유방향의 복수 최단 우회 경로 새로운 해법 연구

장 병 만
(서울산업대학교 공과대학 산업정보시스템공학과, bmc@snut.ac.kr)

A Study on a New Algorithm for K Shortest Detour Path Problem in a Directed Network

Chang, Byung Man
(Dept. of I&ISE, Seoul National University of Technology)

Abstract

This paper presents a new algorithm for the K shortest path problem in a directed network. After a shortest path is produced with Dijkstra algorithm, detouring paths through inward arcs to every vertex of the shortest path are generated. A length of a detouring path is the sum of both the length of the inward arc and the difference between the shortest distance from the origin to the head vertex and that to the tail vertex. K-1 shorter paths are selected among the detouring paths and put into the set of K paths. Then detouring paths through inward arcs to every vertex of the second shortest path are generated. If there is a shorter path than the current Kth path in the set, this path is placed in the set and the Kth path is removed from the set, and the paths in the set is rearranged in the ascending order of lengths. This procedure of generating the detouring paths and rearranging the set is repeated for the K-1st path of the set.

This algorithm can be applied to a problem of generating the detouring paths in the navigation system for ITS and also for vehicle routing problems.

Key words: K Shortest Path, Detour Path, ITS, Dijkstra Method

1. 서론

본 연구에서는 K-1개의 우회경로를 포함한 K개의 최단경로를 찾는 복수최단경로문제(K-Shortest Path Problem)의 새로운 해법을 제시하고자 한다.

복수최단경로문제는 물류와 교통흐름에서 우회경로와 다른 수송경로를 찾는 문제나 ITS(Intelligent Transport System)내의 차량안내시스템 내에서 모든 지점간의 복수최단경로 등을 구하는 문제 등에 요긴한 해법 연구에 사용될 수 있다.

이 복수최단경로문제는 환을 가지는 경로를 허용하지 않는 경우를 다룬다. 유방향의 네트워크 G(V,E) 상에서 한방향의 호를 가지며, V는 n개 마디의 집합이고, E는 m개 호의 집합이며(m≥n), 비음의 호의 길이를 가지며 호(u,v)∈E 에서 d(u,v) = d(v,u)를 가정한다.

Lawler[7], Drefus[2], Yen[10]등은 $O(Kn^3)$의 해법을 제안했으며 Katoh외 2인[5]과 Hadjiconstantinou와 Christofides[4]는 $O(Kn^2)$해법을 제안했다. Yen[10]은 유방향이나 무방향 네트워크에서 후보경로들을 구하고 그 중 가장 짧은 경로를 찾는 과정을 반복하는 $O(Kn^3)$의 계산 복잡도를 가지는 해법을 개발하였다. Katoh외 2인 [5]은 무방향 네트워크상에서 그 지점간 최단경로를 구한 뒤, 매번 특정마디와 최종마디까지 각 마디와 호를 지나는 최단경로 가운데 3가지의 짧은 후보경로들을 구하고 그 중 가장 짧은 경로를 찾는 방법을 반복하는 O(Kn2)의 계산 복잡도를 가지는 우수한 해법을 개발했다.

기존의 해법 가운데 Connecting Algorithm은 최종 계산 후 K개의 경로가 정해지나, Label Setting Problem은 계산과정을 통해 최적해 중의 경로가 하나씩 정해진다. 본 연구의 해법은 Label Setting Algorithm의 한 유형이다.

여기서 사용하는 notation들은 다음과 같이 정의한다.

G(V,E) : 도로집합 E와 지점집합 V으로 구성된 유방향 네트워크

s : 출발지

t : 도착지

T_s : s 에서 모든 지점(vertex)까지의 최단경로 나무

C_{ij} : 도로 (ij) 통과 거리 (시간, 비용)

$\overline{C}_{i,j}$: 도로(ij)를 통과할 때 우회되는 거리의 길이

π_i : T_s상에서 지점 i까지의 최단거리

f_i : T_s상에서 지점 i의 전 지점 (predecessor)

$SP(s,a)$:s에서 지점 a까지 최단경로,$SP(s,a) \in T_s$

$v^l(i)$: P^l의 i번째 지점(노드)

q_l : P^l의 통과지점 개수, $v^l(q_1) = t$, $v^l(1) = s$

$N(P^l)$: P^l의 통과지점들의 집합

a : P^l상의 한 지점으로 진입하는 우회도로나 우회호 $(a,v^l(m))$의 출발지점, $(a,v^l(m)) \not\in T_s$,

r_l : s에서 P^l이 생기게 된 우회도로(i,j)의 i 지점까지의 통과지점의 수

P^1 : 가장 짧은 최단경로, $P^1 = [v^1(1), v^1(2), \cdots, v^1(q_1)]$

P^2: P^1의 한 지점으로 진입하여 우회하는 최단 우회경로, 2번째로 짧은 최단 경로

P^l : l 번째로 짧은 최단 우회경로, $P^l = [v^l(1), v^l(2), \cdots, v^l(q_l)]$

$P^l(i,j)$: P^l 의 부분경로이며, 지점 i 에서 지점 j 까지의 부분경로

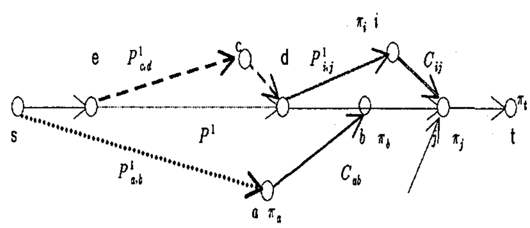
$P^l_{i,j}$: P^l 에서 나온 우회경로이며, 우회도로 (i,j)을 지나는 경로

$P^l_{i,j} = SP(s,i) \rightarrow (i,j) \rightarrow P^l(j,t)$ 로 표시한다.
$L(P^l)$: 경로 P^l 의 길이
$LD(P^l)$: P^l이 P^1보다 우회하는 거리 길이의 합
UB : k번째 우회경로의 길이, 우회경로 길이의 상한
KSP : 최적인 K개의 최단 경로의 집합, $KSP=\{P^{1*},P^{2*},\cdots,P^{k*}\}$
KP : K개의 경로의 집합, $KP=\{P^1,P^2,\cdots,P^k\}$

2. 해법 개발

본 연구에서는 첫단계로 출발지 s 에서 목적지 t 까지의 최단경로 P^1 과 모든 지점까지의 최단경로와 경로의 길이를 Dijkstra 법으로 구하고, 다음 단계로 P^1 의 우회경로들을 P^1의 각 지점들에 진입하는 각각의 우회호(우회도로) $(i,j) \not\in T_s$를 경유하는 우회경로 $P^1_{i,j}$ 들을 모두 구하고, 이 가운데서 P^1을 포함하여 K 개의 짧은 경로를 길이의 오름차순으로 정열하여 초기해 $KP=\{P^1,P^2,\cdots,P^k\}$를 정한다.
그 다음 단계로 $KP=\{P^1,P^2,\cdots,P^k\}$ 내의 우회경로 $P^2,...,P^{k-1}$ 각각에 대한 우회경로들을 차례로 찾는데, 상기의 P^1 의 우회경로들을 찾은 방법대로 각 지점들로 진입하는 우회도로들에 대해서 찾고 이 가운데서 P^k 보다 짧은 새로운 우회경로들을 KP 에 진입시키고 현재의 P^k를 탈락시켜 대체하는 절차를 P^{k-1} 경로에 대해서까지 반복 적용하여, 최적의 K개 최단경로의 해 $KSP=\{P^1,P^2,\cdots,P^k\}$를 찾는다.

P^1의 우회경로들은 P^1의 각 지점 $j \in N(P^1)$들에 진입하는 각각의 우회호(우회도로) $(i,j) \not\in T_s$, $SP(s,i) \in T_s$ 를 경유하는데, 이 우회경로 $P^1_{i,j}$들을 모두 찾을 수 있다.



<그림 1> 최단경로 P^1과 그 우회경로들

여기서 P^1은 s→e→d→b→j→t 일 때, $P^1_{i,j}$는 s→e→d→i→j→t, $P^1_{a,b}$ 는 s→a→b→j→t, $P^1_{c,d}$ 는 s→e→c→d→b→j→t 이다.

P^2가 $(a^*,v^1(i^*))$를 지나는 최단우회경로 라면, 이 P^2는 s에서 a^*까지는 최단의 경로 $SP(s,a^*) \in T_s$를 거친 후, a^*에서 하나의 우회도로$(a^*,v^1(i^*))$ $(\not\in T_s)$를 가지며, 이 우회도로의 끝지점인 P^1의 경로 상의 하나의 지점 $v^1(i^*)$으로 진입하여 P^1의 후부분경로인 $P^1(v^1(i^*),t)$를 거쳐 목적지 t로 간다.
P^1의 우회경로들 가운데 짧은 길이의 K-1개를 길이의 오름차순으로 선택 정렬하여 초기해 $KP=\{P^1,P^2,\cdots,P^k\}$ 를 구성한다.
UB = P^k로 둔다. $KP=\{P^1,P^2,\cdots,P^k\}$내에 들지 못한 경로들의 길이는 P^k보다 길고 그 우회경로들은 더 길므로, P^k 보다 긴 경로들은 제거(분지끝)하고 그 우회경로의 탐색을 위한 고려 대상에서 제외시킨다.
그리고 P^1의 우회경로들은 s에서 t까지 연결되는 경로이므로P^1과 반드시 우회도로를 통해 만나게 되는데, P^2는 이 우회도로 가운데 가장 짧으므로 P^2가 최단의 우회경로이다. 그러므로 $KSP=\{P^1,P^2\}$로 둔다.

우회경로 $P^l_{i,j}$는 하나의 경로 P^l의 우회경로라고 하자. 이 우회경로 $P^l_{i,j}$은 출발지점 s에서 우회도로의 꼬리지점 i까지는 π_i의 값을 가지는 최단경로 $SP(s,i) \in T_s$이고, P^l의 하나의 지점 j 로 진입하는 우회도로 $(i,j) \not\in T_s$를 가지며, P^l 상에서 우회도로와 만나는 지점 j 에서 목적지점 t까지는 P^l의 후반부 경로 $P^l(j,t)$로 구성된다.
이 우회경로는 $P^l_{i,j}$ 는 $SP(s,i) \in T_s, i \not\in N(P^l)$ 와 우회도로 $(i,j) \not\in T_s$ 와 $P^l(j,t)$로 구성되어 있으며, $P^l_{i,j} = SP(s,i) \rightarrow (i,j) \rightarrow P^l(j,t)$ 로 표시된다.

정리1: $SP(s,i) \in T_s$를 지나 우회도로 $(i,j) \not\in T_s$, $j \in N(P^l)$를 경유하는 우회경로 $P^l_{i,j}$이 우회하는 거리 $\overline{C}_{i,j}$ 는 $\pi_i + C_{i,j} - \pi_j$ 이다
증명:
$$\begin{aligned}\overline{C}_{i,j} &= L(P^l_{i,j}) - L(P^l)\\ &= [L(SP(s,i)) + C_{i,j} + L(P^l(j,t))]\\ &\quad - [L(SP(s,j)) + L(P^l(j,t))]\\ &= L(SP(s,i)) + C_{i,j} - L(SP(s,j))\\ &= \pi_i + C_{i,j} - \pi_j\end{aligned}$$

그러므로 우회도로 $P^l_{i,j}$는 P^l 에 비하여 우회하게 되는 거리는 다음과 같다.
$$\overline{C}_{i,j} = \pi_i + C_{i,j} - \pi_j \qquad \cdots\cdots\cdots\cdots\cdots\cdots(1)\text{식}$$

그러므로 우회경로의 길이 $L(P^l_{i,j})$ 는 다음과 같다.
$$\begin{aligned}L(P^l_{i,j}) &= L(SP(s,i)) + C_{i,j} + L(P^l(j,t))\\ &= \pi_i + C_{i,j} + (L(P^l) - \pi_j)\\ &= (\pi_i + C_{i,j} - \pi_j) + L(P^l)\\ &= \overline{C}_{i,j} + L(P^l) \qquad \cdots\cdots\cdots\cdots\cdots\cdots(2)\text{식}\end{aligned}$$

그러므로 $\overline{C}_{i,j}$는 P^l의 각 지점 $j \in N(P^l)$로 인접지점 $i \not\in N(P^l)$에서 들어오는 우회도로 $(i,j) \not\in T_s$에 대해 들어올 때 증가하는 우회경로의 추가되는 우회거리 이므로, 이 $\overline{C}_{i,j}$를 P^l의 각 지점 $j = v^l(m) \in N(P^l)$, $m = 2,...,q_l$ 에 대해 진입하는 우회도로 (i,j) 별로 구하고, (1)식에 의해 $L(P^l_{i,j})$을 계산할 수 있다.
$L(P^l_{i,j})$ < UB 이면 $P^l_{i,j}$ 가 P^k 를 대신하여 KP 에 들어가고 P^k 는 탈락된다.

정리2: P^l은 P^1에서 계속 여러 우회도로 $(i,j)\in P^l, (i,j)\not\in T_s$ 들을 거쳐 온 우회경로이면, 거쳐온 우회도로의 길이의 합은 $\sum_{\substack{(i,j)\in P^l\\(i,j)\not\in T_s}} \overline{C}_{i,j}$ 이고,

우회경로 P^l의 길이는 다음과 같다.

$$L(P^l) = \sum_{\substack{(i,j)\in P^l\\(i,j)\not\in T_s}} \overline{C}_{i,j} + L(P^1)$$

증명: 만약 P^l 이 그림과 같이 P^{l-1}에서 우회도로 (a,b), P^{l-1}은 P^{l-2}에서 우회도로 (c,d), P^{l-2}는 P^{l-3} 에서 우회도로 (e,f),....., P^2은 P^1에서 우회도로 (g,h)를 거쳐서 나온 우회경로라면, P^l의 길이는 다음과 같다.

$$\begin{aligned} L(P^l) &= \overline{C}_{a,b} + L(P^{l-1}) \\ &= \overline{C}_{a,b} + \overline{C}_{c,d} + L(P^{l-2}) \\ &= \overline{C}_{a,b} + \overline{C}_{c,d} + \overline{C}_{e,f} + L(P^{l-3}) \\ &= \cdots\cdots \\ &= \overline{C}_{a,b} + \overline{C}_{c,d} + \overline{C}_{e,f} + \ldots + \overline{C}_{g,h} + L(P^1) \\ &= \sum_{\substack{(i,j)\in P^l\\(i,j)\not\in T_s}} \overline{C}_{i,j} + L(P^1) \qquad \cdots\cdots\cdots(3)\text{식} \end{aligned}$$

또한 P^l이 P^1에서 계속 여러 우회도로 $(i,j)\in P^l, (i,j)\not\in T_s$ 들을 거쳐 거쳐온 우회도로의 길이의 합은 다음과 같다.

$$LD(P^l) = L(P^l) - L(P^1) = \sum_{\substack{(i,j)\in P^l\\(i,j)\not\in T_s}} \overline{C}_{i,j} \qquad \cdots\cdots\cdots(4)\text{식}$$

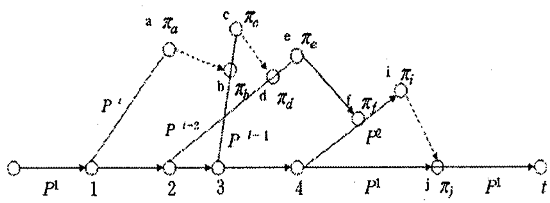
우회경로 P^l의 길이가 다음과 같고,

$$L(P^l) = \sum_{\substack{(i,j)\in P^l\\(i,j)\not\in T_s}} \overline{C}_{i,j} + L(P^1)$$

P^l의 우회경로 $P^l_{i,j}$의 길이는 다음과 같이 표현할 수 있다

$$\begin{aligned} L(P^l_{i,j}) &= L(SP(s,i)) + C_{i,j} + L(P^l(j,t)) \\ &= \overline{C}_{i,j} + L(P^l) \\ &= \sum_{\substack{(i,j)\in P^l_{ij}\\(i,j)\not\in T_s}} \overline{C}_{i,j} + L(P^1) \end{aligned}$$

그러므로 Dijkstra법으로 구한 T_s의 짧은 경로가지의 끝지점이 i이고 이 지점이 t와 연결이 안되어 있고 많이 떨어져 있다면, 우회도로를 하나이상 k-1개까지 거쳐서 P^1나 t를 만날 수 있다.
그러나 우회도로를 지날수록 우회거리의 합이 $\sum_{\substack{(i,j)\in P^l\\(i,j)\not\in T_s}} \overline{C}_{i,j}$ 만큼 증가하게 되어 있다.



<그림 2> P^l과 P^l이 우회한 이전의 경로들

경로명	경로	길이
P^1	s→1→2→3→4→j→t	$L(P^1)$
P^2	s→1→2→3→4→i→j→t	$L(P^2)=\overline{c}_{ij} + L(P^1)$
P^3	s→1→2→d→e→f→i→j→t	$L(P^{l-2})=\overline{c}_{ef} + L(P^2)$
P^{l-1}	s→1→2→3→c→d→e→f→i→j→t	$L(P^{l-1})=\overline{c}_{cd} + L(P^{l-2})$
P^l	s→1→a→b→c→d→e→f→i→j→t	$L(P^l)=\overline{c}_{ab} + L(P^{l-1})$

<표 1> 우회경로의 통과 지점들과 길이

P^l의 각 지점별로 진입하는 우회도로를 찾을 때, P^l의 경로 $P^l(s,t)$ 상에서 s를 제외한 각 지점으로의 진입 우회도로들을 찾아서 최단 우회경로를 찾는다.
그런데 P^l이 더 짧은 경로 $P^r, 1\le r\le l-1$ 에서 우회도로 $(j,v^r(m))$을 거쳐서 지나오는 우회경로이라면, j 에서 t 까지의 부분경로 $P^l(j,t)$상의 각 지점으로 진입하는 우회도로들은, 경로 P^1에서 P^{l-1} 까지를 포함한 KP 의 경로들을 찾는 과정에서 점검되었고, 이 과정에서 생긴 우회경로 P^e의 길이가 UB보다 작았으면, 이미 그 경로 P^e는 KP에 포함되어 있게 된다. KP 에 포함 안 된 $P(j,t)$상의 각 지점 별로 진입하는 우회도로들의 우회경로는 P^k 보다 길이가 더 길므로 찾을 필요가 없다.
그러므로 P^l의 우회경로를 찾을 때 부분경로 $SP(s,i)$ 즉 P^l의 부분경로 $P^l(s,i)\in T_s$상에서 s를 제외한 각 지점별로 우회경로를 찾고, P^l의 지점 j에서 t 까지의 부분경로 $P^l(j,t)$상의 각 지점으로 진입하는 우회도로들은 찾지 않는다.
그러면 새로운 우회경로의 탐색에서 계산복잡도를 줄일 수 있다.

이상의 과정을 $P^l, l=2,\ldots,k-1$ 의 경로들에 대해서 반복하면서, $UB(=P^k)$보다 짧은 길이의 경로 $P^l_{a,v^l(h^*)}$를 찾아 KP에 진입시키고 현재 P^k를 탈락시키면서, $KSP=\{P^1,P^2,\ldots,P^l,P^{l+1}\}$를 확정시킨다. P^{k-1} 경로까지 상기 과정을 반복하면 최종적으로 $KSP=\{P^1,P^2,\ldots,P^l,\ldots,P^k\}$가 구해진다.

정리3: 본 해법으로 구한 최종해 $KSP=\{P^1,\ldots,P^l,\ldots,P^k\}$는 최적해이다.

증명 : P^1은 최단 경로이다.
모든 우회경로는 P^1과 s와 t가 같기 때문에 P^1과 하나의 우회도로를 거쳐 만나게 되기 때문이다. 그래서 본 해법에서와 같이 P^1의 s에서 t까지의 경로 상의 모든 지점 $v^1(j)\ j=2,\cdot\cdot\cdot,q_1$ 각각에 대해 우회하여 진입하는 도로 $(a,v^1(j))$ $(\not\in T_s)$를 지나는 우회경로들을 찾아서 그 가운데

서 최단의 우회경로를 선정한 것이 P^2이다.

P^3은 P^1과 하나의 우회도로를 거쳐 만나거나, P^2와 하나의 우회도로를 거쳐 만나는 경로 즉 P^1에서 2개의 우회도로를 거치는 경로들 중에서 짧은 경로이다. 본 해법에서 P^1에서 P^2를 구할 때 P^1과 단 하나의 우회도로를 거쳐 만나는 경로들이 모두 점검되어 그 중에 짧은 K개가 선정되어 KP를 구성하였고, P^2에서 P^3를 구할 때 P^2와 단 하나의 우회도로를 거쳐 만나는 경로 즉 P^1에서 2개의 우회도로를 거치는 경로들이 점검되어 KP가 개선된다. KP 가운데 P^3는 P^2 다음으로 짧은 경로이고 이보다 더 짧은 우회경로가 없기 때문에 최적해에 요소중 하나이다.

상기와 같은 방법을 반복하여 우회경로 P^3부터 P^{k-1}까지의 각 경로 P^l에 대해서 각 경로의 s를 제외한 부분경로 $SP(s,j^*)\in T_s$ 상의 각 지점 j로 진입하는 우회도로(i,j)를 통과하는 우회경로를 발생시켜 현재의 P^k보다 짧은 경로가 나오면, 이는 P^1과 단 하나의 우회도로를 거쳐 만나거나, P^2와 단 하나의 우회도로를 거쳐 만나는 경로 즉 P^1에서 2개의 우회도로를 거쳐 P^1과 만나는 경로이거나, P^{l-1}와 단 하나의 우회도로를 거쳐 만나는 경로 즉 P^1에서 $l-1$개의 우회도로를 거쳐 P^1과 만나는 경로들 중에서 이제까지 드러나지 않았던 짧은 경로이다. 현재의 P^k를 탈락시키고 이 짧은 경로를 진입시키는 방식으로 KP를 개선시키면서, 긴 경로들은 제거(분지 끝)하고 그 우회경로들도 탐색 고려 대상에서 제외시키는 작업을 반복한다. 이렇게 구한 $KSP=\{P^1,...,P^l,...,P^k\}$는 K개 최단 경로로 구성된 최적해이다.

k개 최단경로를 구하기 위한 해법의 절차는 분지한계법의 개념을 활용한다.

분지전략에서는 P^1과 P^2부터 P^{k-1}경로에 대해서 각 지점으로 진입하는 우회경로를 구하여 UB보다 작은 경로들이 발생하면 이 경로를 현재의 P^k와 대체하고, 경로길이의 오름차순으로 KP를 정리한다.

탐색전략에서는 다음의 우회경로 P^l을 KP에서 찾는다. $l=k$이면 STOP한다.

한계전략에서는 P^l의 우회경로를 찾고, P^l의 부분경로 $P^l(s,j),\in T_s$ 상의 각 지점 j^*으로 진입하는 우회도로 $(i,j)\not\in T_s$, $j\in P^l(v^l(2),j)$들에 의한 증가되는 길이 $\overline{C}_{i,j}$를 각각 점검하여, $\overline{C}_{i,j} < UB-L(P^l)$이면, $P^l_{i,j}$를 진입시키고 현재의 P^k를 탈락시킨다. $\overline{C_{ij}} \geq UB - L(P^l)$이면 $P^l_{i,j}$가 P^k보다 같거나 길므로 경로를 찾지 않고 버리도록 하여, 계산과정을 간소화하도록 한다.

여기서 K번째 우회경로의 길이 $L(P^k)$를 UB로 두고 이 UB를 낮추어 간다.

$l=l+1$로 두고 탐색전략으로 간다.

위의 과정을 $k-1$번째 경로 P^{k-1}에 대해서 까지 반복한다

P^l의 우회경로를 점검할 때, $\overline{C_{ij}}$를 각 $(i,j)\not\in T_s$, $j\in P^l(v^l(2),j)$에 대해 계산하고, $\overline{C_{ij}} < UB - L(P^l)$이면 $P^l_{i,j}$를 KP에 넣고, 이 $P^l_{i,j}$의 $SP(s,i)\in T_s, i\in N(P^l_{i,j})$를 기억하여 이 $P^l_{i,j}$의 우회경로를 찾을 때 활용하고, 그렇지 않으면 무시한다.

계산복잡도(Complexity)의 검토

P^1을 Dijkstra법으로 계산--- $O(n^2)$

초기해 KP 를 구하기 위해, P^1상의 각 지점을 진입하는 우회도로를 점검하여 $\overline{C_{ij}}$를 구하는 것 --- $O(n*n)$,

KP 내의 $P^2,...,P^k$를 오름차 순으로 정리하는 것 --- $O(n)$

상기 과정을 $P^2,...,P^{k-1}$에 대한 우회경로를 찾으면서 K-2회 반복

총 계산복잡도는 다음과 같이 $K.O(n^2)$ 이 된다.

$O(n^2) + (K-2)\cdot O(n^2 + n) = K\cdot O(n^2)$

3. KSP의 우회경로 해법

단계1. 네트워크 $G(V,E)$의 도로(호)와 거리값 정보 및 K값 입력

단계2. T_s를 Dijkstra법으로 생성한다.
P^1, $L(P^1)$을 구한다.
$l=1$

단계3. 분지

1. P^1의 s를 제외한 지점 $j=v^1(2),v^1(3),...,v^1(q_1)$, $v^1(q_1)=t$ 에 우회하여 진입하는 우회도로 $(i,j), j\in N(P^1)$, $(a,j)\not\in T_s$들을 찾고, 각 우회도로 (i,j)를 거칠 때 우회하는 추가 길이 $\overline{C}_{i,j}$를 구한다.
   - $\overline{C}_{ij} = \pi_i + C_{ij} - \pi_j,\ j\in N(P^1)$
2. P^1과 우회경로들의 길이의 오름 차 순으로 K개의 짧은 경로들을 선정하고 길이를 계산한다.
   - $L(P^1_{i,j}) = L(P^1) + \overline{C}_{ij}$
   - $KP=\{P^1,P^2,P^3,\cdots,P^k\}$
   - $L(P^1),L(P^2),...,L(P^k)$ 계산

   만약 Path의 수가 K개 미만이면 모두 택한다
3. $UB=L(P^k)$
   $l=l+1$
   $KSP=\{P^1,...,P^l\}$

단계4. 탐색

1. $K=2$ 이거나 $l=K$이면, STOP.
   최적해 $KSP=\{P^1,...,P^l,...,P^k\}$ 결정
   길이 $L(P^1),L(P^2),...,L(P^k)$ 계산
2. o/w, KP내의 P^l 을 선택한다.

단계5. 한계

1. P^l이 만들어진 우회도로(i,j)의 지점 i를 확인
   부분경로 $P^l(v^l(2),i),\in T_s$ 상의

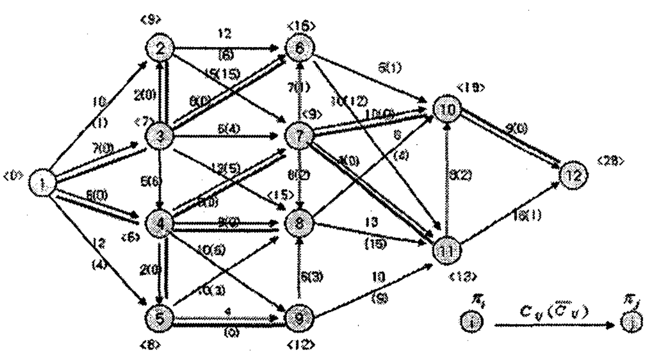
각 지점 i^*의 각 우회도로 (a,i^*)에 대해서
$\overline{C}_{a,i^*} = \pi_a + C_{a,i^*} - \pi_{i^*}$ 계산

2. P^l_{a,i^*}의 우회거리의 길이 계산
--- $LD(P^l_{a,i^*})$
3. $LD(P^l_{a,i^*}) = \sum_{\substack{(i,j)\in P^l(a,i^*) \\ (i,j)\not\in T_s}} \overline{C}_{i,j} > UB - L(P^1)$,
$\forall i^* \in P^l, (a,i^*) \not\in T_s$이면
$l = l+1$, 단계 4로 간다.
o/w, 우회경로 $P^l_{a,i^*}, (a,i^*) \not\in T_{s,} i^* \in P^l$ 들을 모두 KP에 포함시키고, 현재의 P^k를 탈락시킨다.
4. 개선해를 작성한다.
KP내의 경로들을 오름차순으로 K개 경로를 선정하여 정렬시킨다.
단계 3-3으로 간다.

4. 적용예제

그림 1의 K=10인 최단경로 문제를 개발한 해법으로 풀어보면 다음과 같다.

단계 1. K=10
각 도로별 거리 정보 입력
단계 2. T_s를 Dijkstra법으로 생성 --- 그림3 참조
P^1: 1-4-7-10-12,
$L(P^1)$ = 28
$l = 1$



<그림3> 예제 네트워크와 T_s

단계 3. 1) P^1의 각 지점으로 진입하는 우회도로별 $\overline{C}_{ij}$계산
$\overline{C}_{11,12} = 13 + 16 - 28 = 1$,
$\overline{C}_{6,10} = 1$, $\overline{C}_{8.10} = 4$, $\overline{C}_{11,10} = 2$
$\overline{C}_{2,7} = 15$, $\overline{C}_{3,7} = 4$, $\overline{C}_{3,4} = 6$
2) P^1과 우회경로들의 길이의 오름차순으로 K개의 짧은 경로들을 선정하고 길이를 계산한다.

지점	우회 도로 (i,j)	우회 거리 $\overline{C}_{ij}$	우회경로 길이 $L(P^l)$	우회경로	경로 번호
12	(11,12)	1	28+1 =29	1-4-7-11-12	P^2
10	(6,10)	1	28+1 =29	1-3-6-10-12	P^3
	(11,10)	2	28+2 =30	1-4-7-11-10-12	
	(8,10)	4	28+4 =32	1-4-8-10-12	P^4 P^5
7	(3,7)	4	28+4 =32	1-3-7-10-12	P^6
4	(3,4)	6	28+6=34	1-3-4-7-10-12	P^7
7	(2,7)	15	28+15=43	1-3-2-7-10-12	P^8

$UB = \infty \rightarrow$ P^{10} 이 아직 안 나왔음으로
$KSP = [P^1, P^2]$
$l = l+1$

(Iteration 1)
단계 4. P^2선택
단계 5. P^2는 P^1의 우회도로(11,12)를 지나가는 우회경로이다.
$P^2 = P^1_{11,12}$: 1-4-7-11-12
P^2에서 각 지점별로 우회도로 탐색하여 우회경로 생성
P^2의 우회경로생성

지점	우회도로(i,j)	$LD(P^l_{a,i^*})$	비고
11	(9,11)	1+9=10	***
	(6,11)	1+12=13	
	(8,11)	1+15=16	
7	(2,7)	1+15=16	
	(3,7)	1+4=5	***
4	(3,4)	1+6=7	***

여기 나온 해에서 우회도로(3, 7), (3, 4), (9,11)을 지나는 경로를 KP에 포함시키고, 현재 가장 긴 우회경로인 P^8를 탈락시킨다.
$LD(P^8)$ = 15 > $LD(P^2_{3,7})$ = 5
> $LD(P^2_{3,4})$ = 7
> $LD(P^2_{9,11})$ = 10
개선된 해 KP를 작성한다.

번호	우회경로	$LD(P^l_{a,i^*})$	우회도로
P^1	1-4-7-10-12	1	
P^2	1-4-7-11≈12	1	(11,12)
P^3	1-3-6≈10-12	1	(6,10)
P^4	1-4-7-11≈10-12	2	(11,10)
P^5	1-4-8≈10-12	4	(8,10)
P^6	1-3≈7-10-12	4	(3,7)
P^7	1-3≈7-11≈12	5	(11,12),(3,7)
P^8	1-3≈4-7-10-12	6	(3,4)
P^9	1-3≈4-7-11≈12	7	(11,12),(3,4)
P^{10}	1-4-5-9≈11≈12	10	(11,12),(9,11)

단계 3.
UB = 28+10 = 38
$KSP = [P^1, P^2, P^3]$
l = 2 + 1 = 3

(Iteration 2)
단계 4, P^3 선택

단계 5, P^3 : 1-3-6-10-12,
$LD(P^3)$ = 1
우회도로(6.10)

지점	우회도로(i,j)	$LD(P^l_{a,i*})$	비고
12	없음		
10	(8,10) (11,10)		
6	(7,6) (2,6)	1 + 1 = 2 1 + 6 = 7	*** ***
3			우회진입경로 없음

우회경로 생성
우회도로(7. 6), (2,6)을 지나는 경로가 $LD(P^3_{7,6})$ = 2, $LD(P^3_{2,6})=7$
현재의 P^{10}을 탈락시키고, P^9는 $LD(P^l_{a,i*})$=7 이므로 그대로 둔다. $P^3_{7,6}$을 KP에 진입시킨다.

개선된 해 KP를 작성한다.

번호	우회경로	$LD(P^l_{a,i*})$	우회도로
P^1	1-4-7-10-12	0	
P^2	1-4-7-11 ≈ 12	1	(11,12)
P^3	1-3-6 ≈ 10-12	1	(6,10)
P^4	1-4-7-11 ≈ 10-12	2	(11,10)
P^5	1-4-7 ≈ 6 ≈ 10-12	2	(6,10),(7,6)
P^6	1-4-8 ≈ 10-12	4	(8,10)
P^7	1-3 ≈ 7-10-12	4	(3, 7)
P^8	1-3 ≈ 7-11 ≈ 12	5	(11,12),(3,7)
P^9	1-3 ≈ 4-7-10-12	6	(3, 4)
P^{10}	1-3 ≈ 4-7-11 ≈ 12	7	(11,12),(3, 4)

UB = 28+7 = 35
$KSP=[P^1,P^2,P^3,P^4]$
l = 3 + 1 = 4

(Iteration 3)
단계 4, P^4 선택
단계 5, P^4 : 1-4-7-11-10-12,
$LD(P^4)$ = 2
우회도로(11,10)
우회경로 생성

지점	우회도로(i,j)	$LD(P^l_{a,i*})$	비고
11	(9,11) (8,11) (6,11)	2 + 9 = 11 2 + 15 =17 2 + 12 =14	
7	(3,7) (2,7)	2 + 4 = 6 2 + 15 =17	***
4	(3,4)	2 + 6 = 8	

$LD(P^{10})$ = 7 > $LD(P^4_{3,7})$ = 6

현재의 P^{10}을 탈락시키고 P^4_{37}을 KP에 진입시킨다

개선된 해 KP를 작성한다.

번호	우회경로	$LD(P^l_{a,i*})$	우회 도로
P^1	1-4-7-10-12	0	
P^2	1-4-7-11 ≈ 12	1	(11,12)
P^3	1-3-6 ≈ 10-12	1	(6,10)
P^4	1-4-7-11 ≈ 10-12	2	(11,10)
P^5	1-4-7 ≈ 6 ≈ 10-12	2	(6,10),(7,6)
P^6	1-4-8 ≈ 10-12	4	(8,10)
P^7	1-3 ≈ 7-10-12	4	(3, 7)
P^8	1-3 ≈ 7-11 ≈ 12	5	(11,12)(3,7)
P^9	1-3 ≈ 4-7-10-12	6	(3, 4)
P^{10}	1-3 ≈ 7-11 ≈ 10-12	6	(11,10),(3,7)

UB = 6+28 = 34
$KSP=[P^1,P^2,P^3,P^4,P^5]$
l = 4 + 1 = 5 , 단계 4로 간다

(Iteration 4)
단계 4, P^5 선택
단계 5, P^5 : 1-4-7-6-10-12,
$LD(P^5)$ = 2
우회도로(7,6)
우회경로 생성

지점	우회도로(i,j)	$LD(P^l_{a,i*})$	비고
7	(3,7) (2,7)	2 + 4 = 6 2 +15 = 17	
4	(3,4)	2 + 6 = 8	

$LD(P^{10})$ = 6 = $LD(P^5_{3,7})$
현재의 해를 그대로 유지시킨다
$KSP=[P^1,P^2,P^3,P^4,P^5,P^6]$
l = 5 + 1 = 6 단계 4로 간다

(Iteration 5)
단계 4, P^6 선택
단계 5, P^6 : 1-4-8-10-12 ,
$LD(P^6)$ = 4
우회경로(8,10)
우회경로 생성

지점	우회도로	$LD(P^l_{a,i*})$	비고
8	(7,8) (3,8) (5,8) (9,8)	4 + 2 = 6 4 + 5 = 9 4 + 3 = 7 4 + 3 = 7	LD(P^{10}) =6 이므로 개선안됨
4	(3,4)	4 + 6 = 10	

$LD(P^{10}) = 6 = LD(P^6_{7,8})$
현재의 해를 그대로 유지시킨다.
$KSP = [P^1, P^2, P^3, P^4, P^5, P^6, P^7]$
$l = 6 + 1 = 7$, 단계 4로 간다

(Iteration 6)
단계 4, P^7 선택
단계 5, P^7 : 1-3-7-10-12,
$LD(P^7) = 4$
우회도로(3,7)
우회경로 생성

지점	우회도로(i,j)	$LD(P^l_{a,i*})$	
3	×		

현재의 해를 그대로 유지시킨다.
$KSP = [P^1, P^2, P^3, P^4, P^5, P^6, P^7, P^8]$
$l = 7 + 1 = 8$, 단계 4로 간다
(Iteration 7)
단계 4, P^8 선택
P^8 : 1-3-7-11-12 ,
$LD(P^8) = 5$
우회경로(3,7)
우회경로 생성 - 지점 3에서는 진입하는 우회경로 없음
현재의 해를 그대로 유지시킨다
$KSP = [P^1, P^2, P^3, P^4, P^5, P^6, P^7, P^8, P^9]$
$l = 8 + 1 = 9$
(Iteration 8)
단계 4, P^9 선택
$LD(P^9) = LD(P^{10}) = 6$
$LD(P^9) > 6$ 이므로 현재의 해를 유지
$KSP = [P^1, P^2, P^3, P^4, P^5, P^6, P^7, P^8, P^9, P^{10}]$
$l = 9 + 1 = 10$, 단계 4로 간다

(Iteration 9)
단계 4 $l = K = 10$, STOP

그러므로 최종해 KSP는 다음과 같다

번호	우회경로	$LD(P^l_{a,i*})$	$L(P^l)$최단 경로길이
P^1	1-4-7-10-12	0	28
P^2	1-4-7-11≈12	1	29
P^3	1-3-6≈10-12	1	29
P^4	1-4-7-11≈10-12	2	30
P^5	1-4-7≈6≈10-12	2	30
P^6	1-4-8≈10-12	4	32
P^7	1-3≈7-10-12	4	32
P^8	1-3≈7-11≈12	5	33
P^9	1-3≈4-7-10-12	6	34
P^{10}	1-3≈7-11≈10-12	6	34

5. 결론 및 추후 방향

본 연구에서는 K개의 복수최단 경로문제에 대해서 중복 방문이 없는 해를 구하는 $K.O(n^2)$의 해법을 제시하였다.

이 해법은 전방향의 최단 경로 나무 T_s를 구한 후, 각 도로별로 우회거리 $\overline{C}_{ij}$ 를 구하고, 이 우회거리 정보를 이용하여 T_s상에서 우회 최단 경로들을 구하고, 여기서 K개 경로를 구하고 이 경로들에서 더 짧은 우회경로를 찾아 개선하는 과정을 반복하여 K개의 최단 경로를 구하는 해법으로 분지한계법의 개념을 이용한 최적해법이다.

추후 연구 과제로는 기존의 최단 경로 문제처럼 알고리즘의 실행을 보다 효율적으로 할 수 있으면서 계산 복잡도가 낮은 보다 효율적인 해법을 찾는 것과, 실제 도로망이나 교통·수배송 등 활용 영역에서의 검증이 필요하다. 또한 ITS 등과 같은 실제 활용 영역에서의 특성상 요구되는 모든 마디간의 복수 최단 경로 문제 해결을 위한 효율적인 해법의 연구가 계속적으로 요망된다.

참 고 문 헌

[1] Dijkstra, E.W., A note on two problems in connection with graphs, Numerische Mathematik, 1(1959), pp.269-271.
[2] Dreyfus, S., An appraisal of some shortest path algorithms, Oper Res, 17, 2(1969), pp. 395-412.
[3] Eppstein, D., Finding the K shortest paths, SIAM J. Comput., 28, 2 (1998), pp.652-673.
[4] Hadjiconstantinou, E., and N. Christofides, An efficient implementa -tion of an algorithm for finding K shortest paths, Networks, 34 (1999), pp.88-101.
[5] Katoh, N., T. Ibaraki, and H. Mine, An efficient algorithm for K shortest simple paths, Networks, 12(1982), pp.411-427.
[6] Lalgudi, K.N. and M.C. Papaefthymiou, Computing strictly-second shortest paths, Information Processing Letters, 63 (1997), pp. 177-181.
[7] Lawler, E., A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem, Management Sci, 18, 7(1972), pp.401-405.
[8] Lawler, E., Combinatorial Optimization : Networks and Matroids, Holt Reinhart and Winston, New York, 1976.
[9] Shier, D., On algorithm for finding the K shortest paths in a network, Networks, 9 (1979), 195-214.
[10] Yen, J. Finding the K shortest loopless paths in a network, Management Sci, 17 (1971), pp.712-716.
[11] 장병만, "복수최단경로의 새로운 최적해법", 한국경영과학회지, 제26권 제3호(2001), pp.79-94
[12] 장병만, "복수최단경로문제의 새로운 해법연구", 경영과학, 제15권 제2호(1998), pp.229-237