20

Dissimilar Alternative Path Search Algorithm Using a Candidate Path Set[1]

Yeonjeong Jeong and Dong-Kyu Kim
Expressway & Transportation Research Institute, Seoul National University
South Korea

1. Introduction

There is a growing need for navigation services providing multiple dissimilar alternative paths that reflect a variety of user preferences and a dynamic/stochastic variety of travel time and cost. Providing multiple dissimilar paths, in addition to the shortest path based on a certain attribute (travel cost, time or length), may contribute to extending the market of navigation services and attract new customers since navigation users may prefer other attributes, such as driving convenience and fewer right/left turns, to just saving some minutes of driving time.

The traditional method for searching multiple paths is the K-shortest path search method, which provides paths in the order of cost (Yen, 1971; Shier, 1976; 1979; Martins, 1984; Azevedo et al., 1993). However, the multiple paths found by this method tend to be very similar to each other and cannot truly be called alternative paths. A reasonable alternative path should not only have acceptable attribute value(s) but should also be dissimilar to previously identified paths in terms of the links used (Park et al., 2002).

To avoid unreasonable searches for the multiple paths, some methods were suggested in the previous literature. Barra et al. (1993) suggested a link penalty method. In this method, links of a path which was searched in the previous step have a penalty and the next path is searched using a shortest path search algorithm with consideration of the penalties of links. The link penalty method has no way of evaluating the quality of the set of the paths it generates in terms of the spatial differences and the costs of the paths while it can be easily used due to its simplicity (Akgün et al., 2000). Furthermore, this method cannot consider the cost constraint of the alternative paths while generalized cost is one of the most important to select alternative paths in navigation services. The link penalty method is, therefore, not suitable for the path search algorithm for navigation services.

Meng et al. (2005) and Martí et al. (2009) formulated the multiple dissimilar path problems as a multi-objective problem and suggested several solution algorithms. These problems have multiple objectives varying with the purpose of researches such as minimizing average or total costs of the paths, minimizing total population exposed and maximizing the average of the dissimilarity between the paths. However, it takes significant time to solve the multi-objective problem while these problems can produce the paths that meet the given

[1] A preliminary version of this paper has appeared in Jeong (2010) writen in Korean.

conditions. The multi-objective problem is not suitable for the application of navigation services since search time also is important as much as dissimilarity in navigation services.
On the other hand, Park et al. (2002) and Jeong et al. (2010) formulated the multiple dissimilar path problems as using cost (time) and dissimilarity constraints. Each of methods has one objective function, such as minimizing the cost of alternative path (Park et al., 2002) or minimizing the shared length ratio between an alternative path and the previous searched path (Jeong et al., 2010). The method suggested in Park et al. (2002) is based on the efficient vector labeling approach (EVL) wherein a cost constraint is used for pruning a transportation network and an overlap constraint is applied to maintain the dissimilarity of alternative paths. The method developed by Jeong et al. (2010) provides dissimilar alternative paths satisfying both user-specified allowable shared length ratio with already searched paths and travel cost ratio against the shortest path. These methods can find alternative dissimilar paths in rational time but they cannot sometimes provide the number of alternative paths required due to their strict constraints.
The candidate path set (CPS) method is another one to use the candidate path set to provide the number of paths required considering dissimilarity. The method suggested in Jeong et al. (2007) establishes a candidate path set based on path deviation by executing the shortest-path algorithm only once. The CPS method provides candidate paths based on the previously searched paths, and thus can provide more various paths than the classical k-shortest path search algorithm. The merit of the CPS is that dissimilarity is not used as a strict constraint but a criterion for selecting alternative paths. In the next section, we will explain the CPS algorithm based on Jeong et al. (2007) with a travel cost constraint and a dissimilarity selecting criterion.

2. Proposed algorithm

2.1 Algorithm

Step 1 Find shortest paths from each node to a destination

Find all-to-one shortest paths from each node to a destination node s.

Denote the shortest path from some node h to a destination node s by P_1^{hs}.

For our interesting origin node r and destination node s,

$$A^{rs} = A^{rs} \cup \{P_1^{rs}\}$$

$$UB = \alpha \times C[P_1^{rs}]$$

Step 2 Update candidate path set

When $P_k^{rs} = \{r, 1, 2, \ldots, (j-1), j, s\}$

step 2-1

$$R_k^{rj} = P_k^{rs} - \{s\},\ C[R_k^{rj}] = C\left[P_k^{rs}\right] - c^{js},\ L[R_k^{rj}] = L\left[P_k^{rs}\right] - \ell^{js}$$

If $R_k^{rj} \in C^{rj}$

then go to step 3

otherwise

$$P_c^{rs} = R_k^{rj} + P_1^{hs}$$

$$C[P_c^{rs}] = C[R_k^{rj}] + c^{jh} + C[P_1^{hs}]$$

$$L[P_c^{rs}] = L[R_k^{rj}] + \ell^{jh} + L[P_1^{hs}]$$

$$(\text{where, } h \in O^j,\ h \neq r, 1, 2, \ldots, (j-1), j, s)$$

$$C^{rj} = C^{rj} \cup \{R_k^{rj}\}$$

If $C[P_c^{rs}] \leq UB$

then $B^{rs} = B^{rs} \cup \{P_c^{rs}\}$

step 2-2

$j = j - 1$

$R_k^{rj} = R_k^{r(j+1)} - \{j+1\}$, $C[R_k^{rj}] = C[R_k^{r(j+1)}] - c^{j(j+1)}$, $L[R_k^{rj}] = L[P_k^{r(j+1)}] - \ell^{j(j+1)}$

If $R_k^{rj} \in C^{rj}$

then go to step 3

otherwise

$$P_c^{rs} = R_k^{rj} + P_1^{hs}$$

$$C[P_c^{rs}] = C[R_k^{rj}] + c^{jh} + C[P_1^{hs}]$$

$$L[P_c^{rs}] = L[R_k^{rj}] + \ell^{jh} + L[P_1^{hs}]$$

$$(\text{where, } h \in O^j,\ h \neq r, 1, 2, \ldots, j, (j+1))$$

$$C^{rj} = C^{rj} \cup \{R_k^{rj}\}$$

If $C[P_c^{rs}] \leq UB$

then $B^{rs} = B^{rs} \cup \{P_c^{rs}\}$

If $j = r$

then go to step 3

otherwise go to step 2-2

Step 3 select $k+1^{th}$ alternative path

$$P_{k+1}^{rs} = \underset{P_c^{rs} \in B^{rs}}{\operatorname{argmin}} \frac{1}{K} \sum_{k=1}^{K} \frac{L\left[P_k^{rs} \cap P_c^{rs}\right]}{L\left[P_k^{rs}\right]} \quad \left(P_k^{rs} \in A^{rs}\right)$$

If user accepts that path

then the algorithm ends

otherwise go to step 2

where,

k	: Number of alternative paths
h, i, j	: Node Index (node r is an origin and node s is a destination)
α	: User-specified allowable travel cost ratio
UB	: Travel cost constraint (upper bound of alternative path cost)
O^j	: Set of nodes linked from node j
$< i, j >$	: Link between node i and node j

c^{ij} : Cost of link between node i and node j

ℓ^{ij} : Length of link between node i and node j

A^{rs} : Set of alternative paths from node r to node s

B^{rs} : Set of candidate paths from node r to node s

C^{rj} : Set of subpaths for avoiding of making same candidate paths

P_k^{rs} : k^{th} Alternative path from node r to node s $(P_k^{rs} \in A^{rs})$, which is a set of continuous nodes $P_k^{rs} = \{r,1,2,\ldots,(j-1),j,s\}$

P_c^{rs} : Candidate path from node r to node s $(P_c^{rs} \in B^{rs})$, which is a set of continuous nodes $P_c^{rs} = \{r,1,2,\ldots,(j-1),j,s\}$

R_k^{rj} : Subpath of P_{k-1}^{rs} from node r to node j, which is a set of continuous nodes $R_k^{rj} = \{r,1,2,\ldots,(j-1),j\}$

$R_k^{rj} + P_1^{hs}$: Path adding R_k^{rj} and P_1^{hs} with link $< j,h >$

$R_k^{rj} - \{j\}$: Path excluding node j from R_k^{rj}

$C[P_k^{rs}]$: Cost of path P_k^{rs}

$L[P_k^{rs}]$: Length of path P_k^{rs}

$L[P_k^{rs} \cap P_c^{rs}]$: Length of overlapped link between P_k^{rs} and P_c^{rs}

2.2 Illustrative example

In this subsection, we explain how the CPS algorithm works using an example network of Fig. 1, which consists of 9 nodes and 12 undirected links. In Fig. 1, the alphabetical characters within each circle mean node indices; the numbers in front of the parentheses denote link indices; two numbers in the parentheses are cost and length of each link, respectively. In this example, the user-specified allowable travel cost ratio, α, is set to 1.3, which means that we include paths with the travel cost not higher than UB by 30% into the candidate path set .

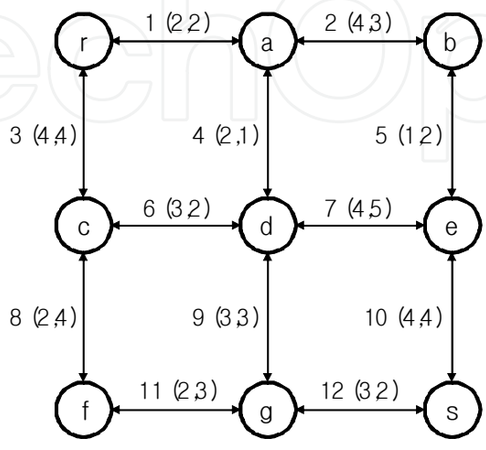


Fig. 1. Example network

2.2.1 Finding the 1st alternative path (the shortest path)

First of all, we found P_1^{hs} from all nodes to a destination node s using a reverse shortest path search algorithm as shown in Fig. 2. Table 1 shows the cost and length of all P_1^{hs}. After then, the shortest path, $P_1^{rs} = \{r,a,d,g,s\}$, is found as shown in Fig. 3. At this time, the minimum travel cost becomes '10', and travel cost constraint, UB is set to '13' as shown in Table 2.

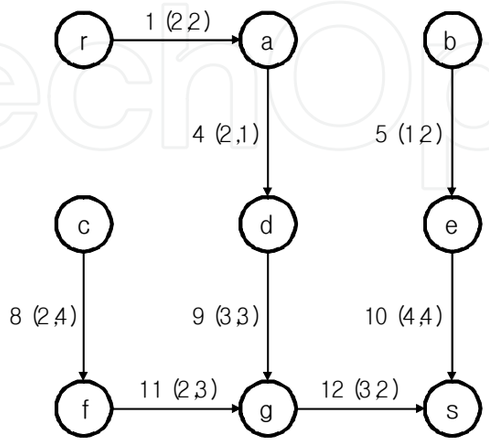


Fig. 2. Shortest paths to node s

h	r	a	b	c	d	e	f	g
$C[P_1^{hs}]$	10	8	5	7	6	4	5	3
$L[P_1^{hs}]$	8	6	6	9	5	4	5	2

Table 1. Path costs and lengths of shortest paths

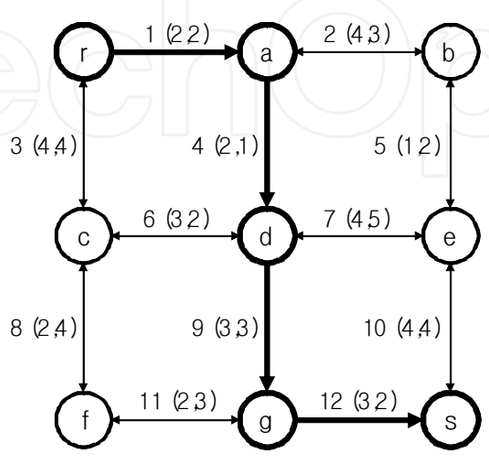


Fig. 3. The 1st shortest path

k	R_k^{rj}	$C[R_k^{rj}]$	h	P_c^{rs}	$C[P_c^{rs}]$	O.R.	P_k^{rs}	$C[P_k^{rs}]$
1							r,a,d,g,s	10

Table 2. Result of the 1st shortest path

2.2.2 Finding the 2nd alternative path

First, we make a subpath, $R_2^{rg} = \{r,a,d,g\}$, by deleting link 12 between node g and node s. The cost of this subpath is calculated as '7' subtracting the cost '3' of deleted link 12 from the cost '10' of $C[P_1^{rs}]$. Then, this subpath is stored in C^{rg}. Node f linked from node g by link 11 is considered as a candidate node for the next path search.

The candidate path set is made as follows: The shortest path from node f to a destination node s has been already determined as P_1^{fs} in the step 1. Therefore, the path adding R_2^{rg} and P_1^{fs}, $R_2^{rg} + P_1^{fs} = \{r,a,d,g,f,g,s\}$, may be the candidate path for the next alternative path. The cost of this path is calculated as '14' by adding the cost '2' of c^{gf} and the cost '5' of $C[P_1^{fs}]$ to the cost '7' of $C[R_2^{rg}]$. This path cost '14' is higher than the user-specified allowable cost ($\alpha \times UB = 13$), and thus the path, $R_2^{rg} + P_1^{fs}$, cannot be included into the candidate path set.

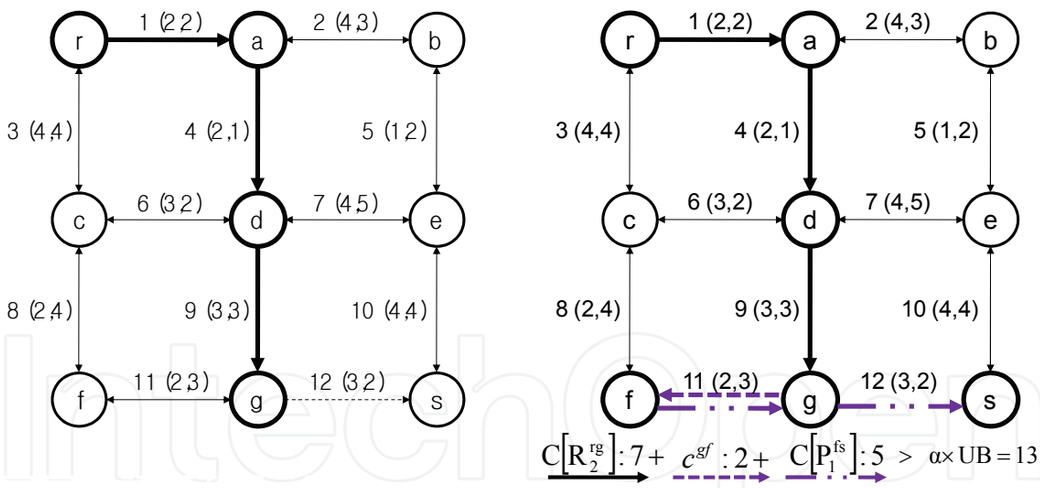


Fig. 4. Updating candidate path set by using subpath R_2^{rg}

Next, a subpath, $R_2^{rd} = \{r,a,d\}$ is investigated by deleting link 9 between node d and node g. The cost of this subpath is calculated as '4' subtracting the cost '3' of deleted link 9 from the cost '7' of $C[R_2^{rg}]$. Then, this subpath is stored in C^{rd}. Node c linked by link 6 and node e linked by link 7 from node d are considered as the next candidate nodes for the next path search.

In the same way mentioned above, two candidate paths, $R_2^{rd} + P_1^{cs} = \{r,a,d,c,f,g,s\}$ and $R_2^{rd} + P_1^{es} = \{r,a,d,e,s\}$, are made for the next alternative path since the shortest paths from

node c and node e to a destination node s, as P_1^{cs} and P_1^{es} have been already determined in the step 1. The cost of the path, $R_2^{rd} + P_1^{cs}$, is calculated as '14' adding the cost '3' of c^{dc} and the cost '7' of $C[P_1^{cs}]$ to the cost '4' of $C[R_2^{rd}]$ and this path cannot be also included into the candidate path set since its cost is higher than the user-specified allowable cost. On the other hand, the cost of the path, $R_2^{rd} + P_1^{es}$ is calculated as '12' adding the cost '4' of c^{de} and the cost '4' of $C[P_1^{es}]$ to the cost '4' of $C[R_2^{rd}]$. We can include this path, $R_2^{rd} + P_1^{es}$, into the candidate path set since its cost is not higher than the user-specified allowable cost.

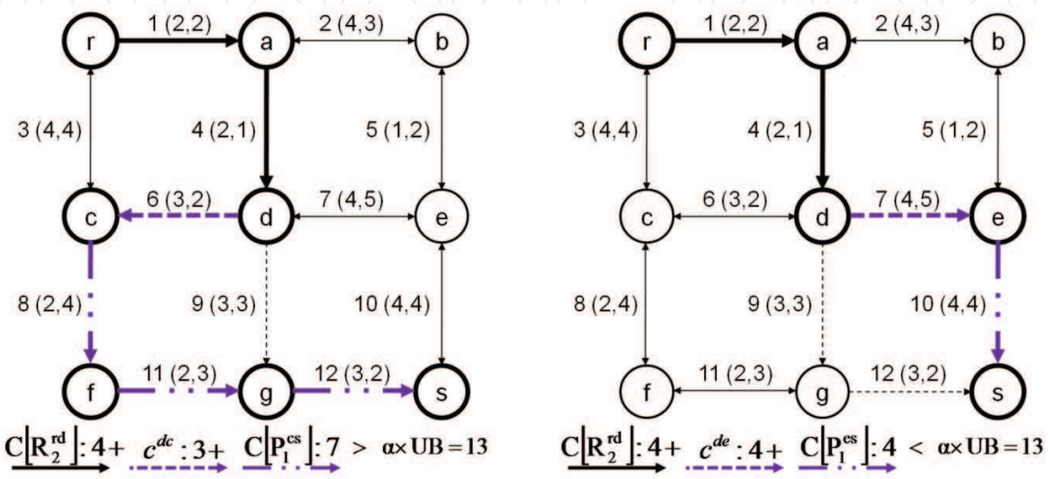


Fig. 5. Updating candidate path set by using subpath R_2^{rd}

Next, a subpath, $R_2^{ra} = \{r, a\}$, is made by deleting link 4 between node a and node d. After calculating the cost of this subpath as '2', this subpath is stored in C^{ra} and node b linked from node a by link 2 is considered as the next candidate. The cost of the path, $R_2^{ra} + P_1^{bs} = \{r, a, b, e, s\}$, can be calculated as '11' adding the cost '4' of c^{ab} and the cost '5' of $C[P_1^{bs}]$ to the cost '2' of $C[R_2^{ra}]$. This path, $R_2^{ra} + P_1^{bs}$, can be also included into the candidate path set since its cost is not higher than the user-specified allowable cost.

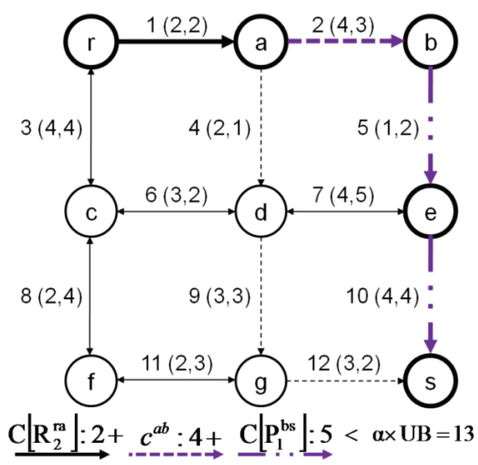


Fig. 6. Updating candidate path set by using subpath R_2^{ra}

Finally, we make a subpath, $R_2^{rr} = \{r\}$, by deleting link 1 between node r and node a. The cost of this subpath is calculated as '0'. This subpath is stored in C^{rr} and node c linked from node r by link 3 will be the next candidate node for the next path search. The path $R_2^{rr} + P_1^{cs} = \{r,c,f,g,s\}$ may be the candidate path for the next alternative path since the shortest path from node c to a destination node s has been already determined as P_1^{cs} in the step 1. The cost of this candidate path is calculated as '11' adding the cost '4' of linked c^{rc} and the cost '7' of $C[P_1^{cs}]$ to the cost '0' of $C[R_2^{rr}]$. And we can also include this path, $R_2^{rr} + P_1^{cs}$, into the candidate path set since its cost is not higher than the user-specified allowable cost.

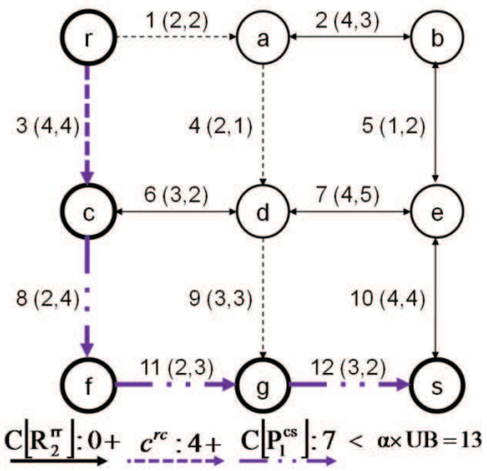


Fig. 7. Updating candidate path set by using subpath R_2^{rr}

Because we have made all subpaths to an origin node r using the shortest path P_1^{rs}, we can select the second alternative path, P_2^{rs} in the step 3. First, we calculate the overlap length ratios of all the paths in the candidate path set by dividing overlapped length of candidate path and the shortest path by the length '8' of the shortest path. The overlapped length ratio of the path, $\{r,a,b,e,s\}$, is the smallest one among the candidate paths, so this path becomes the second alternative path and is deleted in the candidate path set.

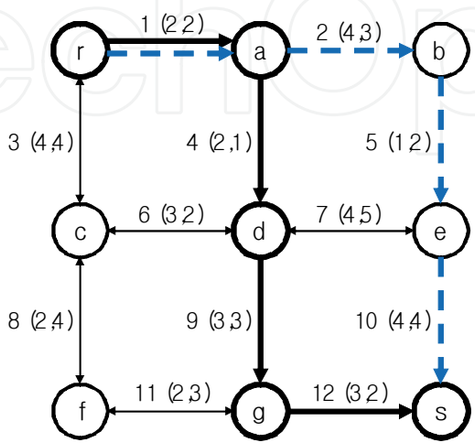


Fig. 8. 1st path and 2nd path

k	R_k^{rj}	$C[R_k^{rj}]$	h	P_c^{rs}	$C[P_c^{rs}]$	O.R.[2]	P_k^{rs}	$C[P_k^{rs}]$
1							r,a,d,g,s	10
2	r,a,d,g	10-3	f	r,a,d,g,f,g,s	7+2+5=14[3]		r,a,b,e,s	11
	r,a,d	7-3	c	r,a,d,c,f,g,s	4+3+7=14			
			e	r,a,d,e,s	4+4+4=12	3 / 8		
	r,a	4-2	b	r,a,b,e,s	2+4+5=11	2/8		
	r	2-2	c	r,c,f,g,s	0+4+7=11	2 / 8		

Table 3. Result of finding the 2nd alternative path

2.2.3 Finding the 3rd alternative path

Next, we will find the third alternative path P_3^{rs} from an origin node r to a destination node s using the second alternative path P_2^{rs}.

First, in the same way mentioned above, we make a subpath, $R_3^{re} = \{r,a,b,e\}$, by deleting link 10 between node e and node s. The cost of this subpath is calculated as '7' subtracting the cost '4' of deleted link 10 from the cost '11' of $C[P_2^{rs}]$. Then, this subpath is stored in C^{re}. Node d linked from node e by link 7 is considered as the first candidate node for the third shortest path search. The cost of the path, $R_3^{re} + P_1^{ds} = \{r,a,b,e,d,g,s\}$, is calculated as '17' adding the cost '4' of c^{ed}, the cost '6' of $C[P_1^{ds}]$ and the cost '7' of $C[R_3^{re}]$. This path cost '17' is higher than the user-specified allowable cost ($\alpha \times UB = 13$), and thus the path, $R_3^{re} + P_1^{ds}$, cannot be included into the candidate path set.

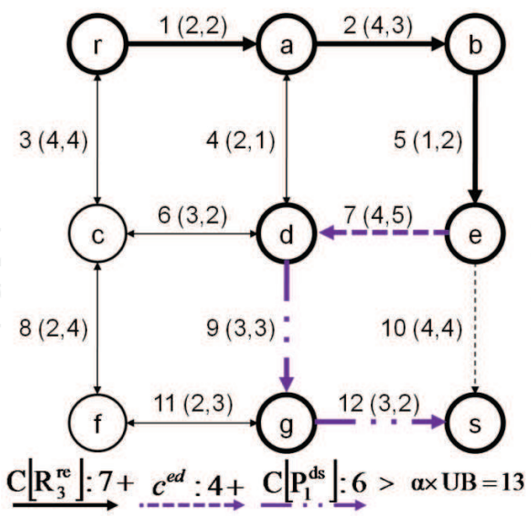


Fig. 9. Updating candidate path set by using subpath R_3^{re}

[2] The overlapped length ratio

[3] The search stops because the path cost is higher than the user-specified allowable cost.

Next, a subpath, $R_3^{rb} = \{r, a, b\}$, is made by deleting link 5 between node b and node e. The cost of this subpath is calculated as '6' subtracting the cost '1' of deleted link 5 from the cost '7' of $C[R_3^{re}]$. Then, this subpath is stored in C^{rb}. Because there is no node linked from node b, the search stops.

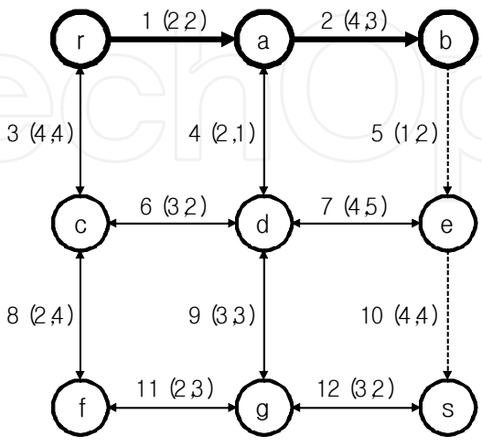


Fig. 10. Updating candidate path set by using subpath R_3^{rb}

Next, we make and investigate a subpath, $R_3^{ra} = \{r, a\}$, by deleting link 2 between node a and node b. The cost of this subpath is calculated as '2' subtracting the cost '4' of deleted link 2 from the cost '6' of $C[R_3^{rb}]$. This subpath, however, has been already included in C^{ra} during the second alternative path search process. Therefore, the search stops and we can select the third alternative path, P_3^{rs} in the step 3.

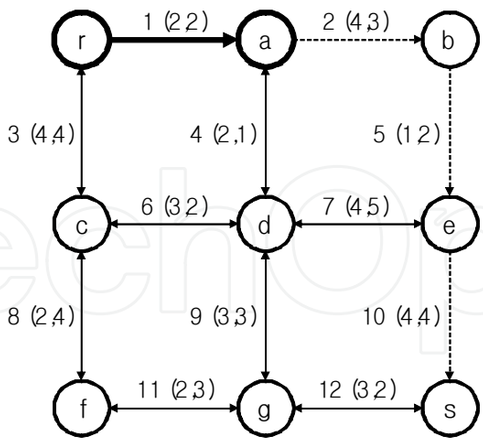


Fig. 11. Updating candidate path set by using subpath R_3^{ra}

First, we calculate the average overlapped length ratios between each path in the candidate path set and the alternative paths. The average overlapped length ratio of the path, $\{r, c, f, g, s\}$ is the smallest one among the candidate paths, so this path becomes the third alternative path and is deleted in the candidate path set.

k	R_k^{rj}	$C[R_k^{rj}]$	h	P_c^{rs}	$C[P_c^{rs}]$	O.R.	P_k^{rs}	$C[P_k^{rs}]$
1							r,a,d,g,s	10
2	r,a,d,g	10-3	f	r,a,d,g,f,g,s	7+2+5=14		r,a,b,e,s	11
	r,a,d	7-3	c	r,a,d,c,f,g,s	4+3+7=14			
			e	r,a,d,e,s	4+4+4=12	$\frac{1}{2}\times(\frac{3}{8}+\frac{6}{12})$		
	r,a	~~4-2~~	~~b~~	~~r, a, b, e, s~~	~~2+4+5=11~~	~~2/8~~		
	r	2-2	c	r,c,f,g,s	0+4+7=11	$\frac{1}{2}\times(\frac{2}{8}+0)$		
3	r,a,b,e	11-4	d	r,a,b,e,d,g,s	7+4+6=17		r,c,f,g,s	11
	r,a,b [4]	7-1						
	r,a [5]	6-4						

Table 4. Result of finding the 3rd alternative path

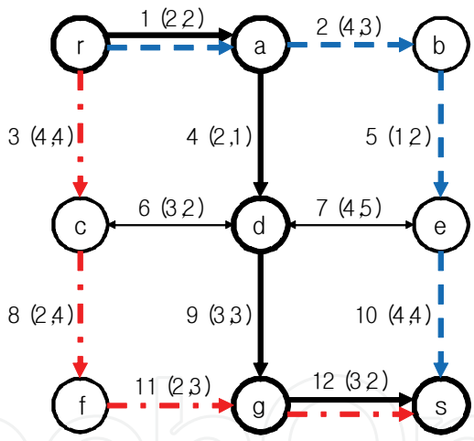


Fig. 12. 1st path, 2nd path and 3rd path

3. Computational results

To check the performance, all methods were applied to a real Chicago and Philadelphia networks[6]. The Chicago network consists of 12,982 nodes and 39,018 links. And the

[4] This search stops since there is no node linked from node b.

[5] This search stops for avoiding of making same candidate paths since this subpath has been already included in the set of subpaths

[6] We used 'chicago_network.txt' and 'philadelphia_network.txt' from http://www.bgu.ac.il/~bargera/tntp/.

Philadelphia network consists of 13,389 nodes and 40,003 links. We used 'cost' columns of both network data as link costs but substituted the 'ftime' value '0.01' for '0'. We used C++ for programming and implemented the solution algorithm using a computer with Pentium Core2 Quad 2.66 GHz processor, a Windows XP operating system, and 2 GB DDR RAM.
First of all, we tested 100 iterations of the algorithm to make candidate path sets from node 25 to node 10348 based on the Philadelphia network. Table 5 shows the number of candidate path and CPU time to the iteration. This test shows the method can provide candidate paths quickly.

iteration	# of Candidate Paths	CPU time	iteration	# of Candidate Paths	CPU time
0	1	0.141	10	274	0.234
1	48	0.141	20	450	0.359
2	91	0.156	30	629	0.547
3	91	0.156	40	847	0.812
4	137	0.172	50	971	1.016
5	137	0.172	60	1,136	1.312
6	175	0.187	70	1,278	1.609
7	218	0.203	80	1,463	2.031
8	256	0.219	90	1,544	2.297
9	274	0.234	100	1,669	2.703

Table 5. Test result of making candidate path sets

We compared the results of the EVL method and the CPS method to evaluate the suitability of these methods for navigation services that finds multiple alternative paths. We selected randomly 1,000 O-D (origin-destination) pairs on the Chicago network and 100 O-D pairs on the Philadelphia network for searching for the shortest path and 9 alternative dissimilar paths. This test determines whether the method can provide as many alternative paths as users want; the larger the number of the paths, the better the method.
In Tables 6 and 7, the figures in parentheses refer to the specified values of constraints. For example, 1.1 and 2.0 of the 'user allowable travel cost ratio' mean that the paths having travel cost higher than the shortest path by 10% and 100%, respectively, should not be selected for the next alternative path. In the same manner, 0.5 and 0.8 of the 'user allowable overlapped length ratio' mean that the paths having 50% and 80% or more overlapped length ratios with already searched paths, respectively, should not be selected. The value of

the overlapped length ratio constraint of the CPS method is 1.0 since it does not employ the constraint.

Table 6 shows the average numbers of paths searched by each method. The results show that the stronger the cost and overlap constraints, the fewer paths were searched. The CPS method, which uses only the cost constraint, searched alternative paths close to the number of paths to be required (K=9). Because CPS does not consider the overlap constraint, it can provide almost as many alternative paths as users may want.

		User allowable travel cost ratio							
		Chicago network				Philadelphia network			
		Strong (1.1)		Weak (2.0)		Strong (1.1)		Weak (2.0)	
User allowable overlapped length ratio	Strong (0.5)	EVL	2.56	EVL	7.75	EVL	1.74	EVL	7.49
	Weak (0.8)	EVL	4.47	EVL	7.68	EVL	3.61	EVL	7.41
	None (1.0)	CPS	8.80	CPS	8.97	CPS	8.83	CPS	8.91

Table 6. Average numbers of paths searched (K=9)

Table 7 shows the average CPU computation time required for each method. This test determines which method can provide alternative paths more quickly, so the smaller the value, the better the method. The EVL method prunes the network using the cost constraint. Therefore, the average CPU computation time of the EVL method increases rapidly as the cost constraint weakens while its computation time was not significantly influenced by the overlap constraint. The CPS method searched the number of paths required more quickly because it did not consider the overlap constraint.

		User allowable travel cost ratio							
		Chicago network				Philadelphia network			
		Strong (1.1)		Weak (2.0)		Strong (1.1)		Weak (2.0)	
User allowable overlapped length ratio	Strong (0.5)	EVL	0.86	EVL	4.22	EVL	1.31	EVL	6.05
	Weak (0.8)	EVL	0.95	EVL	4.12	EVL	1.55	EVL	6.08
	None (1.0)	CPS	0.50	CPS	0.56	CPS	0.51	CPS	0.62

Table 7. Average CPU computation times (sec)

Tables 8 and 9 show the average overlap ratios for the case when one or more alternative paths are searched and for the overall sample data (1,000 and 100 O-D pairs), respectively. As shown in Table 8, when one or more alternative paths were searched, CPS's average overlapped length ratio is higher than that of EVL which has the overlap constraint.
However, it is too early to conclude that EVL has better performances than CPS since the result of Table 8 did not reflect the cases when no alternative path was searched. When there is no alternative path, a driver is provided only one path. Therefore, in the result of Table 8, the overlapped length ratio was set to 1.0 when there was no alternative path.
As shown in Table 9, the result of CPS is similar to that of EVL if the result of no alternative path is included. In other words, considering only the case when one or more alternative paths were searched, the EVL with the overlapped length ratio constraint searched more dissimilar paths than CPS; however, in general conditions including the case when no alternative path was searched, whether a method considered the overlap constraint or not does not influence the results. Summarizing all the results, then, the CPS method finds more number of dissimilar alternative paths with similar overlap ratios to the EVL more quickly.

		User allowable travel cost ratio							
		Chicago network				Philadelphia network			
		Strong (1.1)		Weak (2.0)		Strong (1.1)		Weak (2.0)	
User allowable overlapped length ratio	Strong (0.5)	EVL	0.30	EVL	0.27	EVL	0.33	EVL	0.27
	Weak (0.8)	EVL	0.57	EVL	0.52	EVL	0.58	EVL	0.52
	None (1.0)	CPS	0.60	CPS	0.56	CPS	0.58	CPS	0.50

Table 8. Average overlap ratios for the cases when one or more alternative paths were searched

		User allowable travel cost ratio							
		Chicago network				Philadelphia network			
		Strong (1.1)		Weak (2.0)		Strong (1.1)		Weak (2.0)	
User allowable overlapped length ratio	Strong (0.5)	EVL	0.49	EVL	0.28	EVL	0.49	EVL	0.31
	Weak (0.8)	EVL	0.62	EVL	0.53	EVL	0.63	EVL	0.55
	None (1.0)	CPS	0.60	CPS	0.56	CPS	0.59	CPS	0.51

Table 9. Average overlap ratios for the overall sample data

4. Conclusions

In this chapter, we explained that the candidate path set is made by executing a shortest path search algorithm only once and compared the efficient vector labeling method (EVL) and the candidate path set method (CPS) to investigate the conditions for dissimilar paths-search algorithms by which drivers can select their own best path.

Navigation services should provide dissimilar alternative paths until their users are satisfied with a path based on their own criteria. This study suggests the method of selecting the path having the minimum average overlapped length ratio with a previously searched path as the alternative path from among paths that satisfy only the cost constraint. A test based on a real Chicago and Philadelphia networks showed that the proposed method can provide the number of alternative dissimilar paths required more rapidly. The generalized cost constraints are applied to all conditions at the same ratio since the travel cost stems from the entire path. On the other hand, the overlap constraints can vary among the alternative paths. For example, a driver may search for alternative paths until a path is provided that does not include the section he or she does not want. Therefore, navigation services must provide alternative paths to satisfy the various needs of users.

Drivers can select their own best paths from the alternative paths provided by using the information on several alternative paths and can see to what degree the alternative path provided by the service or that they select on their own is better than other paths. Drivers can also become familiar with unfamiliar regions by using different paths.

5. References

Akgün, V., Erkut, E., and Batta, R. (2000). On Finding Dissimilar Paths, *European Journal of Operational Research*, Vol. 121, pp. 232-246, 0377-2217

Azevedo, J. A., Costa, M. E. O. S., Madeira, J. J. E. R. S., and Martins, E. Q. V. (1993). An Algorithm from the Ranking of Shortest Paths, *European Journal of Operational Research*, Vol. 69, pp. 97-106, 0377-2217

Barra, T., Perez, B. and Anez, J. (1993). Multidimensional Path Search and Assignment, *Proceedings of 21st PTRC Summer Annual Conference*, pp. 307-319, Manchester, England, PTRC

Jeong, Y. J., Hong, Y., and Kim, T. J. (2007). Flexible Multipath Search Algorithm for Multipurpose Location-based Activities, *Transportation Research Record: Journal of the Transportation Research Board*, No. 2039, pp. 50-57, 0361-1981

Jeong, Y. J., Kim, T. J., Park, C. H., and Kim, D. -K. (2010). A Dissimilar Alternative Paths-Search Algorithm for Navigation Services: A Heuristic Approach, *KSCE: Journal of Civil Engineering*, Vol. 14, No. 1, pp.41-49, 1226-7988

Jeong, Y. J. (2010). A Dissimilar Paths Search Algorithm for a Multi-purpose Trip, Ph.D. Dissertation, Seoul National University.

Martí, R., Luis González Velarde, J., and Duarte, A. (2009). Heuristics for the Bi-Objective Path Dissimilarity Problem. *Computers and Operations Research*, Vol. 36, No. 11, pp. 2905-2912, 0305-0548

Martins, E. Q. V. (1984). An Algorithm for Ranking Paths that May Contain Cycles. *European Journal of Operational Research*, Vol. 18, pp. 123-130, 0377-2217

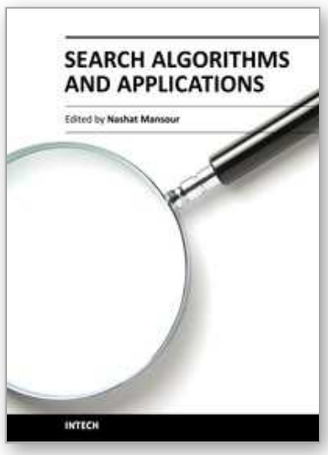
Meng, Q., Lee, D. -H., Cheu, R. L. (2005). Multiobjective Vehicle Routing and Scheduling Problem with Time Window Constraints in Hazardous Material Transportation. *ASCE: Journal of Transportation Engineering*, Vol. 131, No. 9, pp. 699-707, 0733-947X

Park, D., Sharma, S. L., Rilett, L. R., and Chang, M. (2002). Identifying Multiple Reasonable Alternative Routes: Efficient Vector Labeling Approach. *Transportation Research Record: Journal of the Transportation Research Board*, No. 1783, pp. 111-118, 0361-1981

Shier, R. D. (1976). Iterative Methods for Determining the K Shortest Paths in a Network. *Networks*, Vol. 6, pp. 205-229, 0028-3045

Shier, R. D. (1979). On Algorithms from Finding the K Shortest Paths in a Network. *Networks*, Vol. 9, pp. 195-214, 0028-3045

Yen, J. Y. (1971). Finding the K shortest Loopless Paths in a Network. *Management Science*, Vol. 17, pp. 712-716, 0025-1909



Search Algorithms and Applications

Edited by Prof. Nashat Mansour

ISBN 978-953-307-156-5
Hard cover, 494 pages
Publisher InTech
Published online 26, April, 2011
Published in print edition April, 2011

Search algorithms aim to find solutions or objects with specified properties and constraints in a large solution search space or among a collection of objects. A solution can be a set of value assignments to variables that will satisfy the constraints or a sub-structure of a given discrete structure. In addition, there are search algorithms, mostly probabilistic, that are designed for the prospective quantum computer. This book demonstrates the wide applicability of search algorithms for the purpose of developing useful and practical solutions to problems that arise in a variety of problem domains. Although it is targeted to a wide group of readers: researchers, graduate students, and practitioners, it does not offer an exhaustive coverage of search algorithms and applications. The chapters are organized into three parts: Population-based and quantum search algorithms, Search algorithms for image and video processing, and Search algorithms for engineering applications.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen