 & \text{if } n = d \\ 0 & \text{otherwise} \end{cases} \quad \forall n \in N \quad (6)$$

where

$$\psi_{ab} = \rho_{ab} \cdot \sigma_a \cdot \sigma_b \ (\psi_{aa} = \sigma_a^2).$$

γ_1 in (4) represents the value of path travel time reliability (measured by variance) relative to the expected travel time. x_a is the binary decision variable that equals one if link a is part of the m-v shortest path and zero otherwise. The integer constraints shown by (5) make the problem difficult to solve.

On the other hand, the m-v shortest path problem in this study is formulated as the following linear programming problem

$$\min g_2 = \sum_{a\in A} \mu_a \cdot \omega_{a\text{^}a} + \gamma_1 \cdot \sum_{a\in A} \sum_{b\in A} \psi_{ab} \cdot \omega_{a\text{^}b} \quad (7)$$

w.r.t. $\omega_{a\text{^}b} \ \forall a, b \in A$,

s.t.

$$\begin{cases} \sum_{b\in L_n^{in}} \omega_{b\text{^}b} - \sum_{c\in L_n^{out}} \omega_{c\text{^}c} = -1 & \text{if } n = o \\ \sum_{b\in L_n^{in}} \omega_{b\text{^}b} - \sum_{c\in L_n^{out}} \omega_{c\text{^}c} = 1 & \text{if } n = d \\ \sum_{b\in L_n^{in}} \omega_{a\text{^}b} - \sum_{c\in L_n^{out}} \omega_{a\text{^}c} = 0 \ \forall a \in A & \text{otherwise} \end{cases} \quad (8)$$

$$\omega_{a\text{^}b} = 0 \text{ if } \{a, b\} \subseteq L_n^{out} \ \forall n \in N \quad (9a)$$

$$\omega_{a\text{^}b} = 0 \text{ if } \{a, b\} \subseteq L_n^{in} \ \forall n \in N \quad (9b)$$

$$\omega_{a\text{^}b} = \omega_{b\text{^}a} \ \forall a, b(\neq a) \in A \quad (10)$$

$$\omega_{a\text{^}b} \geq 0 \ \forall a, b \in A \quad (11)$$

The m-v shortest path problem in this study can be solved by assigning the origin-destination (o-d) demand of one to the shortest path between the o-d pair in a network. The linear function (7) is the total m-v travel time cost to be minimized. The o-d demand assigned to a path or a link can be regarded as path choice probability or link choice probability, respectively. The total m-v travel time cost is calculated using the given mean link travel times and variance and covariance of link travel times in the network.

On (8), the first and second equality constraints show that the o-d demand flows out from the origin node o via links in the set L_o^{out} and flows into the destination node d via links in the set L_d^{out}. The third equality constraint is introduced for the calculation of travel time covariance. It shows that the sum of link flows that pass through link $a \in A$ and flow into node n via links in the set L_n^{in} has to flow out from the node via links in the set L_n^{out}, c.f., (14) in Uchida (2015). Accompanied with two boundary conditions which are the first and second equality constraints of (8), the third equality constraint enables to calculate the flow that passes through two different links a and b,

$\omega_{a\hat{}b}$ $\forall a, b(\neq a) \in A$, without enumerating paths from the origin node to the destination node. Such flow costs $\gamma_1 \cdot \psi_{ab} \cdot \omega_{a\hat{}b}$ in the objective function. On the other hand, the flow on link a, $\omega_{a\hat{}a}$ $\forall a \in A$, costs $(\mu_a + \gamma_1 \cdot \psi_{aa}) \cdot \omega_{a\hat{}a}$ in the objective function. The number of equality constraints in (8) is $2 + |A| \cdot (|N| - 2)$. The number of equality constraints from the third constraint in (8) can be large if a network is large. In such a network, however, in most of the links in A that seem not to be used by the virtual o-d demand, both sides of the equality condition become zero. Therefore, depending on the predefined virtual o-d demand, one can reduce network size to reduce the number of equality constraints denoted by the third constraint in (8).

(9) shows that no flow passes through different two links in the set of L_n^{in} or L_n^{out} which is defined at node $n \in N$. The number of equality constraints by (9) is determined depending on the network topology addressed. (10) represents that the symmetric entries have the same value. From (10), if (9) is not considered, the number of unknown variables of this problem is $\sum_{k=1}^{|A|} k = \frac{1}{2} \cdot |A| \cdot (|A| + 1)$ where $|A|$ is the number of links in the network.

5. THE M-V SHORTEST PATH PROBLEM EXPRESSED IN A HYPERGRAPH

(1) Total unimodularity of the constraints (8)-(10)

In this section, it is shown that the m-v shortest path problem can be regarded as a linear programming problem in a hypergraph which is comprised of hyperlinks. Each hyperlink in a hypergraph in this study has at most three nodes which are tail node(s) and head node(s). Based on the discussion on the problem in a hypergraph, it is also shown that the m-v shortest path problem studied in the previous section has the integer solution. In the following, the formulation of the m-v shortest path problem in a hypergraph as well as the illustrative numerical expressions for the network shown in Figure1, are presented.

The constraints shown by (8) and (9) for the example network in Figure1 are shown as

$$\boldsymbol{H}_p \boldsymbol{\omega}^T = \boldsymbol{h}_p^T \tag{12}$$

where

$$\boldsymbol{H}_p = \begin{pmatrix} \boldsymbol{H}_1 & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{H}_4 \end{pmatrix}$$

$$\boldsymbol{H}_1 = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 \\ 0 & 1 & 1 & -1 \end{pmatrix}$$

$$\boldsymbol{H}_a = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix} \forall a \in \{2,3\}$$

$$\boldsymbol{H}_4 = \begin{pmatrix} 1 & -1 & -1 & 0 \\ 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\boldsymbol{\omega} = (\boldsymbol{\omega}_1 \quad \dots \quad \boldsymbol{\omega}_4)$$

$$\boldsymbol{\omega}_a = (\omega_{a\hat{}1} \quad \dots \quad \omega_{a\hat{}4}) \; \forall a \in \{1,4\}$$

$$\boldsymbol{\omega}_2 = (\omega_{2\hat{}1} \quad \omega_{2\hat{}2} \quad \omega_{2\hat{}4})$$

$$\boldsymbol{\omega}_3 = (\omega_{3\hat{}1} \quad \omega_{3\hat{}3} \quad \omega_{3\hat{}4})$$

$$\boldsymbol{h_p} = (\boldsymbol{h}_1 \quad \dots \quad \boldsymbol{h}_4)$$

$$\boldsymbol{h}_1 = (-1 \quad 0 \quad 0)$$

$$\boldsymbol{h}_a = (0 \quad 0) \; \forall a \in \{2,3\}$$

$$\boldsymbol{h}_4 = (0 \quad 0 \quad 1)$$

In this example, link1 is the origin link and link4 is the destination link. The coefficient matrix $\boldsymbol{H}_p$ for the network shown in Figure 1 is summarized in Table 1. $\boldsymbol{H}_p$ is a block diagonal matrix that can be regarded as a node-hyperlink incident matrix of the hypergraph shown in Figure2. This hypergraph is comprised of four sub-hypergraphs. Each sub-hypergraph in the figure is disconnected from each other. In this study, the sub-hypergraph that is situated at the top of the figure is referred to as the origin link-specific hypergraph. The sub-hypergraph that is situated at the bottom of the figure is referred to as the destination link-specific hypergraph. The second and third sub-hypergraphs in the figure are referred to as the link2-specific hypergraph and the link3-specific hypergraph, respectively. Link$a$-specific hypergraph ($a \in \{2,3\}$) has three hyperlinks, i.e., $a\hat{}1$, $a\hat{}2$ (or $a\hat{}3$) and $a\hat{}4$, and two nodes, i.e., a_n_1 and a_n_2. The origin or destination link-specific hypergraph has an additional node of o or d, respectively. The flow on hyperlink $a\hat{}b$ ($b \in \{1,\dots,4\}$) is $\omega_{a\hat{}b}$ that corresponds to the flow passing through both links a and b in the network shown in Figure1. Since $\omega_{2\hat{}3} = \omega_{3\hat{}2} = 0$ from (9a) or (9b), there is no hyperlink corresponding to such hyperlink flows in the link$a$-specific hypergraph ($a \in \{2,3\}$). The coefficient matrix of $\boldsymbol{H}_a$ is also regarded as a node-hyperlink incident matrix of the link$a$-specific hypergraph.

Table 1 Node-hyperlink incident matrix of the hypergraph

	1^1	1^2	1^3	1^4	2^1	2^2	2^4	3^1	3^3	3^4	4^1	4^2	4^3	4^4
o	-1													
1_n_1	1	-1	-1											
1_n_2		1	1	-1										
2_n_1					1	-1								
2_n_2						1	-1							
3_n_1								1	-1					
3_n_2									1	-1				
4_n_1											1	-1	-1	
4_n_2												1	1	-1
d														1

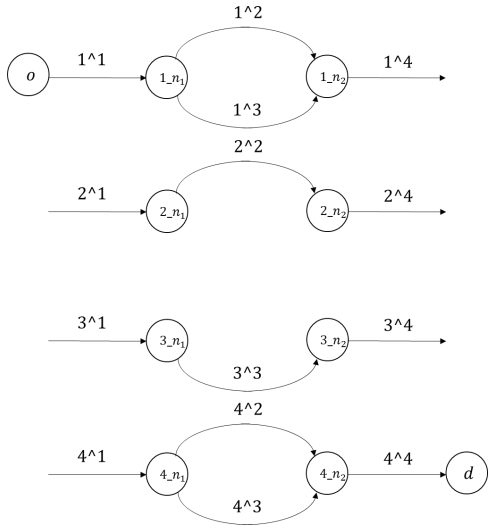

Figure 2 Hypergraph.

For expressing the constraints shown by (10), we introduce the following matrix

$$\boldsymbol{J} = \begin{pmatrix} \boldsymbol{J}_{12} \\ \boldsymbol{J}_{13} \\ \boldsymbol{J}_{14} \\ \boldsymbol{J}_{24} \\ \boldsymbol{J}_{34} \end{pmatrix}$$

where

$$\boldsymbol{J}_{ab} = \begin{pmatrix} \overbrace{\mathbf{0}}^{a\hat{}b} & \overbrace{1}^{} & \mathbf{0} & \overbrace{-1}^{b\hat{}a} & \mathbf{0} \end{pmatrix}$$
$$\{(a,b)|a \in \{1,\ldots,3\}, b > a, \{a,b\} \nsubseteq L_{n_1}^{out}\}$$

Note that, $\boldsymbol{J}_{23}$ is not shown in $\boldsymbol{J}$ since $\omega_{2\hat{}3} = \omega_{3\hat{}2} = 0$ from (9a) or (9b). Finally, the constraints shown by (8), (9) and (10) are denoted as

$$\boldsymbol{H}\boldsymbol{\omega}^T = \boldsymbol{h}^T \tag{13}$$

where

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{H}_p \\ \boldsymbol{J} \end{pmatrix}$$
$$\boldsymbol{g} = (\boldsymbol{g}_p \quad \mathbf{0})$$

The coefficient matrix of $\boldsymbol{H}$ for the network shown in Figure1 is summarized in Table 2. $\boldsymbol{H}$ can be regarded as a node-hyperlink incident matrix of the directed hypergraph shown in Figure 3. We call this directed hypergraph as the augmented hypergraph. Each link-specific hypergraph is now connected with each other by the hyperlinks and nodes added by (10). Also, each link-specific hypergraph has an origin node and a destination node. For example, the origin link-specific hypergraph has the origin node o and destination node C. The destination link-specific hypergraph has the origin node C and the destination node d. Each hyperlink in the augmented hypergraph has at most three nodes that are head node(s) and tail node(s). For example, hyperlink 1^2 has one tail node 1_n_1 and two head nodes 1_n_2 and A. A hyperlink in the augmented hypergraph is represented by each column of $\boldsymbol{H}$ or Table 2, where tail node(s) and head node(s) are denoted by -1 and 1, respectively.

Table 2 The coefficient matrix of $\boldsymbol{H}$.

	1^1	1^2	1^3	1^4	2^1	2^2	2^4	3^1	3^3	3^4	4^1	4^2	4^3	4^4
o	-1													
1_n_1	1	-1	-1											
1_n_2		1	1	-1										
2_n_1					1	-1								
2_n_2						1	-1							
3_n_1								1	-1					
3_n_2									1	-1				
4_n_1											1	-1	-1	
4_n_2												1	1	-1
d														1
A		1			-1									
B			1					-1						
C				1							-1			
D						1						-1		
E										1			-1	

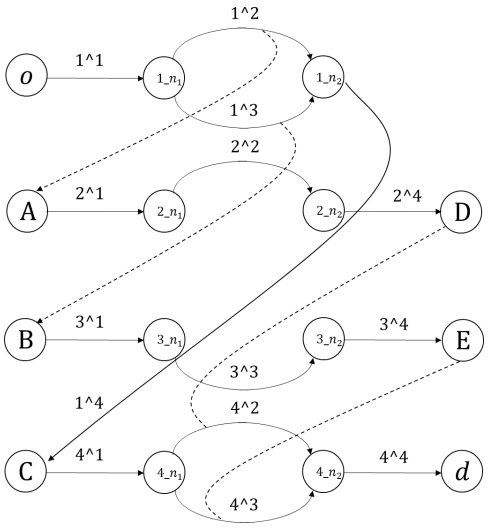

Figure 3 Augmented hypergraph.

We now suppose that each hyperlink flow in the augmented hypergraph is either zero or one. The augmented hypergraph and (13) show that the o-d flow of one is generated from the origin node o and flows into the origin link-specific hypergraph via the hyperlink 1^1. The o-d flow reaches its destination node C. If hyperlink $1\hat{}a$ $(a \in \{2,\ldots,4\})$ in the origin link-specific hypergraph carries flow, such hyperlink flow $\omega_{1\hat{}a}$ becomes the o-d flow for the link$a$-specific hypergraph, $\omega_{a\hat{}1}$, via the hyperlinks added by (10). For example, if hyperlink 1^2 carries flow, the corresponding hyperlink flow of $\omega_{1\hat{}2} = 1$ flows into both nodes 1_n_2 and A. Thus, the hyperlink flow flowing into node A becomes the o-d flow for the link2-specific hypergraph, $\omega_{2\hat{}1}$, and reaches its destination node D. Since hyperlink 1^3

carries no flow, in this case, all hyperlink flows in the link3-specific hypergraph carry no flow, i.e., $\omega_{3\hat{}a} = 0$ $\forall a \in \{1,3,4\}$. All hyperlink flows in the destination link-specific hypergraph are determined by the hyperlink flows $\omega_{a\hat{}4}$ $\forall a \in \{1,2,3\}$ that are calculated in the link$a$-specific hypergraph. For example, if hyperlink 2^4 in the link2-specific hypergraph carries the flow, i.e., $\omega_{2\hat{}4} = 1$, such flow becomes the hyperlink flow, $\omega_{4\hat{}2}$, in the destination link-specific hypergraph. As mentioned earlier, when the o-d demand of zero is assigned to the link$a$-specific hypergraph $\forall a \in \{2,3,4\}$, every hyperlink flow in the link$a$-specific hypergraph is zero, i.e., $\omega_{a\hat{}b} = 0$ $\forall b \in \{1,\ldots,4\}$ and that becomes the hyperlink flows in the other link-specific hypergraphs, i.e., $\omega_{b\hat{}a} = 0\ \forall b \in \{1,\ldots,4\}$. This property plays an important role when solving the m-v shortest path problem in this study.

Since the coefficient matrix $\boldsymbol{H}$ shown in (13) is totally unimodular to be shown, the m-v shortest path problem has the solution of $\omega_{a\hat{}b} \in \{0,1\} \forall a, b \in A$. This is true for a general directed network. We will discuss the total unimodularity of the coefficient matrix corresponding to a general directed network. The definitions of a unimodular matrix and a totally unimodular matrix are shown next.

A square nonsingular matrix $\boldsymbol{M} \in \mathbb{Z}^{n\times n}$ is unimodular if $det(\boldsymbol{M}) \in \{1,-1\}$. A matrix $\widetilde{\boldsymbol{A}} \in \mathbb{Z}^{m\times n}$ is totally unimodular if every square submatrix of $\widetilde{\boldsymbol{A}}$ is unimodular or singular, i.e., $det(\widetilde{\boldsymbol{A}}) \in \{0,1,-1\}$. Therefore, the necessary condition for the coefficient matrix $\boldsymbol{A}$ shown in (3) to be totally unimodular is

$$\boldsymbol{A} \in \{0,1,-1\}^{m\times n}$$

since the determinant of every one by one submatrix of $\boldsymbol{A}$ needs to be $0, 1$ or -1.

Theorem 1.
Node-link incident matrix $\boldsymbol{A}$ of a network shown in (3) is totally unimodular.

Lemma 1.
Every diagonal matrix, $\boldsymbol{H}_a$, for the link$a$-specific hypergraph in $\boldsymbol{H}_p$ shown in (12) is totally unimodular.

Proof is given in Appendix 1.

Consider next the coefficient matrix of the augmented hypergraph $\boldsymbol{H} \in \{0,1,-1\}^{m\times n}$ corresponding to a general directed network. From the definition, each column of $\boldsymbol{H}$ has at most three nonzero entries.

Lemma 2.
$\boldsymbol{H}$ is totally unimodular.

Proof is given in Appendix 2.

Corollary 1.
The m-v shortest path search problem has an integer solution.

Since $\boldsymbol{H}$ is totally unimodular and the objective function is of the linear form, the m-v shortest path search problem in this study has the integer solution. It goes without saying, the m-v shortest path problem formulated in this study can be solved by a standard algorithm of a linear programming problem.

6. NONLINEAR OPTIMIZATION PROBLEM FOR THE M-V SHORTEST PATH PROBLEM AND ITS ALGORITHM

Although it is shown that the m-v shortest path search problem in this study has an integer solution, the number of unknown variables in the problem becomes large when addressing a large network even though (9) and (10) reduce some unknown variables that are not needed in solving the problem. In the following, we will propose a formulation of the m-v shortest path search problem which is easier to solve problem for a large network. The number of unknown variables in the problem is $|A|$.

We consider first the solution of a linear programming problem defined for the origin link-specific hypergraph as:

$$\min d(\boldsymbol{\omega}_1)$$
w.r.t. $\boldsymbol{\omega}_1$ s.t. (3) and
$$\boldsymbol{\omega}_1 \geq \boldsymbol{0}. \tag{14}$$

Note that, $\boldsymbol{A}$ in (3) is totally unimodular since that is a node-link incident matrix of a directed network. As long as the objective function $d(\boldsymbol{\omega}_1)$ is of the linear form with respect to $\omega_{1\hat{}a}$ $\forall a \ni A$, the solution of this problem has an integer solution. Suppose that a basic solution of the problem is denoted as $\widehat{\omega}_{1\hat{}a} \forall a \in A$. The solution can be classified into two kinds of hyperlink flows, i.e., $\widehat{\omega}_{1\hat{}a} = 0\ \forall a \in A_{zero}$ and $\widehat{\omega}_{1\hat{}a} = 1\ \forall a \in A_{one}$ where A_{zero} and A_{one} ($A = A_{zero} \cup A_{one}$ and $A_{zero} \cap A_{one} = \{\emptyset\}$) are the sets of links in a directed network. The hyperlinks associated with links in A_{one}, $1\hat{}a\ \forall a \in A_{one}$, carry flow and those associated with links in A_{zero}, $1\hat{}a\ \forall a \in A_{zero}$, carry no flow in the origin link-specific hypergraph. The hyperlinks that carry flow $\widehat{\omega}_{1\hat{}a} = 1\ \forall a \in A_{one}$ comprise an acyclic path from the origin node o to its destination node in the origin link-specific hypergraph.

We will consider then the hyperlink flows in the link$a$-specific hypergraph, $\widehat{\omega}_{a\hat{}b} (a \neq 1)$, that are implicitly associated with the hyperlink flows in the augmented hypergraph. Consider first the hyperlink flows $\widehat{\omega}_{a\hat{}b}\ \forall b \in A$ in the link$a$-specific hypergraph

where $a \in A_{zero}$. Since $a \in A_{zero}$, therefore $\widehat{\omega}_{1\wedge a} = 0\ \forall a \in A_{zero}$. And thus, $\widehat{\omega}_{1\wedge a}$ becomes the o-d flow for the link$a$-specific hypergraph $\widehat{\omega}_{a\wedge 1}$. Finally, we obtain $\widehat{\omega}_{a\wedge b} = 0\ \forall a \in A_{zero}, \forall b \in A$. Consider next the hyperlink flows $\widehat{\omega}_{a\wedge b}\ \forall b \in A$ in the link$a$-specific hypergraph where $a \in A_{one}$. Some of the hyperlink flows in the link$a$-specific hypergraph are given as $\widehat{\omega}_{a\wedge b} = 0\ \ \forall b \in A_{zero}$. Since $\widehat{\omega}_{1\wedge a} = 0\ \ \forall a \in A_{zero}$, there exist hyperlink flows such that $\widehat{\omega}_{b\wedge a} = 0\ \forall a \in A_{zero}\ \forall b \in A$ in the link$b$-specific hypergraph. Since $\widehat{\omega}_{a\wedge b} = \widehat{\omega}_{b\wedge a}$, it is shown that $\widehat{\omega}_{a\wedge b} = 0\ \forall b \in A_{zero}$. The rest of the hyperlink flows in the link a -specific hypergraph are given as $\widehat{\omega}_{a\wedge b} = 1\ \forall b \in A_{one}$. Since the o-d flow must reach its destination node, the hyperlinks $a\wedge b\ \ \forall b \in A_{one}$ comprise an acyclic path from the origin node to the destination node in the link$a$-specific hypergraph. The path in the link$a$-specific hypergraph $\forall a \in A_{one}$ is the same as that in the origin link-specific hypergraph in terms of network topology. The insights provided above are summarized as follows by supposing $\widehat{\omega}_{1\wedge a} = 0\ \forall a \in A_{zero}$ and $\widehat{\omega}_{1\wedge a} = 1\ \forall a \in A_{one}$.

Lemma 3.
If $a \in A_{zero}$ then it is shown that $\widehat{\omega}_{a\wedge b} = 0\ \forall b \in A$. If $a \in A_{one}$ then it is shown that $\widehat{\omega}_{a\wedge b} = 0\ \forall b \in A_{zero}$ and $\widehat{\omega}_{a\wedge b} = 1\ \forall b \in A_{one}$.

The proof is given in Appendix 3.

By using this property, every hyperlink flow in each link-specific hypergraph can be calculated by using hyperlink flows in the origin link-specific hypergraph. The hyperlink flows in the link$a$-specific hypergraph except for the origin link-specific hypergraph are

$$\widehat{\omega}_{a\wedge b} = \begin{cases} 0 & \text{if } \widehat{\omega}_{1\wedge a} = 0 \\ \widehat{\omega}_{1\wedge b} & \text{if } \widehat{\omega}_{1\wedge a} = 1 \end{cases} \forall a(\neq 1) \in A, \forall b \in A$$

The conditions shown above are equivalent to

$$\widehat{\omega}_{a\wedge b} = \widehat{\omega}_{1\wedge a} \cdot \widehat{\omega}_{1\wedge b}\ \forall a(\neq 1) \in A, \forall b \in A$$

The conditions above show that the number of unknown variables in the m-v shortest path search problem in this study can be reduced to $|A|$. The corresponding problem by using the hyperlink flows in the origin link-specific hypergraph is formulated as

$$\begin{aligned} \min e(\boldsymbol{\omega}_1) &= \boldsymbol{\omega}_1(diag(\boldsymbol{\mu}) + \gamma_1 \cdot \boldsymbol{\Sigma})\boldsymbol{\omega}_1^T \\ &= \textstyle\sum_{a\in A} \mu_a \cdot \omega_{a\wedge a} + \gamma_1 \cdot \sum_{a\in A}\sum_{b\in A} \psi_{ab} \cdot \omega_{a\wedge b} \end{aligned} \quad (15)$$

w.r.t. $\boldsymbol{\omega}_1$, s.t. (3), (9a), (9b) and (14),
where

$$\omega_{a\wedge b} = \omega_{1\wedge a} \cdot \omega_{1\wedge b}\ \forall a \in, \forall b \in A \quad (16)$$

$$\boldsymbol{\mu} = (\mu_1 \quad \cdots \quad \mu_{|A|})$$

$\boldsymbol{\Sigma}$ in (15) is the variance-covariance matrix of link travel times in a directed network. In this problem, the o-d flow of one is generated from the origin node of o and finally reaches its destination node in the origin link-specific hypergraph. The objective function in this problem is no longer a linear function of the variables, $\omega_{1\wedge a}\ \forall a \in A$, because of (16). The second constraint does not allow the o-d flow flowing into the hypergraph to split at each node in the origin link-specific hypergraph. Therefore, the feasible solution is always $\omega_{1\wedge a} = \{0,1\}\ \forall a \in A$.

The Hessian of the objective function is $2 \cdot (diag(\boldsymbol{\mu}) + \gamma_1 \cdot \boldsymbol{\Sigma})$. Since $\boldsymbol{\Sigma}$ is a positive semidefinite matrix, i.e., $\boldsymbol{x}\boldsymbol{\Sigma}\boldsymbol{x}^T \geq \boldsymbol{0}\ \forall \boldsymbol{x} \neq \boldsymbol{0}$, and thus $2 \cdot \boldsymbol{x}(diag(\boldsymbol{\mu}) + \gamma_1 \cdot \boldsymbol{\Sigma})\boldsymbol{x}^T > 0\ \ \forall \boldsymbol{x} \neq \boldsymbol{0}$. This implies that the Hessian is a positive definite matrix. It follows that the objective function is strictly convex. Although, the constraints denoted by $\boldsymbol{A}\boldsymbol{\omega}_1^T = \boldsymbol{b}^T$ form a convex set, the constraints denoted by (9a) or (9b) do not form a convex set. Therefore, the problem generally has no unique solution because of the non-convexity and non-smoothness of the feasible set.

To cope with this property of the problem, we propose an algorithm which solves correctly the problem shown above (the problem that maximizes (15) s.t (3), (9a), (9b) and (14)) by iteratively solving the sub-problem that is obtained by linearly approximating the original objective function at a feasible solution. The objective function approximated at a feasible solution, i.e., at the nth iteration solution $\boldsymbol{\omega}_1^n$, is

$$\begin{aligned} &e(\boldsymbol{\omega}_1) \approx \tilde{e}(\boldsymbol{\omega}_1) = e(\boldsymbol{\omega}_1^n) + \nabla e(\boldsymbol{\omega}_1^n)(\boldsymbol{\omega}_1 - \boldsymbol{\omega}_1^n) \\ &= e(\boldsymbol{\omega}_1^n) - \sum_{a\in A} (\omega_{1\wedge a}^n - \omega_{1\wedge a}) \cdot \lambda_{1\wedge a}^n \\ &= e(\boldsymbol{\omega}_1^n) - \sum_{a\in A} \lambda_{1\wedge a}^n \cdot \omega_{1\wedge a}^n + \sum_{a\in A} \lambda_{1\wedge a}^n \cdot \omega_{1\wedge a} \end{aligned}$$

where

$$\begin{aligned} \lambda_{1^a}^n &= \frac{\partial e(\boldsymbol{\omega}_1^n)}{\partial \omega_{1^a}} \\ &= 2 \cdot \left((\mu_a + \gamma_1 \cdot \psi_{aa}) \cdot \omega_{1\wedge a}^n + \gamma_1 \sum_{b(\neq a)\in A} \psi_{ab} \cdot \omega_{1\wedge b}^n \right) \end{aligned}$$

The first and second terms of the right-hand side of the approximated function are constant terms. Therefore, the solution of the minimization problem of $\tilde{e}(\boldsymbol{\omega}_1)$ is obtained by minimizing $\sum_{a\in A} \lambda_{1\wedge a}^n \cdot \omega_{1\wedge a}$. The corresponding minimization sub-problem is formulated as

$$min \textstyle\sum_{a\in A} \lambda_{1\wedge a}^n \cdot \omega_{1\wedge a} \quad (17)$$

w.r.t. $\boldsymbol{\omega}_1$, s.t. (3) and (14).

The solution of this problem is obtained by assigning the o-d demand of one to the minimum path between o-d pair in the origin link-specific hypergraph where each hyperlink cost is given by $\lambda_{1\hat{}a}^{n}$. The solution of the problem shown above, i.e., $\boldsymbol{\omega}_1^{n+1}\left(\omega_{1\hat{}a}^{n+1} \in \{0,1\}\ \forall a \in A\right)$, is the feasible solution at the next iteration. Since the o-d flow does not split at each node in the link-specific hypergraph, the constraints shown by (9a) or (9b) always hold. Therefore, $\boldsymbol{\omega}_1^{n+1}$ is regarded as an updated basic solution to the original linear programming problem. The value of the objective function (15) can be calculated from each basic solution $\boldsymbol{\omega}_1^{n+1}$. As long as the value of the objective function decreases, the basic solution is updated by solving the sub-problem formulated above. The algorithm proposed in this study is summarized as follows.

Algorithm 1.

Step 0: Initialization
Set $n = 0$. Set an initial feasible solution $\boldsymbol{\omega}_1^n$ and calculate the value of $e(\boldsymbol{\omega}_1^n)$.
Step 1: Solve the sub-problem
Solve the sub-problem that minimizes (17) s.t. (3) and (14). The solution is denoted as $\boldsymbol{\omega}_1^{n+1}$.
Step 2: Termination and the solution update
If $\nabla e(\boldsymbol{\omega}_1^n)(\boldsymbol{\omega}_1^{n+1} - \boldsymbol{\omega}_1^n) \geq 0$, then the solution of the original problem is $\boldsymbol{\omega}_1^n$. Set $n = n + 1$ and go to Step 1 otherwise.

7. THE M-S SHORTEST PATH PROBLEM

The m-s shortest path problem in a correlated network in this study is formulated as the following concave programming problem (CPP)

$$\min g_3 = \sum_{a\in A} \mu_a \cdot \omega_{a\hat{}a} + f(\eta) \qquad (18)$$

s.t. (8)-(11), where

$$f(\eta) = \gamma_2 \cdot \sqrt{\eta}$$

$$\eta = \sum_{a\in A}\sum_{b\in A} \psi_{ab} \cdot \omega_{a\hat{}b}$$

γ_2 in (18) represents the value of path travel time reliability (measured by standard deviation) relative to the expected travel time. Although the feasible region forms a convex set, the objective function of the problem is strictly concave. Therefore, a local minimum solution of the m-s shortest path problem can be found if a standard algorithm for nonlinear optimization problems is applied to the CPP. Although it is known that each local solution is a corner solution, that is not a global solution which we need to find. Therefore, we will next present a global optimization algorithm for the m-s shortest path problem.

The m-s shortest path problem in this study is the CPP under linear constraints (Tuy, 1964). Global optimization algorithms for the CPP, i.e., the Concavity-Cut Method and the Branch-and-Bound Method, were proposed in Tuy (1964). In this paper, a global optimization algorithm proposed in Soland (1974) is presented. This algorithm was developed by modifying the Falk-Soland Branch-and-Bound Method originally proposed in Falk and Soland (1969).

In this algorithm, a feasible region of M for the value of η shown in (18) is firstly set by determining both a lower bound l and an upper bound u as follows.

$$M: [l, u] \qquad (19)$$

Note that the value of η is calculated subject to (8)-(11). The feasible region denoted by (19) is then bisectionally divided into the following two regions:

$$M_1: [l_1(= l), u_1(= \eta^*)]$$
$$M_2: [l_2(= \eta^*), u_2(= u)],$$

where $\eta^* = \sum_{a\in A}\sum_{b\in A} \psi_{ab} \cdot \omega^*_{a\hat{}b}$ in which $\omega^*_{a\hat{}b}$ is the optimal solution to the relaxation problem defined below.

$$\min \hat{g}_3 = \sum_{a\in A} \mu_a \cdot \omega_{a\hat{}a} + h(\eta) \qquad (20)$$

s.t. (8)-(11), where

$$h(\eta) = \frac{f(u)-f(l)}{u-l} \cdot \eta + \frac{u \cdot f(l) - l \cdot f(u)}{u-l} \qquad (21)$$

The above objective function is linear, and a convex envelope (Falk and Soland, 1969) of the function (18) over the feasible region M. The slope and the intercept of (21) is determined by the feasible temporal region M. By utilizing this property of the relaxation problem, the proposed algorithm for the m-s shortest path problem is described as follows.

Algorithm 2.

Step0: Initialization
$M: [0, \infty]$, $\hat{g}_3' = -\infty$
Record the feasible region M to the top of stack list L
Step1: Termination
If there is no feasible region in the stack list L, then finish the calculation
Step2: Solve the relaxation problem
Pick up the feasible region M from the stack list L
Solve the relaxation problem subject to the feasible region M and obtain the auxiliary solution $\omega^*_{a\hat{}b}\ \forall a, b \in A$
Step3: Solution update
If $\hat{g}_3' < \hat{g}_3$, then $\hat{g}_3' = \hat{g}_3$
Step4: Branching and bound operation

If $l = \eta^*$ or $u = \eta^*$, then go to Step1
Else, branching the feasible region as M_1: $[l, \eta^*]$ and M_2: $[\eta^*, u]$
Record the feasible region M_2 and M_1 to the stack list L
Go to Step1

Note that the stack list is a data structure that fulfills the LIFO (last in, first out) principle. Here, the feasible region M corresponds to an element of the stack list, L. The relaxation problems solved in Step2 have the same problem structure as the m-v shortest path problem addressed in this study. Therefore, they are easily solved by a solution algorithm for the m-v shortest path problem, such as the Simplex method or the algorithm shown in the previous section. In the objective function of (18), since $\sum_{a \in A} \mu_a \cdot \omega_{a \wedge a}$ is in a linear form and $f(\eta)$ is a strictly concave function, the value of the objective function in the relaxation problem is equal to or less than that of the original CPP, i.e.,

$$\hat{g}_3' \leq g_3. \tag{22}$$

It follows the solution of the relaxation problem solved in Step2 is the lower bound of the CPP solution. The algorithm updates the lower bound of the solution of the CPP until a global solution is found. Finally, the algorithm finds the global solution. It is known that the above algorithm finds the global solution by branching the feasible region M as shown in Step 4 if the constraints of CPP (and the relaxation problem) is a convex polygon (Falk and Soland, 1969).

8. NUMERICAL EXPERIMENTS

(1) Small network

A test network shown in Figure 4 (Shahabi et al., 2013) is used for demonstrating the algorithm proposed in this study. This network is comprised of six nodes and nine links. A number on each link in the network is a link number, and that in each node is a node number. The number of unknown variables in this network is 45, and that of equality constraints relating to (8) is 38.

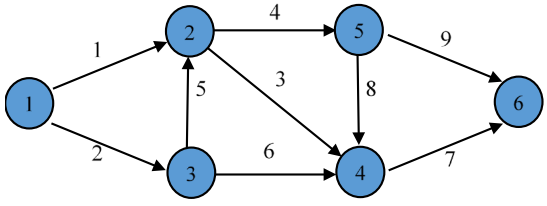

Figure 4 test network

Mean link travel times and variance-covariance matrix of two link travel times are given as

$$\boldsymbol{\mu} = (\mu_1 \quad \cdots \quad \mu_9) = (5 \quad 4 \quad 1 \quad 2 \quad 2 \quad 2 \quad 6 \quad 4 \quad 4)$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} 5 & 1.2 & 0.75 & 0.75 & -0.325 & 0.75 & 0.5 & 0.7625 & 1.5 \\ & 3 & 0.875 & -0.5 & 0.55 & -0.8 & -1.125 & 0.95 & 0.7875 \\ & & 2 & 1 & 1 & 0.8 & -0.3 & 0.3875 & 0.5125 \\ & & & 3 & 1.25 & 0.9 & 0.5 & -0.325 & 1 \\ & & & & 1 & 0.7 & -0.5 & 0.375 & 0.275 \\ & & & & & 3.5 & 0.625 & -0.3625 & -0.1375 \\ & & & & & & 2 & 0.8125 & 1 \\ & & & & & & & 3.5 & -0.6125 \\ & & & & & & & & 4 \end{pmatrix}$$

At first, we solved the m-v shortest path problem for this network. Table 3 shows the minimum m-v travel time costs from node one to the nodes of four and six when γ_1 is set to 0.01 and 0.2. The o-d pairs are set between two nodes where two or more paths are available. The m-v shortest paths from node one to node four when γ_1 equals 0.01 and 0.2 are equally comprised of links two and six, and their travel costs are calculated as 6.05 and 6.98, respectively. The m-v shortest paths from node one to node six when γ_1 equals 0.01 and 0.2 comprise links one, four, and six, and those of two, six, and seven, respectively. The corresponding travel costs are calculated as 11.19 and 13.18, respectively.

Table 3 m-v shortest path from node one to nodes four and six

Origin node	Destination node		Path (link sequence)	g_2
1	4	γ_1=0.01	2, 6	6.05
		γ_1=0.2	2, 6	6.98
	6	γ_1=0.01	1, 4, 9	11.19
		γ_1=0.2	2, 5, 6	13.18

We solved the m-s shortest path problem by applying the algorithm shown in section 6 when γ_2 is set to 0.2 and 0.6. The origin node is one, and the destination nodes are four and six in this problem. Table 4 shows the results of the m-s shortest path problem. Table 5 shows the computation process of the m-s shortest path problem from node one to node four when γ_2 is set to 0.2. An initial feasible region of M is set as $[0, 5000]$. The underlined path travel cost in the table is the minimum path travel cost computed by the algorithm. The m-s shortest path from node one to node four when γ_2 equals 0.2 is comprised of links two and six, and its travel cost is calculated as 6.05. Like the same ways, Tables 6-8 show the computation process under the other conditions shown in Table 3, respectively.

Table 4 The m-s shortest path from node one to nodes four and six

Origin node	Destination node		Path (link sequence)	Path travel cost, g_3
1	4	γ_2=0.2	2, 6	6.45
		γ_2=0.6	2, 6	7.34
	6	γ_2=0.2	1, 4, 9	11.86
		γ_2=0.6	1, 4, 9	13.58

Table 5 Results of the m-s shortest path problem from node one to node four ($\gamma_2 = 0.2$)

Iteration	Feasible region M	Path travel cost $\hat{g}_3$
1	[0, 5000]	6.01
2	[0, 4.90]	6.44
3	[4.90, 4.90]	6.45
4	[4.90, 5000]	6.44

Table 6 Results of the m-s shortest path problem from node one to node four ($\gamma_2 = 0.6$)

Iteration	Feasible region M	Path travel cost $\hat{g}_3$
1	[0, 5000]	6.04
2	[0, 4.90]	7.33
3	[4.90, 4.90]	7.34
4	[4.90, 5000]	7.33

Table 7 Results of the m-s shortest path problem from node one to node six ($\gamma_2 = 0.2$)

Iteration	Feasible region M	Path travel cost $\hat{g}_3$
1	[0, 5000]	11.05
2	[0, 18.50]	11.86
3	[18.50, 5000]	11.86

Table 8 Results of the m-s shortest path problem from node one to node six ($\gamma_2 = 0.6$)

Iteration	Feasible region M	Path travel cost $\hat{g}_3$
1	[0, 5000]	11.16
2	[0, 18.50]	12.82
3	[0, 5.90]	13.46
4	[5.90, 5.90]	13.20
5	[5.90, 18.50]	13.46
6	[18.50, 5000]	13.58

It is confirmed that both the m-v shortest path problems and the m-s shortest path problems for the network are correctly solved and that every solution is given by $\omega_{a\wedge b} = \{0,1\}\ \forall a, b \in A$ without introducing integer constraints. Next, we will solve the two kinds of shortest path problems for a larger network for examining the computability of the algorithm proposed in this study.

(2) Sioux Falls network

The proposed model is applied to the Sioux Falls network (Figure 5). The mean and variance-covariance of the link travel time of the entire network are randomly generated. Tables 9 show the conditions and results for the m-v shortest path problem, i.e., O-D pairs, the value of γ_1, the link sequence as a solution, and the minimum path travel cost, g_2. Table 10 shows those for the m-s shortest path problem. Figure 6 shows the shortest paths of the m-v problem and the m-s problem from node one to node twenty, respectively. Figure 7 shows the shortest paths of the m-v problem and the m-s problem from node four to node eighteen. Tables 11 and 12 show the calculation process of the m-s problem of two different origin-destination pairs, respectively. The underlined path costs show the minimum cost obtained through the iterations.

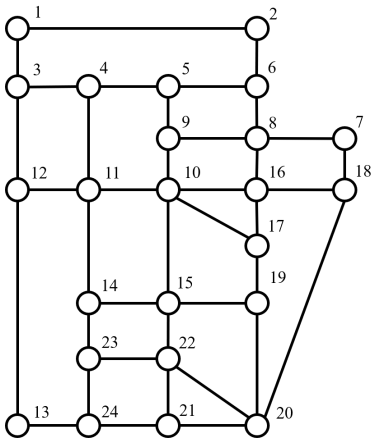

Figure 5 Test network (Sioux Falls)

Table 9 The m-v shortest path problem

Origin node	Destination node		Path (link sequence)	Path travel cost, g_2
1	20	γ_1=0.2	1, 2, 6, 8, 16, 18, 20	6.45
4	18	γ_1=0.2	4, 11, 10, 16, 18	11.86

Table 10 The m-s shortest path problem

Origin node	Destination node		Path (link sequence)	Path travel cost, g_3
1	20	γ_1=0.2	1, 2, 6, 8, 16, 17, 19, 20	6.40
4	18	γ_1=0.2	4, 5, 6, 8, 16, 18	8.81

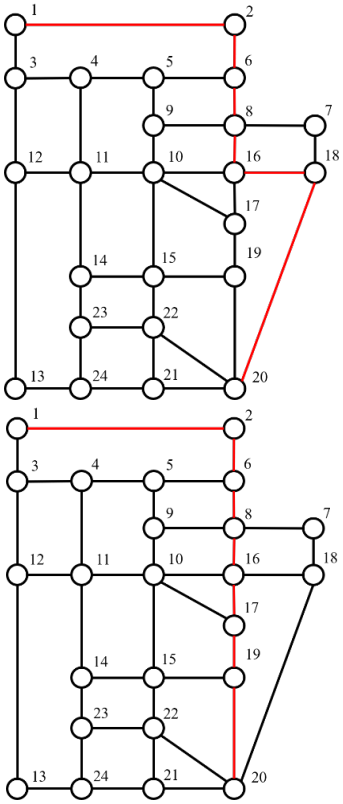

Figure 6 The shortest paths of the m-v problem (top) and m-s problem (bottom) from node 1 to node 20

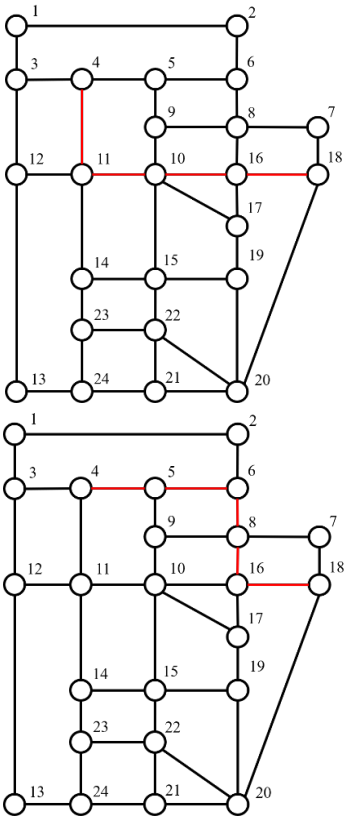

Figure 7 The shortest paths of the m-v problem (top) and m-s problem (bottom) from node 4 to node 18

Table 11 Results of the m-s shortest path problem from node 1 to node 20 ($\gamma_2 = 0.2$)

Iteration	Feasible region M	Path cost $\hat{g}_3$
1	[0, 5000]	3.39
2	[0, 475.09]	6.40
3	[475.09, 475.09]	5.01
4	[475.09, 5000]	6.40

Table 12 Results of the m-s shortest path problem from node 4 to node 18 ($\gamma_2 = 0.2$)

Iteration	Feasible region M	Path cost $\hat{g}_3$
1	[0, 5000]	3.24
2	[0, 293.70]	5.84
3	[293.70, 293.70]	8.81
4	[293.70, 5000]	5.84

We have checked that every solution shown in this section is correctly calculated and that the same solution is obtained when applying the algorithm for the m-v shortest path problem proposed in this study. The m-s shortest path problems were solved within four seconds by applying an algorithm for the linear programming problem. The same tendency was observed for the other o-d pair problems in the network. Once both network settings and the o-d pair are given, both the objective function to be minimized and the corresponding constraints can be mechanically generated by a computer program. Solving the m-v shortest path problem or the relaxation problem for the m-s shortest path problem is easy by applying a standard algorithm for the linear programming problem.

9. CONCLUSIONS

This study proposed a method to find a minimal reliable path cost in a correlated road network. Adopting the iterative labeling algorithm for solving the reliable path search problem in a correlated network is difficult because every link travel time variance/covariance which a driver has experienced must be considered in the problem. Instead of the labeling algorithm, we formulated it as a linear programming-based problem. Depending on the difference in the path choice criteria in a correlated network, two methods for finding the minimal reliable path are proposed. When the path choice criterion of a driver is the sum of mean and variance of path travel time, this study showed that there exists a corresponding linear programming problem to find a minimal reliable path. A conventional algorithm for the linear programming problem solves this shortest path problem, e.g., the simplex method. When the path choice criterion is the sum of mean and standard deviation of path travel time, the minimal reliable path is found by solving

the corresponding concave programming problem. In the proposed algorithm, the relaxation problem that is formulated as a linear programming problem is solved iteratively following a branch-and-bound method. The proposed algorithm finds a global solution of the concave programming problem within finite iterations because of the polygonal constraints of the relaxation problem. The advantage of this method is the easiness of implementation. The conventional algorithm for linear programming can be directly used to solve the proposed method. The proposed methods were verified by the numerical calculations performed in two test networks.

The reader may be interested in solving larger network problems than those addressed in this study. As discussed in the previous section, once both network settings and the o-d pair are given, both the objective function to be minimized, and the corresponding constraints can be mechanically generated by a computer program. Accordingly, the linear programming problems that are needed for solving either the m-v shortest path problem or the m-s shortest path problem can be easily obtained. Therefore, whether the linear programming problems can be solved or not depends on the software that solves the linear programming problems to apply. It should be mentioned that recent such commercial software addresses hundreds of thousands of unknown variables. Still, we may consider the difficulty in solving the linear programming problems in large networks. Therefore, we proposed an algorithm for the m-v shortest path problem, which can also be applied for solving the m-s shortest path problem as well in this study. The number of unknown variables addressed in the algorithm is much smaller than that addressed in the m-v shortest path problem. Thus, it seems that the algorithm proposed in this study efficiently solves a larger problem than those addressed in this study.

Acknowledgments

This study was supported by the JSPS KAKENHI under Grant Number 18H01550 and 20J10083.

Appendix 1

Proof by induction:
From the definition, every column of $\boldsymbol{H}_a$ that is the coefficient matrix of the link$a$-specific hypergraph has at most one 1 and one -1 as its entries. Consider a square submatrix $\boldsymbol{h} \in \{0,1,-1\}^{k\times k}$ of $\boldsymbol{H}_a$. It is obvious that the condition of $det(\boldsymbol{h}) \in \{0,1,-1\}$ holds for $k = 1$. Suppose that the condition holds for every submatrix $\boldsymbol{h} \in \{0,1,-1\}^{k\times k}$ of $\boldsymbol{H}_a$ when $k = n > 1$. Consider then a submatrix $\widehat{\boldsymbol{h}} \in \{0,1,-1\}^{(n+1)\times(n+1)}$ of $\boldsymbol{H}_a$. Each column of $\widehat{\boldsymbol{h}}$ has at most one 1 and one -1 as its entries. If $\widehat{\boldsymbol{h}}$ has a column of zeros, the condition holds since $det(\widehat{\boldsymbol{h}}) = 0$. If every column of $\widehat{\boldsymbol{h}}$ has exactly one 1 and one -1, the condition holds since the row vector of zeros is obtained by summing up all rows in $\widehat{\boldsymbol{h}}$ and thus $det(\widehat{\boldsymbol{h}}) = 0$. The remaining case we have to check is when a column of $\widehat{\boldsymbol{h}}$ has exactly one nonzero entry which is either 1 or -1. Suppose that jth column of $\widehat{\boldsymbol{h}}$ has the nonzero entry at ith row. The condition holds in this case, since

$$det(\widehat{\boldsymbol{h}}) = \pm det\left(\widehat{\boldsymbol{h}}(i,j)\right) \in \{0,1,-1\} \quad (2\text{-}1)$$

(2-1) is obtained by expanding the determinant of $\widehat{\boldsymbol{h}}$ along the jth column, where $\widehat{\boldsymbol{h}}(i,j)$ is the matrix obtained by deleting ith row and jth column from $\widehat{\boldsymbol{h}}$. Note that, $det\left(\widehat{\boldsymbol{h}}(i,j)\right) \in \{0,1,-1\}$ by the induction hypothesis. Therefore, (2-1) holds. The theorem can be obtained by the case where every column of $\widehat{\boldsymbol{h}}$ has exactly one 1 and one -1.

Appendix2

The following theorem is used.

Theorem
If the coefficient matrix of the augmented hypergraph $\boldsymbol{H} \in \{0,1,-1\}^{m\times n}$ is totally unimodular, each collection of columns of $\boldsymbol{H}$ can be split into two parts so that the sum of the columns in one part, minus the sum of the columns in the other part, is a vector with entries 0, +1 and -1 only (Graham et al., 1995).

If a matrix is totally unimodular, the transposed matrix of it is also totally unimodular. Therefore, we will examine the total unimodularity of $\boldsymbol{H}$ by replacing columns with rows in the theorem.

We consider a network as an example that contains the sub-network shown in Figure 8. The corresponding hypergraph is shown in Figure 9. In the hypergraph, two hyperlinks, $a\hat{}b$ and $b\hat{}a$, which are prohibited by (9b) are intentionally depicted. The corresponding node-hyperlink matrix is shown in Table 13. We assume that each row of $\boldsymbol{H}_p$ and the row of $\boldsymbol{J}_{od}$ can belong to the first part and each row of $\boldsymbol{J}$ except for $\boldsymbol{J}_{od}$ can belong to the second part at the initial state. In the following, the sum of the rows in the first part and that in the second part are referred to as the first part row vector and the second part row vector, respectively. When every row of $\boldsymbol{H}$ and $\boldsymbol{J}$ is collected at the initial state, the first part row vector has entries 0, and the second part row vector has entries 0, +1 and -1. Therefore, the condition that *the first*

part row vector minus the second part row vector is a vector with entries 0, +1 and -1 holds. The condition holds in two cases where the first part has some of possible rows and the second part has no row, and where the first part has no row and the second part has some of possible rows at the initial state.

It is known from our formulation that moving the row associated with each node on a link-specific hypergraph from the first part to the second part can have influence only on the values of entries associated with hyperlinks on the link-specific hypergraph in the resultant row vector. It is also known that if the value of an entry in the first part row vector is 1 (or -1) and the corresponding value in the second part row vector is -1 (or 1), the condition does not hold. In the following, we will check whether the condition holds or not after moving some of rows in the first part to the second part when the condition does not hold at the initial state.

Consider the problem in the node-hyperlink matrix for the link*a*-specific hypergraph shown in Table 13. We consider cases where each part has one row at the initial state. In these cases, the condition holds by leaving both parts as they are, or by moving the row in the first part to the second part. We consider then a case where the row associated with node 3_*a* is collected in the first part and two rows associated with nodes *A* and *B* are collected in the second part at the initial state. In this case, the condition does not hold since the values of entries associated with hyperlinks $a\hat{}a$, $a\hat{}b$ and a$\hat{}c$ in the resultant row vector are 1, 0 and -2, respectively. When the row in the first part moves to the second part, the corresponding values are 1, 2 and 0, respectively. This result indicates that when three rows associated with nodes 3_*a*, *A* and *B* are collected, the condition does not hold.

Consider next the problem in the node-hyperlink matrix for the link*b*-specific hypergraph shown in Table 13. We consider cases where each part has one row at the initial state. In these cases, the condition holds by leaving both parts as they are, or by moving the row in the first part to the second part. We consider then a case where the row associated with node 3_*b* is collected in the first part and two rows associated with nodes *A* and *C* are collected in the second part at the initial state. In this case, the condition does not hold since the entries associated with hyperlinks $b\hat{}a$, $b\hat{}b$ and $b\hat{}c$ in the resultant row vector are 2, 1 and -2, respectively. Thus, the row associated with node 3_*b* in the first part moves to the second part. The corresponding values are 0, 1 and 0, respectively. Therefore, the condition holds. If the row associated with node 2_*b* is collected in the first part at the initial state, the condition does not hold since the values of entries associated with hyperlinks $b\hat{}a$, $b\hat{}b$ and $b\hat{}c$ in the resultant row vector are 0, -2 and 0, respectively. In such case, the row associated with node 2_*b* moves to the second part. As a result, the condition seems to hold since the corresponding values are 0, 0 and 0, respectively. In fact, the effects from hyperlinks on link*b*-specific hypergraph that connect to node 2_*b* need to be considered. However, it is shown that if and only if hyperlinks that are prohibited by (9a) or (9b) do not appear on the link*b*-specific hypergraph, the condition holds. This can be easily confirmed by considering a hypergraph corresponding to the sub-network that is comprised of node 2 and some hyperlinks which connect with node 2, and by applying recursively the same discussion as provided above. Also, it is easily confirmed that the same results as obtained in the link*b*-specific hypergraph can be obtained in the case of the link*c*-specific hypergraph.

The results shown above are summarized as follows. The condition holds in both link*b*-specific and link*c*-specific hypergraph problems, whereas the condition does not hold in the link*a*-specific hypergraph problem. Therefore, it is shown that the coefficient matrix of the augmented hypergraph $\boldsymbol{H}$ is not totally unimodular. If (9b) is introduced to the problems we have examined, two hyperlinks $a\hat{}b$ and $b\hat{}a$ disappear from the augmented hypergraph. And thus, the corresponding $\boldsymbol{H}$ becomes totally unimodular. This fact shows that (9a) and (9b) make $\boldsymbol{H}$ totally unimodular, and is true regardless of topology of the subnetwork shown in Figure 8.

The reason why the condition does not hold in $\boldsymbol{H}$ is the existence of the path from node 3_*a* to node 3_*b* that is comprised of two hyperlinks $a\hat{}b$ and $b\hat{}a$. The number of head nodes minus the that of tail nodes of the path is two where head node and tail node represent the origin node(s) and the destination node(s) of a hyperlink, respectively. For example, hyperlink $a\hat{}b$ has one tail node, 3_*a*, and two head nodes, 3_*a* and *A*. If the number of head nodes minus the that of tail nodes of every such path, e.g., paths from node 3_*b* to node 3_*c*, and from node 3_*a* to node 3_*c*, is zero, it is shown that the condition holds. In fact, the constraints, (9a) and (9b), eliminate every path in which the number of head nodes minus the that of tail nodes is not zero from the augmented hypergraph.

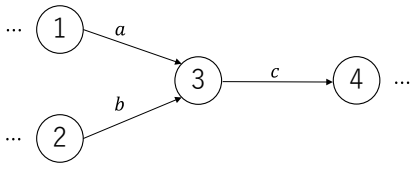

Figure 8 Sub-network

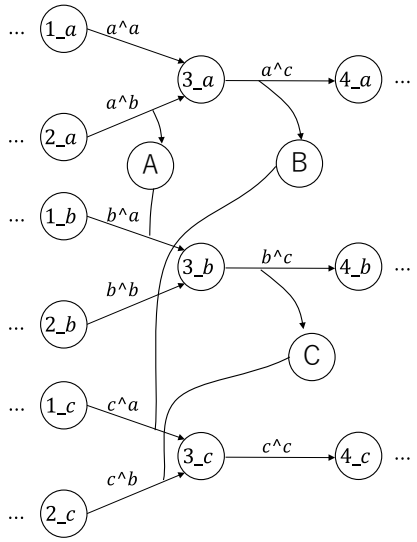

Figure 9 Hyper-graph

Table 13 Node-hyperlink matrix of the hyper-graph

	a^a	a^b	a^c	b^a	b^b	b^c	c^a	c^b	c^c
1_a	-1								
2_a		-1							
3_a	1	1	-1						
4_a			1						
1_b				-1					
2_b					-1				
3_b				1	1	-1			
4_b						1			
1_c							-1		
2_c								-1	
3_c							1	1	-1
4_c									1
A		1		-1					
B			1				-1		
C						1		-1	

Appendix3

It is obvious that $\hat{\omega}_{a\hat{}b} = 0\ \forall b \in A$ if $a \in A_{zero}$ since $\hat{\omega}_{1\hat{}a} = 0$. Suppose that there exists a hyper-link $a\hat{}b$ $(a \neq 1)$ in the link$a$-specific hypergraph of which flow is $\hat{\omega}_{a\hat{}b} = 1$ where $a \in A_{one}$ and $b \in A_{zero}$. Since $b \in A_{zero}$, it is shown that $\hat{\omega}_{1\hat{}b} = 0$. Therefore, we obtain $\hat{\omega}_{b\hat{}c} = 0\ \ \forall c \in A_{zero}$. This contradicts the assumption of $\hat{\omega}_{a\hat{}b} = 1$ where $a \in A_{one}$ and $b \in A_{zero}$. It follows such a hyperlink does not exist. Therefore, if $a \in A_{one}$ then the conditions of $\hat{\omega}_{a\hat{}b} = 0\ \forall b \in A_{zero}$ and $\hat{\omega}_{a\hat{}b} = 1\ \forall b \in A_{one}$ hold.

REFERENCES

1) Boyles, S.D., Waller, S.T., 2010. A mean–variance model for the minimum cost flow problem with stochastic arc costs. Networks 56 (3), 215–227.
2) Chen, B. Y., Shi, C., Zhang, J., H. K. Lam, Li, Q., Xiang, S., 2016. Most reliable pathfinding algorithm for maximizing on-time arrival probability. Transportmetrica B: Transport Dynamics, DOI: 10.1080/21680566.2016.1169953.
3) Chen, A., Zhou, Z., 2010. The α-reliable mean-excess traffic equilibrium model with stochastic travel times, Transportation Research Part B 44(4), 493–513.
4) Coullard, C. R., Ng, P. H., 1995. Totally unimodular Leontief directed hypergraphs, Linear Algebra and its Applications 230, 101 – 125.
5) Falk, J.E., R.M. Soland,. 1969. An algorithm for separable nonconvex programming problems. Management Science 15, 550-569.
6) Fan, Y.Y., Kalaba, R.E., Moore II, J.E., 2005a. Arriving on time. Journal of Optimization Theory and Applications 127 (3), 497–513.
7) Fan, Y.Y., Kalaba, R.E., Moore II, J.E., 2005b. Shortest paths in stochastic networks with correlated link costs. Computers and Mathematics with Applications 49 (9), 1549–1564.
8) Frank, H., 1969. Shortest paths in probabilistic graphs, Operation Research, vol. 17 (4), 583–599.
9) Graham, R., Grötschel, M., and Lovász, L., HANDBOOK OF COMBINATRICS, Edited by 1995 Else-vier Science B.V., (Chapter30, pp1664).
10) Hall, R.W., 1983. Travel outcome and performance: The effect of uncertainty on accessibility, Transportation Research Part B 17(4), 275–290.
11) Hutson, K.R., Shier, D.R., 2009. Extended dominance and a stochastic shortest path problem. Computers & Operations Research 36 (2), 584–596.
12) Khani, A., Boyles, S.D., 2015. An exact algorithm for the mean–standard deviation shortest path problem, Transportation Research Part B vol. 81(P1), 252-266.
13) Lam, W.H.K., Shao, H., Sumalee, A., 2008. Modeling impacts of adverse weather conditions on a road network with uncertainty in demand and supply. Transportation Research Part B 42 (10), 890–910.
14) Lo, H., Tung, Y., 2003. Network with degradable links: Capacity analysis and design, Transportation Research Part B 37 (4), 345–363.
15) Nie, Y., Fan, Y., 2006. Arriving-on-time problem: discrete algorithm that ensures convergence. Transportation Research Record: Journal of the Transportation Research Board 1964 (1), 193–200.
16) Nie, Y.M., Wu, X., 2009. Shortest path problem considering on-time arrival probability. Transportation Research Part B 43 (6), 597–613.
17) Rostami, B., Chassein, A., Hopf, M., Frey, D., Buchheim, C., Malucelli, F., Goerigk, M., 2018. The quadratic shortest path problem: complexity, approximability, and solution methods. European Journal of Operational Research 268 (2), 473–485.
18) Sen, S., Pillai, R., Joshi, S., Rathi, A.K., 2001. A mean–variance model for route guidance in advanced traveler information systems. Transportation Science 35 (1), 37–49.
19) Shahabi, M., Unnikrishnan, A., Boyles, S.D., 2013. An outer approximation algorithm for the robust shortest path problem. Transportation Research Part E: Logistics and Transportation Review 58, 52–66.
20) Shahabi, M., Unnikrishnan, A., Boyles, S.D., 2014. Robust optimization strategy for the shortest path problem under uncertain link travel cost distribution. Computer-Aided

Civil and Infrastructure Engineering.
21) Soland, R.M., 1974. Optimal facility location with concave costs. Operations Research 22. 373-382.
22) Srinivasan, K.K., Prakash, A.A., Seshadri, R., 2014. Finding most reliable paths on networks with correlated and shifted log-normal travel. Times 66, 110–128.
23) Tuy, H., 1964. Concave programming under linear constraints, Soviet Math. 5, 1437-1440.
24) Uchida, K. 2014. Estimating the value of travel time and of travel time reliability in road networks. Transportation Research Part B 66, 129–147.
25) Uchida, K., 2015. Travel time reliability estimation model using observed link flows in a road network, Computer-Aided Civil and Infrastructure Engineering, vol. 30, no. 6, 449–463.
26) Xing, T., and X. Zhou. 2011. Finding the most reliable path with and without link travel time correlation: A Lagrangian substitution based approach. Transportation Research Part B 45 (10), 1660–1679.
27) Zeng, W., Miwa, T., Wakita, Y., Morikawa, T., 2015. Application of lagrangian relaxation approach to α-reliable path finding in stochastic networks with correlated link travel times. Transportation Research Part C 56, 309–334.
28) Zhang, Y., Shen, Z.J.M., Song, S., 2017. Lagrangian relaxation for the reliable shortest path problem with correlated link travel times. Transportation Research Part B 104, 501–521.
29) Zhang, Y., Khani, A., 2019. An algorithm for reliable shortest path problem with travel time correlations, Transportation Research Part B 121, 92–113.
30) Zockaie, A., Nie, Y.M., Wu, X., Mahmassani, H.S., 2013. Impacts of correlations on reliable shortest path finding. Transportation Research Record: Journal of the Transportation Research Board 2334 (1), 1–9.
31) Zockaie, A., Nie, Y.M., Mahmassani, H.S., 2014. Plan B: a simulation-based method for finding minimum travel time budget paths in stochastic networks with correlated link times. In: Transportation Research Board 93rd Annual Meeting (No. 14-5615).

(Received September X, 2022)