International Journal of Reliability, Quality and Safety Engineering

World Scientific
www.worldscientific.com

AN APPROACH FOR THE FAST CALCULATION METHOD OF PARETO SOLUTIONS OF A TWO-OBJECTIVE NETWORK

NATSUMI TAKAHASHI

Department of Management Systems Engineering, Tokyo Metropolitan University, 6-6 Asahigaoka, Hino, Tokyo 191-0065, Japan
takahashi-natsumi@ed.tmu.ac.jp

TOMOAKI AKIBA

Department of Management Information Science, Chiba Institute of Technology, 2-17-1 Tsudanuma, Narashino, Chiba 275-0016, Japan
tomo@akiaki.net

SHUHEI NOMURA and HISASHI YAMAMOTO

Department of Management Systems Engineering, Tokyo Metropolitan University, 6-6 Asahigaoka, Hino, Tokyo 191-0065, Japan
yamamoto@sd.tmu.ac.jp

Received (Day Month Year)
Revised (Day Month Year)

The shortest path problem is a kind of optimization problem and its aim is to find the shortest path connecting two specific nodes in a network, where each edge has its distance. When considering not only the distances between the nodes but also some other information, the problem is formulated as a multi-objective shortest path problem that involves multiple conflicting objective functions. The multi-objective shortest path problem is a kind of optimization problem of multi-objective network. In the general cases, multi-objectives are rarely optimized by a solution. So, to solve the multi-objective shortest path problem leads to obtaining Pareto solutions. An algorithm for this problem has been proposed by using the extended Dijkstra's algorithm. However, this algorithm for obtaining Pareto solutions has many useless searches for paths. In this study, we consider two-objective shortest path problem and propose efficient algorithms for obtaining the Pareto solutions. Our proposed algorithm can reduce more search space than existing algorithms, by solving a single-objective shortest path problem. The results of the numerical experiments suggest that our proposed algorithms reduce the computing time and the memory size for obtaining the Pareto solutions.

Keywords: Two-objective network; Shortest path problem; Pareto solutions; Extended Dijkstra's algorithm.

1. Introduction

The shortest path problem is a kind of optimization problem and its aim is to find the shortest path connecting two specific nodes in a network, where each edge has its distance. The shortest path problem with single objective function is called "single-objective shortest path problem" in this paper. Dijkstra's[1] and the Bellman-Ford algorithms[2,3] are conventional algorithms for solving the single-objective shortest path problem. These algorithms are used to find the shortest path between two specific nodes

in the network. However, these algorithms can solve only the single-objective shortest path problem. When considering not only the distances between the nodes but also some other information, for example, toll, fuel cost, or gradient, the problem is formulated as a multi-objective shortest path problem that involves multiple conflicting objective functions.

The multi-objective shortest path problem is a kind of optimization problem of multi-objective network. The multi-objective network model can be applied to various problems with many conditions, for example, route information system on the map, optimal routing of a network connection to the Internet services, optimally scheduling of a production and distribution systems and multistage-structured modeling for supply chain management systems, etc. Multi-objective network consists of edges (or nodes) with two or more factors. In the general cases, multi-objectives are rarely optimized by a solution. So, to solve the multi-objective shortest path problem leads to obtaining Pareto solutions.

Few algorithms for solving the multi-objective shortest path problem have been proposed. Hara *et al.*[4] proposed a route planning method in car navigation systems by using a multi-objective genetic algorithm (MOGA) for multi-objective shortest path problem in which subjective comfort for driving is maximized. And, Kambayashi *et al.*[5] proposed a MOGA for selecting the shorter route with less number of intersections. The route in their studies is equivalent to the path in our study. Aneja *et al.*[6] defined constrained shortest path problem. They formulated the constrained shortest path problem as a special case of minimal cost flow problem with constraints. Aneja *et al.*[6] proposed the extended Dijkstra's algorithm for a constrained shortest path problem and Miyamoto[7] developed it. Akiba *et al.*[8] applied their extended Dijkstra's algorithm to obtaining the Pareto solutions of two-objective shortest path problem.

In this study, we consider two-objective shortest path problem and propose efficient algorithms for obtaining the Pareto solutions. Though the basic idea is based on Akiba *et al.*[8], our proposed algorithm can reduce more search space than Akiba *et al.*[8], by solving a single-objective shortest path problem. The results of the numerical experiments suggest that our proposed algorithms reduce the computing time and the memory size for obtaining the Pareto solutions.

2. Definition of Problem

2.1. *Network system*

We define V as a set of nodes and E as a set of edges $E = \{e_1, \cdots, e_n\}$, where n is the number of edges and e_j is the j-th edge. Let $G = (V, E)$ be a given network. Node $s\ (\in V)$ means the start node and $t\ (\in V)$ means the terminal node in a network. Furthermore, let *start*(e) and *end*(e) be the start and end node for an edge e, for the convenience. Edge $e\ (\in E)$ has cost vector (c_{1e}, c_{2e}), where element c_{ie} is the cost related to the i-th objective function of edge e for i=1,2, and all c_{ie} are supposed to be

nonnegative values. And, we suppose costs c_{1e} and c_{2e} occur if a target goes along a path from start node to terminal node and the path includes edge *e*. That is, if a target goes along a path with edges e_1, e_2 and e_3, costs $c_{1e_1}+c_{1e_2}+c_{1e_3}$ and $c_{2e_1}+c_{2e_2}+c_{2e_3}$ occur.

2.2. *Shortest path problem of a two-objective network*

For *j*=1,2,…,*n*, let $x_j \in \{0,1\}$ be a binary variable, where $x_j = 1$ if the target goes along edge e_j, and $x_j = 0$ if not. And by using these variables, we define *n*-dimensional vector $\mathbf{x} = (x_1, x_2, \cdots, x_n)$.

First, we consider the following equations.

$$\sum_{\{e_j \in E | start(e_j)=v\}} x_j - \sum_{\{e_j \in E | end(e_j)=v\}} x_j = \begin{cases} 1, & v = s, \\ -1, & v = t, \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

for $v \in V$, $e_j \in E$. Note that *n*-dimensional vector $\mathbf{x}$, satisfying equation (1), can express any paths from start node *s* to terminal node *t*.

Next, we define following function for *i*=1,2.

$$g_i(\mathbf{x}) = \sum_{e_j \in E} c_{ij} x_j. \tag{2}$$

And, let $X \equiv \{\mathbf{x} \mid \mathbf{x} \text{ satisfies equation (1)}\}$, that is, X denotes a set of paths from start node *s* to terminal node *t*. $g_i(\mathbf{x})$ means total cost, when the target goes along a path $\mathbf{x}$.

Next, we define the Pareto solutions considered in this study.

Definition of the Pareto solutions

For $\mathbf{x}, \mathbf{x}' \in X$, path $\mathbf{x}$ becomes a Pareto solution, when there are no other paths $\mathbf{x}'$, which satisfies one of following 3 conditions, among all of *X*.

a) $g_1(\mathbf{x}) > g_1(\mathbf{x}')$ and $g_2(\mathbf{x}) > g_2(\mathbf{x}')$,
b) $g_1(\mathbf{x}) = g_1(\mathbf{x}')$ and $g_2(\mathbf{x}) > g_2(\mathbf{x}')$,
c) $g_1(\mathbf{x}) > g_1(\mathbf{x}')$ and $g_2(\mathbf{x}) = g_2(\mathbf{x}')$.

Therefore, two-objective shortest path problem in this study can be expressed as follows.

Definition of problem

$$g_1(\mathbf{x}) = \sum_{e_j \in E} c_{1j} x_j \rightarrow \min, \tag{3}$$

$$g_2(\mathbf{x}) = \sum_{e_j \in E} c_{2j} x_j \rightarrow \min, \tag{4}$$

s.t.

$$\sum_{\{e_j \in E | start(e_j)=v\}} x_j - \sum_{\{e_j \in E | end(e_j)=v\}} x_j = \begin{cases} 1, & v = s, \\ -1, & v = t, \\ 0, & \text{otherwise}, \end{cases}$$

and

$$x_j \in \{0,1\},\ j=1,2,\ldots,n.$$

3. Existing algorithm

3.1. *Extended Dijkstra's algorithm*

Miyamoto[7] developed the extended Dijkstra's algorithm for solving the constrained shortest path problem based on the Dijkstra's algorithm. Akiba *et al.*[8] presented algorithm for obtaining the Pareto solutions of a two-objective network. Their algorithm is based on Miyamoto's[7] extended Dijkstra's algorithm for a constrained shortest path problem. In this study, we improved Akiba *et al.*'s[8] algorithm to obtain the Pareto solutions of a two-objective network. Before this, we explain the Akiba *et al.*'s[8] algorithm in this section. Now, we define the following notations. For $v \in V$,

$\mathbf{x}_v$: n-dimensional vector, that means a path from start node $s(\in V)$ to a node $v(\in V)$.

W_v : set of adjacent nodes to node v.

And, for $v \in V$ and i=1,2,

l_{iv} : total cost related to i-th objective function when a target go along the path from node s to node v, that is, $l_{iv} \equiv g_i(\mathbf{x}_v)$.

Next, we define "label" (v, l_{1v}, l_{2v}) as the combination of node v and total costs l_{1v} and l_{2v}. And, let L_v be a set of label for node v. Fig. 1 shows the example of network with 2 kinds of costs for all edges and the numerical example to explain labels for all nodes. There are two paths from start node s to node 1. One is the path going along e_1, and from this path, the label for node 1 is generated as (1, 10, 20). The other is the path going along e_2 and e_5, and from this path, the label is calculated as $(1, 20+5, 10+5) = (1, 25, 15)$. Therefore, $L_1 = \{(1,10,20),(1,25,15)\}$. Similarly, $L_2 = \{(2,20,10),(2,25,30)\}$. However label $(2,25,30)$ is dominated by label $(2,20,10)$. Therefore, set L_2 becomes $\{(2,20,10)\}$. In the similar way, $L_t = \{(t,15,25),(t,25,20)\}$. This procedure is based on the extended Dijkstra's algorithm. The extended Dijkstra's algorithm is executed by the following procedure.

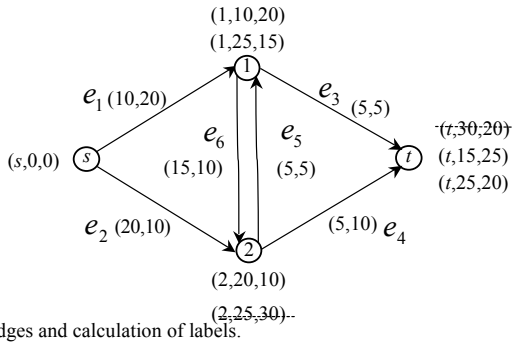

Fig. 1. Costs for all edges and calculation of labels.

Steps in extended Dijkstra's algorithm[8]

STEP 1: (Initialize) Set $L_v \leftarrow \{(s,0,0)\}$, $W_v \leftarrow \phi$ and $v \leftarrow s$.

STEP 2: Obtain the set W_v of adjacent nodes to node *v*.

STEP 3: (Obtain the path to adjacent nodes)

STEP 3-1: Select node $\omega\,(\in W_v)$.

STEP 3-2: Select $(v,l_{1v},l_{2v}) \in \{(\tau,l_{1\tau},l_{2\tau}) \mid ((\tau,l_{1\tau},l_{2\tau}) \in L_v) \cap (\tau = v)\}$

STEP 3-3: Calculate $(\omega, l_{1v}+c_{1e}, l_{2v}+c_{2e})$ for edge *e* when *start*(*e*) is *v* and *end*(*e*) is ω, and set $(\omega,l^*_{1\omega},l^*_{2\omega}) \leftarrow (\omega, l_{1v}+c_{1e}, l_{2v}+c_{2e})$.

STEP 3-4: Compare $(\omega,l^*_{1\omega},l^*_{2\omega})$ with all elements of L_v. If label $(\omega,l^*_{1\omega},l^*_{2\omega})$ is Pareto solution, $(\omega,l^*_{1\omega},l^*_{2\omega})$ is memorized to a set of L_v.

STEP 3-5: Go to STEP 3-6 if all (v,l_{1v},l_{2v}) are selected in STEP 3-2. Go to STEP 3-2 otherwise.

STEP 3-6: Go to **STEP 4** if all nodes in W_v (that is, adjacent nodes to *v*) are selected in STEP 3-1. Go to STEP 3-1 otherwise.

STEP 4: (Obtain the path from known path to adjacent nodes and calculate the labels)

STEP 4-1: Select node $\omega\,(\in W_v)$.

STEP 4-2: Select label $(\sigma,l_{1\sigma},l_{2\sigma}) \in L_v$ for node σ which is one of all nodes connecting with node ω.

STEP 4-3: Calculate $(\omega, l_{1\sigma}+c_{1e}, l_{2\sigma}+c_{2e})$ for edge *e* when *start*(*e*) is σ and *end*(*e*) is ω, and set $(\omega,l^*_{1\omega},l^*_{2\omega}) \leftarrow (\omega, l_{1\sigma}+c_{1e}, l_{2\sigma}+c_{2e})$.

STEP 4-4: Compare $(\omega,l^*_{1\omega},l^*_{2\omega})$ with all elements of L_v. If label $(\omega,l^*_{1\omega},l^*_{2\omega})$ is Pareto solution, $(\omega,l^*_{1\omega},l^*_{2\omega})$ is memorized to a set of L_v.

STEP 4-5: Go to STEP 4-6 if all $(\sigma,l_{1\sigma},l_{2\sigma})$ are selected in STEP 4-2. Go to STEP 4-2 otherwise.

STEP 4-6: Go to **STEP 5** if all nodes in W_v are selected in STEP 4-1. Go to STEP4-1 otherwise.

STEP 5: (Select next node or output Pareto solutions)
STEP 5-1: Go to STEP 5-2 if $v = t$ and all nodes $v(\in V)$ are selected in **STEP 2**. Go to **STEP 2** after selecting $v'(\in W_v)$ and setting $v \leftarrow v'$ otherwise.
STEP 5-2: Output all Pareto solutions $\{(\tau, l_{1\tau}, l_{2\tau}) \mid ((\tau, l_{1\tau}, l_{2\tau}) \in L_v) \cap (\tau = t)\}$ and the algorithm finishes.

Next, we show the searching algorithm for Pareto solutions, which is used at STEP 3-4 and STEP 4-4 in the above algorithm.

Searching procedure for Pareto solutions[8]

STEP 1 : Receive L_v, $(\omega, l^*_{1\omega}, l^*_{2\omega})$.

STEP 2 : Select $(\omega, l'_{1\omega}, l'_{2\omega}) \in \{(\tau, l_{1\tau}, l_{2\tau}) \mid ((\tau, l_{1\tau}, l_{2\tau}) \in L_v) \cap (\tau = \omega)\}$

STEP 3 : Go to **STEP 4** if both of inequalities $l^*_{1\omega} \geq l'_{1\omega}$ and $l^*_{2\omega} \geq l'_{2\omega}$ are satisfied. $L_v \leftarrow L_v \cup \{(\omega, l^*_{1\omega}, l^*_{2\omega})\}$ and go to **STEP 4** otherwise.

STEP 4 : $L_v \leftarrow L_v \setminus \{(\omega, l'_{1\omega}, l'_{2\omega})\}$ if one of following conditions is satisfied.

a) $l^*_{1\omega} < l'_{1\omega}$ and $l^*_{2\omega} < l'_{2\omega}$,

b) $l^*_{1\omega} = l'_{1\omega}$ and $l^*_{2\omega} < l'_{2\omega}$,

c) $l^*_{1\omega} < l'_{1\omega}$ and $l^*_{2\omega} = l'_{2\omega}$.

STEP 5 : Return L_v if all $(\omega, l'_{1\omega}, l'_{2\omega})$ are selected in **STEP 2**. Go to **STEP 2** otherwise.

3.2. *Akiba et al.'s reducing idea for the search space*[8]

The extended Dijkstra's algorithm can obtain the Pareto solutions of a two-objective shortest path problem. However, this algorithm requires much computing time and large memory size for labels when there are many nodes in a network[8]. From this, Akiba *et al.*[8] proposed ideas for reducing the computing time in comparison with the extended Dijkstra's algorithm. The extended Dijkstra's algorithm for obtaining Pareto solutions has many useless searches for paths. The essential point in Akiba *et al.*'s algorithm[8] is to remove such non-useful searches. Now, we define the following notations.

mg_1: minimum value of $g_1(\mathbf{x})$ in all of X, that is, $mg_1 = \min_{\mathbf{x} \in X}\{g_1(\mathbf{x})\}$.

mg_2: minimum value of $g_2(\mathbf{x})$ in all of X, that is, $mg_2 = \min_{\mathbf{x} \in X}\{g_2(\mathbf{x})\}$.

Note the number of path satisfying mg_1 or mg_2 is not necessarily only one. From this, we define the following notations, $X_1 \equiv \{\mathbf{x}' \mid g_1(\mathbf{x}') = \min_{\mathbf{x} \in X}\{g_1(\mathbf{x})\}\}$, $X_2 \equiv \{\mathbf{x}' \mid g_2(\mathbf{x}') = \min_{\mathbf{x} \in X}\{g_2(\mathbf{x})\}\}$, $d_1 \equiv \min_{\mathbf{x}' \in X_2}\{g_1(\mathbf{x}')\}$ and $d_2 \equiv \min_{\mathbf{x}' \in X_1}\{g_2(\mathbf{x}')\}$.

Let path $\mathbf{x}_p$ be Pareto Solution in paths from start node *s* to terminal node *t*. Akiba *et al.*[8] proposed the following theorem.

Theorem(Akiba *et al.*[8]). *A path of Pareto solution* $\mathbf{x}_p(\in X)$ *satisfies both of inequalities* $g_1(\mathbf{x}_p) \le d_1$ *and* $g_2(\mathbf{x}_p) \le d_2$.

Fig. 2 shows the image of reduced search space (that is, the space including all the Pareto solutions) that satisfies inequalities in Theorem[8]. In Fig. 2, horizontal axis and vertical axis mean $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$respectively. All Pareto solutions exist in the rectangle with corners $(0,0)$, $(d_1,0)$, $(0,d_2)$ and (d_1,d_2). Akiba *et al.*[8] discussed the following property for total costs on the path from node *s* to node *v*.

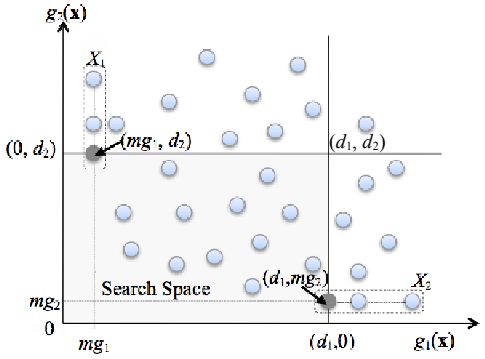

Fig. 2. Image of the reduced search space.

Property(Akiba *et al.*[8]). *If node* $v(\in V)$ *is on the path of Pareto solution* $\mathbf{x}_p$, *vector* (l_{1v}, l_{2v}) *satisfies both of inequalities* $l_{1v} \le d_1$ *and* $l_{2v} \le d_2$.

Based on above the property, Akiba *et al.*[8] proposed the additional process in the extended Dijkstra's algorithm. The process which reduces the search space for the Pareto solutions is shown in the following.

***Additional process of Akiba et al.'s algorithm*[8]**

(Add the step in extended Dijkstra's algorithm)

STEP 0:

(Obtain the shortest path for the first objective function)

STEP 0-1 : Obtain mg_1 by using the Dijkstra's algorithm.

STEP 0-2 : Obtain cost vector (mg_1, d_2) by the value of mg_1 from STEP 0-1.

(Obtain the shortest path for the second objective function)

STEP 0-3 : Obtain mg_2 by using the Dijkstra's algorithm.

STEP 0-4 : Obtain cost vector (d_1, mg_2) by the value of mg_2 from STEP 0-3.

(Replace the STEP 3 at searching procedure for Pareto solutions)

STEP 3: (Search for the Pareto solutions within the reduced search space)

Go to **STEP 4** if one of the following conditions is satisfied.

a) $l^*_{1\omega} \ge l'_{1\omega}$ and $l^*_{2\omega} \ge l'_{2\omega}$,

b) $l^*_{1\omega} > d_1$,

c) $l^*_{2\omega} > d_2$.

Go to **STEP 4** after $L_v \leftarrow L_v \cup \{(\omega, l^*_{1\omega}, l^*_{2\omega})\}$ otherwise.

4. Proposed Algorithm

Akiba *et al.*[8] show their proposed algorithms reduce the computing time and the memory size for obtaining the Pareto solutions. However, more efficient algorithm is required as even Akiba *et al.*'s algorithm[8] needs much computing time and large memory size for a network with more nodes. So, we propose new idea of making the search space for the Pareto solutions reduce more, in this study. First, we state following theorem.

Now, we consider a function $g_1(\mathbf{x}) + wg_2(\mathbf{x})$, where w is a given positive real value.

Theorem. *Let* $\mathbf{x}_s$ *be a path of minimizing* $g_1(\mathbf{x}) + wg_2(\mathbf{x})$ *for* $\mathbf{x} \in X$ *. That is,* $\mathbf{x}_s \equiv \arg\min_{\mathbf{x}\in X}(g_1(\mathbf{x}) + wg_2(\mathbf{x}))$. *Then, path* $\mathbf{x}_s$ *is Pareto solution.*

Proof is shown in Appendix A. Fig. 3 shows the image of solution (path) $\mathbf{x}_s$ in theorem.

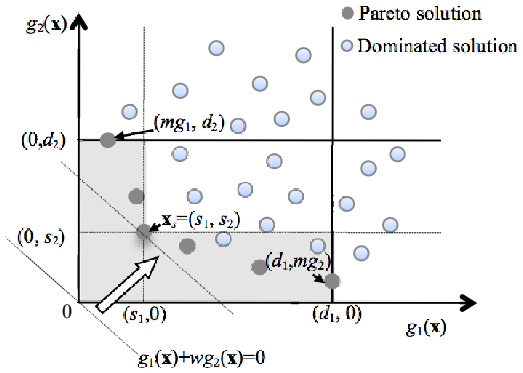

Fig 3. Image of path $\mathbf{x}_s$ and the reduced search space in Theorem and Property.

The essential point in our proposed idea is obtaining method for the path of Pareto solution $\mathbf{x}_s$. As shown in Fig. 3, first, we consider the straight line $g_1(\mathbf{x}) + wg_2(\mathbf{x}) = 0$ and this line moves to upper right direction. When a straight line parallel to $g_1(\mathbf{x}) + wg_2(\mathbf{x}) = 0$ first crosses some path, this path is $\mathbf{x}_s$.

Now, we define notations $s_1 \equiv g_1(\mathbf{x}_s)$ and $s_2 \equiv g_2(\mathbf{x}_s)$ for $\mathbf{x}_s \in X$, that is, we obtain vector (s_1, s_2) of the path $\mathbf{x}_s$. All Pareto solutions exist in the polygon with corners $(0,0)$, $(d_1,0)$, (d_1,s_2), (s_1,s_2), (s_1,d_2) and $(0,d_2)$. Next, we discuss the following property for Pareto solutions.

Property. *A path of Pareto solution* $\mathbf{x}_p (\in X)$ *satisfies one of* $g_1(\mathbf{x}_p) \le s_1$ *and* $g_2(\mathbf{x}_p) \le s_2$.

Property is obvious from the definition of the Pareto solutions. Fig. 3 shows the image of reduced search space that satisfies inequalities in property.

From the above, our proposed algorithm is given by replacing **STEP 0** at Akiba *et al.*'s algorithm[8] and **STEP 3** at searching procedure for Pareto solutions with the following new **STEP 0** and **STEP 3** respectively.

(Replace the STEP 0)

STEP 0 :

(Obtain the shortest path for the first objective function)

STEP 0-1 : Obtain mg_1 by using the Dijkstra's algorithm.

STEP 0-2 : Obtain vector (mg_1, d_2) by the value of mg_1 from STEP 0-1.

(Obtain the shortest path for the second objective function)

STEP 0-3 : Obtain mg_2 by using the Dijkstra's algorithm.

STEP 0-4 : Obtain vector (d_1, mg_2) by the value of mg_2 from STEP 0-3.

(Obtain the shortest path for the $\mathbf{x}_s$)

STEP 0-5 : Obtain $\mathbf{x}_s$ which minimizes $g_1(\mathbf{x}) + wg_2(\mathbf{x})$ by using the Dijkstra's algorithm.

STEP 0-6 : Obtain vector (s_1, s_2) by the path $\mathbf{x}_s = \{\mathbf{x} \mid \mathbf{x} (\in X)$ minimize $g_1(\mathbf{x}) + wg_2(\mathbf{x})\}$ from STEP 0-5.

(Replace the STEP 3 at searching procedure for Pareto solutions)

STEP 3: (Search for the Pareto solutions within the reduced the search space)

Go to **STEP 4** if one of following conditions is satisfied.

a) $l^*_{1\omega} \ge l'_{1\omega}$ and $l^*_{2\omega} \ge l'_{2\omega}$,

b) $l^*_{1\omega} > d_1$,

c) $l^*_{2\omega} > d_2$,

d) $l^*_{1\omega} > s_1$ and $l^*_{2\omega} > s_2$.

Go to **STEP 4** after $L_v \leftarrow L_v \cup \{(\omega, l^*_{1\omega}, l^*_{2\omega})\}$ otherwise.

5. Computational Experience

From Fig. 3, our proposed algorithm makes the search space for the Pareto solutions reduce more clearly than Akiba *et al.* [8]. However, our proposed algorithm needs additional calculation of minimizing $g_1(\mathbf{x}) + wg_2(\mathbf{x})$ by using the Dijkstra's algorithm. That is, our proposed algorithm may not be efficient if it takes much computing time to minimize $g_1(\mathbf{x}) + wg_2(\mathbf{x})$. Therefore, we performed numerical experiments in order to compare our proposed algorithm with the extended Dijkstra's algorithm and Akiba *et al.*'s

idea[8] in terms of actual computing time and the number of labels. The number of labels represents the required memory size for obtaining the Pareto solutions.

Experiments were executed using a PC with Intel Core i5 (3.2 GHz) CPU with 4.0GBytes of RAM, Microsoft Windows 7 professional, Visual Studio 2010 and C programming language. We obtained the Pareto solutions in the cases that the numbers of nodes are 500, 1000, 2000 and 4000. For j=1,2,…,n, we prepared two patterns of costs as follows.

Pattern 1: costs (c_{1j}, c_{2j}) are given values uniformly between 1 and 100.

Pattern 2: c_{1j} are given values uniformly between 1 and 100, and c_{2j} are given values uniformly between 1 and 1000.

Each numerical experiment was executed three times and we show average values of computing times and the numbers of labels in Tables 1 and 2.

Table 1 shows the comparison for computing time and number of labels when the number of nodes is 4000 and costs are prepared by Pattern 1. And, Table 2 shows the comparison for computing time and number of labels when the number of nodes is 4000 and costs are prepared by Pattern 2. In Table 2, we show the comparison of the cases of weight w = 1 and $w = d_1/d_2$. The proportion is the ratio of proposed algorithm to the extended Dijkstra's algorithm in terms of the computing time or the number of labels. And, we calculated variance of computing time for each pattern and compared our proposed algorithm with the existing algorithms, because all costs (these values are gotten by generating uniform random variables) were not the same in each numerical experiment.

The results of the numerical experiments suggest that our proposed algorithm is efficient for obtaining the Pareto solutions. Clearly, computing time becomes shorter and

Table 1. Comparison for computing time and number of labels

	Computing time(sec.)	Variance	Proportion	Number of labels	Proportion
Extended Dijkstra's algorithm[8]	100.82	24.6	-	45.47	-
Akiba *et al*'s idea[8]	46.58	8.10	46.20%	38.57	84.83%
Proposed algorithm	11.12	0.20	11.03%	21.07	46.34%

Table 2. Comparison for computing time and number of labels

	Computing time(sec.)	Variance	Proportion	Number of labels	Proportion
Extended Dijkstra's algorithm[8]	211.15	4.30	-	65.01	-
Akiba *et al*'s idea[8]	102.83	1.80	48.70%	56.50	86.91%
Proposed algorithm (w=1)	28.00	0.00	13.26%	39.45	60.68%
Proposed algorithm ($w=d_1/d_2$)	26.59	0.10	12.59%	38.04	58.51%

the number of labels smaller as the search space becomes smaller. In this numerical experiment, we found that computing time increases exponentially in all algorithms.

However, the results of the numerical experiments suggest that proposed algorithm is more advantageous than the existing algorithms in obtaining the Pareto solutions.

6. Conclusion

In this study, we proposed efficient algorithms for obtaining the Pareto solutions of a two-objective network. The numerical experiments were carried out to compare our proposed algorithms with extended Dijkstra's algorithm and Akiba *et al.*'s idea,[8] in terms of computing time and the number of labels. The results of the numerical experiments suggest that our proposed algorithms reduce the more computing time and the more memory size for obtaining the Pareto solutions than the existing algorithms.

Acknowledgments

This work was partially supported by Grant #25350456, Grant-in-Aid for Scientific Research (c) from JSPS(2013-). The authors thank the JSPS for their support.

Appendix A. Proof of Theorem

If path $\mathbf{x}_s$ is not Pareto solution, there exists path $\mathbf{x}'$ that satisfies one of the following three conditions.

a) $g_1(\mathbf{x}) > g_1(\mathbf{x}')$ and $g_2(\mathbf{x}) > g_2(\mathbf{x}')$,

b) $g_1(\mathbf{x}) = g_1(\mathbf{x}')$ and $g_2(\mathbf{x}) > g_2(\mathbf{x}')$,

c) $g_1(\mathbf{x}) > g_1(\mathbf{x}')$ and $g_2(\mathbf{x}) = g_2(\mathbf{x}')$,

1) Suppose $\mathbf{x}'$ satisfies condition a). As $g_1(\mathbf{x}) > g_1(\mathbf{x}')$ and $wg_2(\mathbf{x}) > wg_2(\mathbf{x}')$ for w, $g_1(\mathbf{x}) + wg_2(\mathbf{x}) > g_1(\mathbf{x}') + wg_2(\mathbf{x}')$. This contradicts to that $\mathbf{x}_s$ minimizes $g_1(\mathbf{x}) + wg_2(\mathbf{x})$.

2) If $\mathbf{x}'$ satisfies b), $g_1(\mathbf{x}) = g_1(\mathbf{x}')$ and $wg_2(\mathbf{x}) > wg_2(\mathbf{x}')$ for w . Then $g_1(\mathbf{x}) + wg_2(\mathbf{x}) > g_1(\mathbf{x}') + wg_2(\mathbf{x}')$ and this contradicts to that $\mathbf{x}_s$ minimizes $g_1(\mathbf{x}) + wg_2(\mathbf{x})$.

3) If $\mathbf{x}'$ satisfies c), $g_1(\mathbf{x}) > g_1(\mathbf{x}')$ and $wg_2(\mathbf{x}) = wg_2(\mathbf{x}')$ for w**.** This is a contradiction.

Hence, **Theorem** holds. □

References

1. E.W. Dijkstra, A note on two problems in connection with graphs, *Numerische Mathematik* **1** (1959) 269-271.
2. R.Bellman, On a Routing Problem, *Quarterly of Applied Mathematics* **16**(1) (1958) 87-90 .
3. L.R.Ford and D.R.Fulkerson, *Flows in Networks* (Princeton University Press, New Jersey, 1962).
4. K. Hara, S. Tsukahara, H. Kanoh, T.Sota and H. Kurokawa, Route Planning for Car Navigation Systems Using Multi-Objective Genetic Algorithm and Predicted Traffic, in *Proc. Information Processing Society of Japan 2006* (2006), pp.31-38 (in Japanese).

5. Y. Kambayashi, H. Yamachi, Y. Tsujimura and H. Yamamoto, Integrating Uncomfortable Intersection-Turns to Subjectively Optimal Route Selection Using Genetic Algorithm, in *Proc. 5th IEEE Int. Conf. on Computational Cybernetics* (2007), pp. 203-208.
6. Y.P. Aneja, V. Aggarwal and K.P.K.Nair, Shortest chain subject to side constrains, *Networks* **13** (1983) 295-302.
7. Y. Miyamoto, Experimental of Analysis for Constrained Shortest Path Problem, in *Proc. Information Processing Society of Japan 2002* (2002), pp. 35-42 (in Japanese).
8. T. Akiba, H. Yamamoto, T. Egashira and S. Iwakami, Fast Algorithm for the Pareto Solutions of a Two-Objective Network, *Journal of the Society of Plant Engineers Japan* **24**(2) (2012) 49-59 (in Japanese).