복수 최단 경로의 새로운 해법에 관한 연구*

장 병 만**

A Study on a new Algorithm for *K* Shortest Paths Problem*

Byung Man Chang**

Abstract

This paper presents a new algorithm for the *K* shortest paths problem in a 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 until the K^{th}-1 path of the set is obtained.

The computational results for networks with about 1,000,000 nodes and 2,700,000 arcs show that this algorithm can be applied to a problem of generating the detouring paths in the metropolitan traffic networks.

Keywords : *K* Shortest Paths, Detour Paths, Dijkstra Method

1. 서 론

본 연구에서는 유방향이나 무방향의 네트워크 상에서 출발지 s에서 목적지 t로 가는 $K(K>1)$ 개의 최단경로들을 찾는 복수최단경로문제(K-Shortest Paths Problem)의 새로운 해법을 제시하고자 한다.

논문접수일 : 2007년 02월 10일 논문게재확정일 : 2008년 07월 08일
* 본 연구는 서울산업대학교의 기성회계 학술연구비 지원으로 수행되었습니다.
** 서울산업대학교 산업정보시스템공학과

복수최단경로문제는 수송과 물류와 교통흐름에서 우회경로들과 다양한 수송경로들을 찾는 문제나 ITS(Intelligent Transport System)내에서 주요 지점들 간의 교통량 분산 유도를 위한 우회경로 탐색, 네비게이션시스템 내에서 복수개의 운행경로안내를 위한 각 지점간의 복수최단경로 탐색, 위험물 수송을 위한 저위험도의 복수 수송경로 탐색 문제 등에 활용되는 해법 연구, 웹환경에서의 문서검색 시 여러 개의 검색결과를 탐색하는 해법 연구에 사용될 수 있다.

이 복수최단경로문제는 환을 가지는 경로를 허용하지 않는 경우를 다룬다. 유방향이나 무방향의 네트워크 $G(V, E)$상에서, V는 n개 마디(지점)의 집합이고, E는 m개 호의 집합이며($m \geq n$), 비음의 호의 길이를 가지며, 호$(u, v) \in E$에서 $d(u, v) = d(v, u)$를 가정한다.

Lawler(1972), Drefus(1969), Yen(1971)등은 $O(Kn^3)$의 해법을 제안했으며 Katoh et al.(1982)과 Hadji-constantinou and Christofides(1999)는 $O(Kn^2)$해법을 제안했다.

Yen(1971)은 유방향이나 무방향 네트워크에서 후보경로들을 구하고 그 중 가장 짧은 경로를 찾는 과정을 반복하는 $O(Kn^3)$의 계산 복잡도를 가지는 해법을 개발하였다.

Katoh et al.(1982)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 $O(Kn^2)$의 계산 복잡도를 가지는 해법을 개발했다.

Martins et al.(2001)은 K번의 각 단계에서 단 하나의 최단경로 만을 구하여 첨가하며, 각 단계에서 기존의 네트워크에서 가상 마디(Dummy Node)와 가상 호(Dummy Arc)를 추가하는 $O(Kn^2)$의 계산 복잡도를 가지는 removing path 해법을 개발했다.

복수최단경로문제를 위한 기존의 해법들은 매 계산과정마다 짧은 경로를 하나씩 더해 K번 반복 후에 K개의 경로가 정해지는 해법인데 반하여, 본 연구에서는 ITS 등 빠른 시간 내에 K개의 짧은 경로인 가능해를 처음부터 구하기 위한 목적으로, 매 계산과정마다 K개의 가능경로를 구하고 최적해 K개의 경로에 포함되는 경로를 하나 이상씩 추가로 확정하면서 최적해를 찾는 해법을 채택하였다.

이러한 계산방식은 기존의 복수최단경로해법과는 다르며, 본 연구에서 처음으로 제시하는 것으로 사료된다.

여기서 사용하는 기호(Notations)들은 다음과 같이 정의한다.

$G(V, E)$: 도로집합 E와 지점집합 V로 구성된 네트워크

s : 출발지

t : 도착지

T_s : s에서 모든 지점(vertex)까지의 최단경로 나무

C_{ij} : 도로 (i, j) 통과 거리(시간, 비용)

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

π_i : T_s상에서 지점 i까지의 최단거리

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

$v^l(i)$: P^l의 i번째 지점(마디)

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

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

$N(P^l)$: P^l의 통과지점들의 집합

a : P^l상의 한 지점으로 진입하는 우회도로나 우회호$(a, v^l(m))$의 출발지점, $(a, v^l(m)) \notin T_s$

P^l : l번째로 짧은 최단 우회경로,

$P^l = [v^l(1), v^l(2), \cdots, v^l(q_l)]$

$P^l(i, j)$: P^l의 부분경로이며, 지점 i에서 지점 j까지의 부분경로

$P^l{}_{i,j}$: P^l에서 나온 우회경로이며, 우회도로 (i, j)을 지나 P^l과 연결되는 경로

$P^l{}_{i,j} = SP(s, i) \rightarrow (i, j) \rightarrow P^l(j, t)$ 로 표시한다.

$L(P^l)$: 경로 P^l의 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. 우회경로 탐색해법 개발

본 해법에서는 첫 단계로 네트워크상에서 첫 단계로 출발지 s에서 목적지 t까지의 최단경로 P^1과 모든 지점까지의 최단경로와 경로들의 길이를 Dijkstra법으로 구하고, 다음 단계에서 P^1의 우회경로들을 P^1의 각 지점 j들에 대해서, 유방향 네트워크에서는 진입하는 각각의 우회호(우회도로), 무방향 네트워크에서는 연결되는 호 $(i, j) \notin T_s$를 경유하는 우회경로 $P^1_{i,j}$들을 모두 구하고, 이 가운데서 P^1을 포함하여 K개의 짧은 경로들을 길이의 오름차순으로 정렬하여 초기해 $KP=\{P^1, P^2, \cdots, P^k\}$를 정한다.

우회경로 $P^l_{i,j}$를 구할 때, P^l의 각 지점 j들에 대해서, 유방향 네트워크에서는 진입하는 각각의 우회호(우회도로), 무방향 네트워크에서는 연결되는 호 $(i, j) \notin T_s$를 경유하는 우회경로를 구하는 것인데, 무방향 네트워크는 여기서 양방향 네트워크이고, 그 외의 다른 절차들은 유방향 네트워크 상에서와 같으므로, 본 연구에서는 유방향 네트워크 상에서 최적의 K개 최단경로의 해를 구하는 절차로 설명하고자 한다.

그 다음 단계에서 $KP=\{P^1, P^2, \cdots, P^k\}$ 내의 우회경로 $P^2, \cdots, P^{k-1}$ 각각에 대한 새로운 우회경로들을 차례로 찾는데, 상기의 P^1의 우회경로들을 찾은 방법대로 $P^2, \cdots, P^{k-1}$의 각 지점들로 진입하는 우회도로들을 통과하는 새로운 우회경로를 찾고, 이 가운데서 P^k보다 짧은 새로운 우회경로들을 KP에 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^2가 P^1의 하나의 지점 j와 연결되는 호 (a^*, j)를 지나는 최단우회경로이면, 이 P^2는 s에서 a^*까지는 최단의 경로 $SP(s, a^*) \in T_s$를 지난 후, a^*에서 하나의 우회도로 $(a^*, j)(\notin T_s)$를 거쳐서, 이 우회도로의 끝 지점인 P^1의 경로 상의 지점 j으로 진입하여 P^1의 후부분경로인 $P^1(j, t)$를 통과하여 목적지 t로 간다.

P^1의 우회경로들 가운데 짧은 길이의 $K-1$개의 경로들을 길이의 오름차순으로 선택 정렬하여 초기해 $KP=\{P^1, P^2, \cdots, P^k\}$를 구성한다.

$UB=P^k$로 둔다. $KP=\{P^1, P^2, \cdots, P^k\}$내에 들지 못한 경로들의 길이는 P^k보다 길고 그 우회경로들은 더 길므로, P^k보다 긴 경로들은 제거(분지 끝)하고, 이후의 우회경로의 탐색을 위한 고려 대상에서 제외시킨다.

그리고 P^1의 우회경로들은 s에서 t까지 연결되는 경로이므로 우회도로를 거쳐서 P^1과 반드시 만나게 되는데, P^2는 이 우회도로 가운데 가장 짧으므로, P^2가 최단의 우회경로이다. $KSP=\{P^1, P^2\}$로 둔다.

상기와 같은 방식으로 우회경로 $P^l_{i,j}$은 출발지점 s에서 우회도로(i, j)의 꼬리지점 i까지는 π_i의 값을 가지는 최단경로 $SP(s, i) \in T_s$를 지나고, P^l의 하나의 지점 j로 진입하는 우회도로 $(i, j) \notin T_s$를 지나며, P^l 상의 지점 j에서 목적지점 t까지는 P^l의 후반부 경로 $P^l(j, t)$를 거치게 되고, $P^l_{i,j} = SP(s, i) \rightarrow (i, j) \rightarrow P^l(j, t)$로 표시된다.

정리 1 : $SP(s, i) \in T_s$를 지나 우회도로 $(i, j) \notin T_s$, $j \in N(P^l)$를 경유하는 우회경로 $P^l_{i,j}$가 P^l에 비해서 우회 하는 거리 $\bar{C}_{i,j}$는 $\pi_i + C_{i,j} - \pi_j$이다

증명 :
$$\begin{aligned} \bar{C}_{i,j} &= L(P^l_{i,j}) - L(P^l) \\ &= [L(SP(s, i)) + C_{i,j} + L(P^l(j, t))] \\ &\quad - [L(SP(s, j)) + L(P^l(j, t))] \\ &= L(SP(s, i)) + C_{i,j} - L(SP(s, j)) \\ &= \pi_i + C_{i,j} - \pi_j \end{aligned} \tag{1}$$

보조정리 1.1 : $L(P^l_{i,j}) = \bar{C}_{i,j} + L(P^l)$

증명 :
$$\begin{aligned} L(P^l_{i,j}) &= L(SP(s, i)) + C_{i,j} + L(P^l(j, t)) \\ &= \pi_i + C_{i,j} + (L(P^l) - \pi_j) \\ &= (\pi_i + C_{i,j} - \pi_j) + L(P^l) \\ &= \bar{C}_{i,j} + L(P^l) \end{aligned} \tag{2}$$

그러므로 $\bar{C}_{i,j}$는 P^l의 각 지점 $j \in N(P^l)$으로 인접지점 $i \notin N(P^l)$에서 들어오는 우회도로 $(i,j) \notin T_s$를 통과할 때 증가하는 우회경로의 추가되는 우회거리 이므로, 이 $\bar{C}_{i,j}$를 P^l의 각 지점 $j = v^l(m) \in N(P^l)$, $m = 2, \cdots, q_l$에 대해 진입 하는 우회도로 (i,j)별로 구하고, 식 (2)에 의해 $L(P^l{}_{i,j})$을 계산할 수 있다.

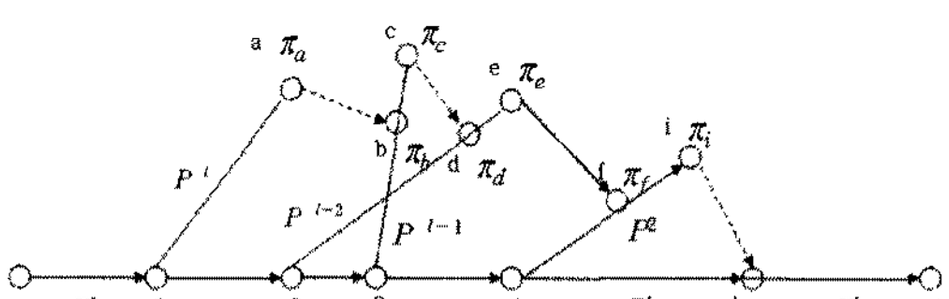
$L(P^l{}_{i,j}) < UB$이면 $P^l_{i,j}$가 P^k를 대신하여 KP에 들어가고 현재의 P^k는 탈락된다.

정리 2 : P^l은 P^1에서 계속 여러 우회도로 $(i,j) \in P^l, (i,j) \notin T_s$들을 거쳐 온 우회경로이면, 우회경로 P^l의 길이는 다음과 같다.

$$L(P^l) = \sum_{\substack{(i,j)\in P^l \\ (i,j)\notin T_s}} \bar{C}_{i,j} + L(P^1)$$

증명 : 만약 P^l이 <그림 1>과 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), $\cdots$, P^2은 P^1에서 우회도로 (g,h)를 거쳐서 나온 우회경로라면, P^l의 길이는 다음과 같다.

$$\begin{aligned} L(P^l) &= \bar{C}_{a,b} + L(P^{l-1}) \\ &= \bar{C}_{a,b} + \bar{C}_{c,d} + L(P^{l-2}) \\ &= \bar{C}_{a,b} + \bar{C}_{c,d} + \bar{C}_{e,f} + L(P^{l-3}) \\ &= \cdots\cdots \\ &= \bar{C}_{a,b} + \bar{C}_{c,d} + \bar{C}_{e,f} + \cdots + \bar{C}_{g,h} + L(P^1) \\ &= \sum_{\substack{(i,j)\in P^l \\ (i,j)\notin T_s}} \bar{C}_{i,j} + L(P^1) \end{aligned} \tag{3}$$



〈그림 1〉 P^l과 P^l이 우회한 이전의 경로들

보조정리 2.1 : P^l이 P^1에서 계속 여러 우회도로 $(i,j) \in P^l, (i,j) \notin T_s$들을 거쳐온 우회도로의 길이의 합 $LD(P^l)$은 $\sum_{\substack{(i,j)\in P^l \\ (i,j)\notin T_s}} \bar{C}_{i,j}$이다.

〈표 1〉 우회경로의 통과 지점들과 길이

경로명	경로	길이
P^1	$s \to 1 \to 2 \to 3 \to 4 \to j \to t$	$L(P^1)$
P^2	$s \to 1 \to 2 \to 3 \to 4 \to i \to j \to t$	$L(P^2) = \bar{c}_{ij} + L(P^1)$
P^3	$s \to 1 \to 2 \to d \to e \to f \to i \to j \to t$	$L(P^{l-2}) = \bar{c}_{ef} + L(P^2)$
P^{l-1}	$s \to 1 \to 2 \to 3 \to c \to d \to e \to f \to i \to j \to t$	$L(P^{l-1}) = \bar{c}_{cd} + L(P^{l-2})$
P^l	$s \to 1 \to a \to b \to c \to d \to e \to f \to i \to j \to t$	$L(P^l) = \bar{c}_{ab} + L(P^{l-1}) = \sum_{\substack{(i,j)\in P^l \\ (i,j)\notin T_s}} \bar{C}_{i,j} + L(P^1)$

증명 : $LD(P^l) = L(P^l) - L(P^1)$

$$\begin{aligned} &= \sum_{\substack{(i,j)\in P^l \\ (i,j)\notin T_s}} \bar{C}_{i,j} + L(P^1) - L(P^1) \\ &= \sum_{\substack{(i,j)\in P^l \\ (i,j)\notin T_s}} \bar{C}_{i,j} \end{aligned} \tag{4}$$

그러므로 P^l의 지점 j으로 인접지점 i에서 연결되어 들어오는 우회경로 $P^l{}_{i,j}$의 길이는 다음과 같이 표현할 수 있다

$$\begin{aligned} L(P^l_{i,j}) &= L(SP(s,i)) + C_{i,j} + L(P^l(j,t)) \\ &= \bar{C}_{i,j} + L(P^l) \\ &= \sum_{\substack{(i,j)\in P^l_{i,j} \\ (i,j)\notin T_s}} \bar{C}_{i,j} + L(P^1) \end{aligned}$$

예를 들어, P^l과 P^1에서 우회하는 경로들이 <그림 1>와 같다면, 우회경로의 통과지점들과 경로길이는 <표 1>과 같이 된다.

그러므로 Dijkstra법으로 구한 T_s의 경로가지의 끝지점이 i이고 이 지점이 t와 연결이 안되어 있고 많이 떨어져 있다면, 우회도로를 하나이상 K-1개까지 거쳐서 P^1이나 t를 만날 수 있으며, <그림 1>와 <표 1>에서처럼 우회거리의 합이 $\sum_{\substack{(i,j)\in P^l \\ (i,j)\notin T_s}} \bar{C}_{i,j}$만큼 증가하게 된다.

그리고 새로운 우회경로의 탐색에서 계산복잡도를 줄이기 위해서 우회경로 탐색 대상이 되는 호와

마디들을 줄이는 방안들을 찾아보고자 한다.

P^1은 모든 마디에 대해 진입호 별로 우회경로를 구하여 $P^2, \cdots, P^K$를 만들어 내었다.

그런데 P^l의 각 지점별로 진입하는 우회도로를 찾을 때, P^l의 경로에서 s를 제외한 각 지점으로의 진입 우회도로들을 찾아서 최단 우회경로를 찾는다. 이 때 P^l이 더 짧은 경로 $P^r, 1 \le r < l-1$에서 우회도로 $(j, v^r(m))$을 거쳐서 지나오는 우회경로이라면, j에서 t까지의 부분경로 $P^l(j,t)$상의 각 지점으로 진입하는 우회도로들은, 경로 P^1에서 P^{l-1}까지를 포함한 KP의 경로들을 찾는 과정에서 점검되었고, 이 과정에서 생긴 우회경로 가운데 길이가 UB보다 작은 경로는 이미 KP에 포함되어 있고, 길이가 UB보다 긴 경로들은 KP에서 탈락되어 있다. KP에 포함 안 된 $P(j,t)$상의 각 지점 별로 진입하는 우회경로는 P^k보다 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 지점 별로 우회경로를 찾고, P^l의 지점 j에서 t까지의 부분경로 $P^l(j,t)$상의 각 지점으로 진입하는 우회도로들은 찾지 않는다.

그러면 새로운 우회경로의 탐색에서 계산복잡도를 줄일 수 있다.

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

정리 3 : 본 해법으로 구한 최종해 $KSP=\{P^1, \cdots, P^l, \cdots, P^k\}$는 최적해이다.

증명 : P^1은 최단 경로이다. 모든 우회경로는 P^1과 s와 t가 같기 때문에 P^1과 하나의 우회도로를 거쳐 만나게 된다. 이 P^1과 하나의 우회도로(호)를 거쳐 만나게 되는 우회경로들 가운데서 가장 짧은 경로가 가장 최단의 우회경로인데 이것이 P^2이다. 안 나타난 경로는 이보다 또 다른 우회도로(호)를 우회하는 경로이므로 더 길다.

또한 P^2에서 P^3을 구할 때 P^2와 단 하나의 우회도로(호)를 거쳐 만나는 경로들이 점검된다. P^1과 P^2의 우회경로들 가운데 가장 짧은 것이 P^3인데, 그 외 안 나타난 경로가 있다면 또 다른 우회도로(호)를 우회하는 경로여서 P^3 보다 길므로, P^3가 P^2 다음으로 짧은 최적의 경로이다.

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, 현재의 P^k를 탈락시키고 이 짧은 경로를 진입시키는 방식으로 KP를 개선시키면서, 긴 경로들은 제거(분지 끝)하고 그 우회경로들도 탐색 고려 대상에서 제외시키는 작업을 반복하여서 구한 $KSP=\{P^1, \cdots, P^l, \cdots, P^k\}$는 K개 최단 경로로 구성된 최적해이다.

본 해법은 최단경로의 우회경로들을 찾는 방안을 사용하므로 KSD(K Shortest Paths Algorithm using Detouring Paths)라고 명명하고, 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 $(i,j^*) \notin T_s$, $j^* \in P^l(v^l(2), j)$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 KP에 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 탐색 전략으로 간다.

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

3. KSD(복수최단경로) 해법

단계 1 : 네트워크 $G(V, E)$ 상에서, T_s를 Dijkstra 법으로 생성한다.

P^1, $L(P^1)$을 구한다.

$l = 1$

단계 2 : 분지

1. P^1의 s를 제외한 각 지점 j별로 진입 우회도로 $(i, j), j \in N(P^1)$, $(i, j) \notin T_s$를 거칠 때의 우회거리 $\overline{C_{i,j}}$를 구한다.
   - $\overline{C}_{ij} = \pi_i + C_{ij} - \pi_j,\ j \in N(P^1)$
2. P^1과 우회경로들의 길이의 오름차순으로 K개의 짧은 경로들을 선정하고 길이를 계산한다.
   - $L(P^1_{i,j}) = L(P^1) + \overline{C}_{ij}$
   - $KP = \{P^1, P^2, P^3, \cdots, P^k\}$
   - $L(P^1), L(P^2), \cdots, L(P^k)$ 계산
3. $UB = L(P^k)$

   $l = l+1$

   $KSP = \{P^1, \cdots, P^l\}$

단계 3 : 탐색

1. $K = 2$이거나 $l = K$이면, STOP.

   최적해 $KSP = \{P^1, \cdots, P^l, \cdots, P^k\}$ 결정

   길이 $L(P^1), L(P^2), \cdots, L(P^k)$ 계산
2. o/w, KP내의 P^l을 선택한다.

단계 4 : 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

   $\overline{C}_{a,i^*} \geq \pi_a + C_{a,i^*} - \pi_{i^*}$ 계산
2. $\overline{C}_{a,i^*} \geq UB - L(P^l)$이면 $l = l+1$, 단계 3로 간다.

   o/w, 우회경로 P^l_{a,i^*}, $(a, i^*) \notin T_s$, $i^* \in P^l$들을 KP에 포함시키고 ,현재의 P^k를 탈락시킨다.
3. 개선해를 작성한다.

   KP 내의 경로들을 오름차순으로 K개 경로들을 정렬시킨다.

   단계 2~3으로 간다.

• KSD 해법의 계산복잡도(Complexity)의 검토

개발된 KSD해법의 각 단계별로 요구되는 계산복잡도는 다음과 같다.

P^1을 Dijkstra법으로 계산하는데는 $O(m)$, 초기해 KP를 구하기 위해 P^1상의 각 지점을 진입하는 우회도로를 점검하여 $\overline{C_{ij}}$를 구하는 것에는 $O(m^* \log n)$, KP내의 $P^2, \cdots, P^k$를 오름차순으로 정리하는 것에 $O(n^* \log n)$, 상기 과정을 $P^2, \cdots, P^{k-1}$에 대한 우회경로를 찾으면서 K-2회 반복하게 되므로, 총 계산복잡도는 다음과 같이 계산되어 $K \cdot O(m \times \log n)$이 된다.

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

4. 사례 계산결과 분석

개발된 KSD 해법을 C언어로 코딩하여, 인텔 Pentium(R) 4 Dual CPU 3.20GHz, 1.00GB RAM의 PC로 실험하였으며, 사용된 사례는 제 9회 DIMACS implementation Challenge에서 제공하는 Benchmark instance를 사용하였다. 이 사례(Instance)는 미국의 New York, Colorado, Florida의 실제 도로 네트워크와 거리 정보와 자료를 사용한 것이다.

사용된 사례는 3가지 네트워크인데 크기는 <표

2>와 같고, 계산결과는 <표 3>과 같다.

〈표 2〉 사례의 네트워크 크기

사 례	마디의 수(n)	호의 수(m)	지 역
1	264,346	733,846	NY
2	435,666	1,057,066	Colorado
3	1,070,376	2,712,798	Florida

〈표 3〉 KSD에 의한 계산결과

CPU Time(단위 : 초)

사례	경유 마디	K = 1	K = 5	K = 10	K = 25	K = 50
1	30	0.188	0.188	0.188	0.188	0.235
	260	0.172	0.187	0.218	0.359	1.265
2	104	0.297	0.313	0.329	0.360	0.453
	1092	0.313	0.437	0.437	0.890	253.9
3	152	0.791	0.828	0.828	0.843	1.047
	1466	0.781	1.031	1.062	2.078	159.8

• KSD 해법의 계산 결과에 관한 분석

사례들을 계산하면서 발견되는 KSD해법의 특징들은 다음과 같다.

제안된 해법은 각 단계에서 K개의 최단경로를 구하기 위해 모든 우회경로들의 우회도로(진입호)에 대해서 우회거리를 구하고, 정렬을 하여 K개의 경로를 구하므로, 각 단계에서 발생되는 우회경로의 개수의 상한이 초기 최단경로의 경유 지점(마디)의 수에 의해 결정된다. 따라서 알고리즘의 수행속도가 최단경로의 경유 지점(마디) 수에 크게 영향을 받는 것을 확인할 수 있었다.

특히 우회경로 K개를 각 단계마다 모두 발생시키므로, K가 50개 이상으로 커질수록 사례의 네트워크상의 특성에 따라서 CPU 시간과 소요되는 데이터 저장크기가 크게 증가할 수도 있겠다.

개발된 KSD 코드의 성능은 K가 20~30개 이하의 일반적인 교통과 수송과 물류 관련의 복수최단경로문제를 위해서는 개발된 해법이 우수한 것을 보여 준다고 사료된다.

또한 최근에 발표된 Martins et al.(2001)의 re-moving path 해법과 비교해 볼 때, 본 해법은 가상 마디와 가상 호를 추가하지 않으므로, 실제 대도시 내의 도로 상에서 각 지점간의 복수최단경로들을 실시간으로 짧은 시간 내에 여러 문의자에게 계산하여 제공하는 경우에 기존 네트워크를 확장할 필요가 없어서 좋은 알고리즘이라 할 수 있다. 특히 기존의 해법들이 매 단계마다 하나의 경로만을 구하여 추가하는 것에 비하여, KSD 해법은 처음부터 K개의 경로들을 구하므로 다수의 문의자에게 가능해를 더 짧은 시간 내에 제공할 수 있는 장점이 있다.

그리고 본 해법에 관한 구체적인 적용 예제에 관심이 있는 분은 장병만 외(2008)의 문헌을 참고하면 도움이 될 것 이다.

5. 결론 및 추후 연구방향

본 연구에서는 K개의 복수최단 경로문제에 대해서 중복 방문이 없는 해를 구하는 $K \cdot O(m^* \log n)$의 KSD 해법을 개발하였고, C 코드를 작성하여 실제 대규모의 도로 네트워크상에서 적용하여 계산한 결과와 분석을 제시하였다.

이 KSD 해법은 전방향의 최단 경로 나무 T_s를 구한 후, 각 도로별로 우회거리 $\bar{C}_{ij}$를 구하여서 K개의 우회 최단 경로들을 구하고, 매 단계마다 더 짧은 우회경로를 찾아 긴 경로를 대체하여 K개 경로의 해를 개선하는 과정을 반복하는 해법으로 분지한계법의 개념을 이용한 최적해법이다.

이 해법은 기존의 해법과는 다르게, K개의 짧은 경로의 해를 처음부터 구할 수 있고, 매 계산과정마다 K개의 가능경로를 구하고 최적해 K개의 경로에 포함되는 경로를 하나씩 추가로 확정해 바꿔 넣으면서 최적해를 찾는 해법으로, 실제 교통과 수송 상황에서 최단시간내에 유용한 해를 구해준다.

또한 실제 대도시 내의 도로 상에서 각 마디간의 복수최단경로들을 기존의 네트워크를 확장하지 않고 실시간으로 짧은 시간 내에 복수의 문의자에게 계산하여 제공하므로 각 마디간 복수최단경로를 구하는데 잘 활용할 수 있겠다.

추후 연구 과제로는 KSD 알고리즘의 실행을 보다 효율적으로 할 수 있으면서 계산 복잡도가 낮은 효율적인 해법으로 보완하는 것과, 실제 도로망이나 교통·수배송 등 활용 영역에서의 검증이 더 필요하다. 또한 ITS 등과 같은 실제 활용 영역에서의 특성상 요구되는 모든 마디간의 복수 최단 경로 문제 해결을 위한 DB 구조에 대한 연구가 요망된다.

참 고 문 헌

[1] 장병만, 김시곤, "유방향의 복수최단경로 해법에 관한 연구", 「한국SCM학회지」, 제8권, 제1호(2008), pp.83-92.

[2] 장병만, 민윤홍, "복수최단경로 새로운 해법 연구", IE/MS 춘계공동학술대회 발표집(2008).

[3] Dijkstra, E.W., "A note on two problems in connection with graphs," *Numerische Mathematik*, Vol.1(1959), pp.269-271.

[4] Dreyfus, S., "An appraisal of some shortest path algorithms," *Oper Res*, Vol.17, No.2 (1969), pp.395-412.

[5] Eppstein, D., "Finding the K shortest paths," *SIAM J.Comput.*, Vol.28, No.2(1998), pp. 652-673.

[6] Hadjiconstantinou, E. and N. Christofides, "An efficient implementation of an algorithm for finding K shortest paths," *Networks*, Vol.34(1999), pp.88-101.

[7] Katoh, N., T. Ibaraki, and H. Mine, "An efficient algorithm for K shortest simple paths," *Networks*, Vol.12(1982), pp.411-427.

[8] Lawler, E., "A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem," *Management Sci.*, Vol.18 (1972), pp.401-405.

[9] Lawler, E., *Combinatorial Optimization : Networks and Matroids*, Holt Reinhart and Winston, New York, (1976), pp.100-104.

[10] Martins, E.Q., M. Pascoal, and J.L. Santos, "A new improvement for a K shortest paths algorithm," Investigacao Operacional, Vol. 21, No.1(2001), pp.47-60.

[11] Shier, D., "On algorithm for finding the K shortest paths in a network," *Networks*, Vol.9(1979), pp.195-214.

[12] Yen, J., "Finding the K shortest loopless paths in a network," *Management Sci.*, Vol. 17(1971), pp.712-716.