도로 네트워크에서의 Some-to-some 최단경로 생성 알고리듬 성능비교
A Comparison of Algorithms for Computing Some-to-some Shortest Path on Street Networks

김병인, 정상원

포항공과대학교 산업경영공학과 (bkim, pheonix@postech.ac.kr)

Abstract

본 논문은 차량운행경로문제(Vehicle Routing Problem)의 기본적인 정보가 되는 Origin-Destination (OD) Matrix 의 생성에 관한 문제를 다룬다. OD Matrix 를 만들기 위해서는 몇 개의 지점으로부터 몇 개의 지점(some-to-some)까지의 최단 거리를 계산해야 한다. 이를 위해 Dijkstra 알고리듬과 같은 one-to-all 알고리듬을 반복하여 사용할 수 있고 Floyd-Washall 알고리듬과 같은 all-to-all 알고리듬을 사용할 수 있으며 some-to-some 을 위해 고안된 알고리듬을 사용할 수 있다. 본 연구에서는 이러한 알고리듬들이 실제 도로 네트워크의 OD Matrix 를 생성할 때 어떤 성능을 보이는지 비교 분석한다.

1. 연구개요

최단경로문제(shortest path problem)는 네트워크상에서 임의의 두 지점을 잇는 최단경로를 찾는 것으로 다양한 응용분야를 갖고 있다. 지금까지 많은 연구자들에 의해 이를 위한 알고리듬들이 개발되어 왔는데 구하고자 하는 최단경로가 몇 개의 지점에서 몇 개의 지점까지이냐에 따라 크게 한 지점에서 모든 지점까지(one-to-all) 알고리듬과 모든 지점에서 모든 지점까지(all-to-all) 알고리듬 으로 분류된다. 현재 one-to-all 문제의 경우 Dijkstra 알고리듬(Dijkstra, 1959)이 가장 효율적인 것으로 알려져 있으며 계산속도를 향상시키기 위해 여러 가지 데이터 구조들이 시도되고 있다. 또한 one-to-one 문제를 위해 Dijkstra 알고리듬을 변형한 A^* 알고리즘이 있다. All-to-all 문제의 경우에는 Floyd-Warshall 알고리듬(Floyd, 1962)이 가장 효율적인 것으로 알려져 있다.

본 연구에서 고찰하고자 하는 것은 차량운행경로문제(Vehicle Routing Problem: VRP)나 외판원 문제(Traveling Salesman Problem) 의 기본적인 정보가 되는 Origin-Destination (OD) Matrix의 생성에 관한 문제를 다룬다. 실제 문제에서의 OD Matrix생성에 관한 사례의 예는 Weigel과 Cao(1999), Sahoo 등(2005)에서 찾을 수 있다. OD matrix는 그림 1에서와 같이 교차로 (intersection)와 도로(street)로 구성된 네트워크상에서 지점과 지점의 거리정보를 나타내는 것으로 어떤 고객의 위치에서 다른 고객의 위치까지 이동하는 데 소요되는 시간이 얼마인가에 대한 정보를 갖는다. 이 정보는 VRP문제의 기본적인 입력 데이터가 된다. OD Matrix를 만들기 위해서는 몇 개의 지점으로부터 몇 개의 지점까지 즉, some-to-some의 최단 거리를 계산해야 한다. 그림 1의 경우, 전체 네트워크는 14개의 도로 끝 지점, 11개의 교차점, 6개의 고객지점 등 31개의 node가 있다. 즉, one-to-all이나 all-to-all 알고리즘의 경우 31이 all에 해당하는 숫자이다. 하지만 VRP에서 필요한 OD matrix는 31개의 node에서 31개의 node간의 정보가 아니라 6개의 node에서 6개의 node 간의 거리 정보이다.

OD Matrix 를 생성하기 위해 one-to-all 알고리듬을 반복하여 사용할 수 있고 all-to-all 알고리듬을 사용할 수 있으며, some-to-some 을 위해 고안된 알고리듬을 사용할 수 있다. 본 연구에서는 이러한 알고리듬들이 실제 도로 네트워크의 OD Matrix 를 생성할 때 어떤 성능을 보이는지 비교 분석한다.

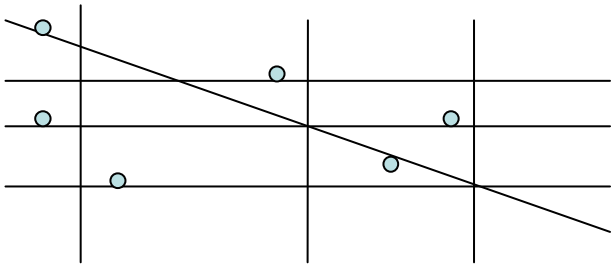

그림 1. Some-to-some 최단경로 생성문제

본 논문은 다음과 같이 구성된다. 2절에서 최단경로 알고리듬에 대한 기존연구를 고찰하고 3절에서는 실제 도로 네트워크를 이용한 각 알고리듬들의 성능을 비교한다. 4절에서는

요약정리하며 결론을 맺는다.

2. 최단경로 알고리듬 고찰

Cherkassky 등 (1996)은 one-to-all 최단경로 알고리듬들의 복잡도(complexity)를 분석하고 가상적인 네트워크에 적용한 결과를 비교 분석하였다. 여러 알고리듬들 중 double bucket 데이터 구조를 이용한 Dijkstra 알고리듬의 성능이 우수하였다. 또한 도로네트워크와 같이 특별한 구조를 갖는 문제에서는 Pallottino (1984)의 TWO-Q 알고리듬이 로버스트하고 우수한 결과를 나타내었다.

Zhan 과 Noon (1998)은 다양한 실제 도로 네트워크상에서 기존의 one-to-all 최단경로 알고리듬들의 성능을 비교하였다. 비교한 15 개의 알고리듬 중 가장 우수한 성능을 보인 알고리듬은 Pallottino (1984)의 TWO-Q 알고리듬이다. 여기서 성능이 우수하다는 것은 계산 시간이 적게 걸린다는 것을 의미한다. 또한 one-to-one 이나 one-to-some 에서는 Bucket 데이터 구조를 이용한 Dijkstra 알고리듬(DIKAB: Dijkstra's algorithm implemented with approximate buckets)을 추천하였다. Bellman-Ford 알고리듬 과 Dijkstra 알고리듬은 교과서에서 가장 널리 소개되는 알고리듬들이지만, 데이터구조를 고려하지 않은 단순한 (naïve) 구현은 15 개의 알고리듬들 중 가장 좋지 않은 결과를 보였다.

Zhan 과 Noon (2000)은 Zhan 과 Noon (1998)의 one-to-all 문제에 가장 우수한 성능을 보인 TWO-Q 알고리듬과 DIKAB 알고리듬이 one-to-one 문제에 어떤 성능을 보이는지를 실험하였다. 실험결과 출발지에서 비교적 가까이에 있는 목적지까지의 최단 경로를 찾는 데는 DIKAB 가 우수하고 그 외는 TWO-Q 가 우수하였다. 경계가 되는 거리는 출발지에서 가장 먼 거리 지점까지의 최단거리의 40% 거리였다.

지리 정보 시스템(Geographic Information System), 위성 위치 추적 시스템(Global Positioning System) 및 이들을 이용한 네비게이션 시스템의 발달과 함께 one-to-one 최단경로문제에 대한 연구는 활발히 진행되고 있다. Goldberg 와 Harrelson (2005), Sanders 와 Schultes (2005), Schulz 등 (2002), Ikeda 등 (1994) 의 연구가 그 중에 포함된다.

Wang 등(2005)은 some-to-some 최단경로문제를 위하여 행렬(matrix) 연산에 근거한 분할(decomposition) 알고리듬을 제안하였다. 제안된 분할 알고리듬 DLU 는 Carre(1971)가 all-to-all 최단경로 문제를 위하여 제안한 분할 알고리듬에 기반한다. 제안된 알고리듬을 Cherkassky 등 (1996)의 가상적인 네트워크의 some-to-some 최단경로 문제에 적용한 결과 TWO-Q 등과 같은 label-correcting 알고리듬에는 성능이 미치지 못하였다. 하지만 실제 항공 네트워크문제에 적용하였을 때는 제안된 알고리듬의 계산시간이 적게 소요됨을 보였다. 그들이 사용한 항공 네트워크 문제는 arc/node 비율이 3 에서 6 사이의 값을 가지고 node 수는 130 에서 1093 사이의 값을 가진다. 네트워크의 node 수를 N 이라고 할 때 그들이 실험한 출발지점/도착지점 수는 N 인 경우와 0.5N 경우였다. 즉 N 개의 지점에서 N 개의 지점까지의 최단경로를 구하는 경우와 0.5N 개의 지점에서 0.5N 개의 지점까지의 최단경로를 구하는 경우이다.

Wang 등(2005)이 고려한 some-to-some 문제는 본 연구에서 고려하는 some-to-some 과는 의미하는 바가 다르다. Wang 등(2005)에서는 출발지(origin)와 도착지(destination) 쌍(pair)의 수가 N 혹은 0.5N 으로 주어지고, 본 연구에서는 출발지와 도착지 지점이 주어지며 출발지들과 도착지들의 모든 쌍 간의 최단경로를 구하는 것을 고려한다. 예를 들어, 출발지 수와 도착지 수가 0.5N 으로 주어졌다면 Wang 등(2005)에서는 0.5N 쌍의 최단경로를 찾는 것이 문제이지만 본 연구에서는 0.5N X 0.5N, 즉, $0.25N^2$ 쌍 간의 최단경로 거리를 계산하여야 한다.

3. 실험결과

본 연구에서는 실제 도로 네트워크에서 OD matrix를 생성시키고자 할 때 어떤 알고리듬을 사용하는 것이 효율적인가에 대한 성능비교를 한다. 고려하는 알고리듬은 기존의 연구로부터 도로 네트워크에서 뛰어난 성능을 보인 Pallottino(1984)의 TWO-Q 알고리듬과 Cherkassky 등 (1996)의 DIKAB 알고리듬, 그리고 Binary Heap 데이터 구조를 이용한 Dijkstra 알고리듬(DIKBH) 등의 one-to-all 알고리듬 들과 all-to-all 알고리듬인 Floyd-Warshall FW 알고리듬 (Floyd, 1962, Radin, 1998), one-to-one 알고리듬인 A^* 알고리듬 (Goldberg와 Harrelson, 2005), 그리고 some-to-some 알고리듬인 Wang 등 (2005)의 DLU 알고리듬이다. A^* 알고리듬에서 lower bound를 위해서 직선거리(Euclidean Distance)와 최대주행속도가 사용되며 데이터 구조로는

Binary Heap이 사용된다.

실험에 사용된 도로네트워크는 지리 정보 시스템으로부터 얻어진 실제 도로네트워크이고 사용한 데이터 셀의 특징을 표 1에서 보인다. 실험에 사용된 네트워크는 Node수 611개에서부터 688,489개까지의 다양하다. 그림 1에서 Network 3의 도로구조를 예로 보인다.

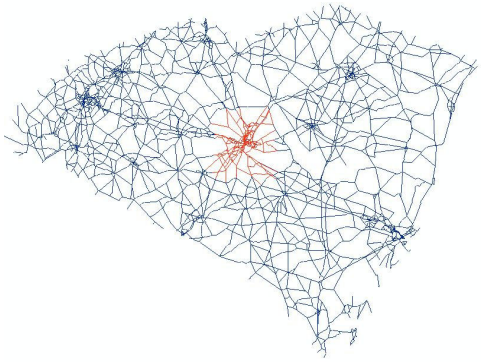

그림 1. Network3의 도로 네트워크

각각의 데이터 셀에 대해 출발지점의 수와 도착지점의 수를 데이터 셀 node수의 0.1%, 0.5%, 1%, 5%, 10%, 20%, 50%, 100% 개수를 갖는 some-to-some 문제를 고려한다. 따라서 100%인 경우 all-to-all 최단경로 문제가 된다. 출발지점과 도착지점은 node들 중에서 중복을 피하여 무작위로 선택하도록 하였다. 여기서 출발지점 및 도착지점을 stop으로 표현한다. 즉, stop은 node의 부분집합이 된다.

최단경로를 위해 도로 네트워크를 컴퓨터상에 표현 (representation) 하는 데이터 구조로 forward star 가 가장 효율적인 것으로 알려져 있기 때문에 (Cherkassky 등, 1996) 본 연구에서도 그 구조를 사용한다. Forward star에서는 두 개의 array가 사용되는데, 첫 번째 array는 arc정보를 저장하고 두 번째 array에서는 node정보를 보관한다. 한 node에서 출발하는 arc들은 arc array에서 서로 이웃하도록 저장한다. Node array 에서는 그 node에서 출발하는 arc의 첫 번째 index를 보관하게 된다. 그 node에서 출발하는 arc들은 자신의 첫 번째 arc index부터 다음 node의 첫 번째 arc index 에서 1 적은 index까지의 arc 들이다.

알고리듬들은 C언어로 구현 되었으며 Microsoft사의 Visual C++ 6.0 컴파일러로 컴파일 되었다. 실험은 Windows XP를 운영체제로 하는 Intel Xeon CPU 3.4 GHz, 3.25GB RAM PC에서 실시되었다.

표2-표7에서 실험결과를 보인다. 네트워크가 큰 문제의 경우 Floyd-Warshall (FW)이나 DLU알고리듬은 과다한 컴퓨터 메모리를 요구하여 계산을 수행하지 못했다. 또한 A*의 경우에도 과다한 계산 시간이 요구되어 실험을 수행하지 않았다. 수행하지 않은 실험은 표에서 빈칸으로 남겨져 있다.

Node수가 작은 Network1과 Network2에서는 label correcting 알고리듬인 TWO-Q 알고리듬이 가장 우수한 결과를 나타내었으나 Node수가 6300개 이상인 네트워크들에서는 label setting 알고리듬인 DIKAB가 가장 우수한 결과를 보였다. 이 결과는 Zhan 과 Noon (1998)의 결과와 일치하지 않는다. Zhan 과 Noon (1998)에서는 도로 네트워크에서의 one-to-all에서 TWO-Q가 가장 좋은 결과를 보인다고 하였으나, 본 연구에서는 네트워크 크기가 커질 때는 Two-Q보다 DIKAB가 오히려 효율적임을 알 수 있었다.

네트워크의 크기가 커짐에 따라 one-to-all 알고리듬이 한번 one-to-all 최단경로를 계산하는 데 소요되는 시간의 추이를 보고자 각 표에서 마지막 열을 이용하여 평균시간을 계산하였다. 표 8이 그 결과를 보인다. 예를 들어, Network1에서의 DIKAB의 평균시간은 표2의 마지막 열 0.39초를 stop수 612로 나눈 값이다. 표 8을 그래프로 표현한 것이 그림 2이다. 그림에서 보는 바와 같이 네트워크 크기가 증가함에 따라 one-to-all 알고리듬의 계산시간이 증가하게 되는데 그 증가의 정도가 TWO-Q가 가장 크고 DIKAB가 가장 작음을 알 수 있다. 따라서 큰 크기의 도로 네트워크에서는 TWO-Q보다 DIKAB가 성능이 좋다는 것을 알 수 있다.

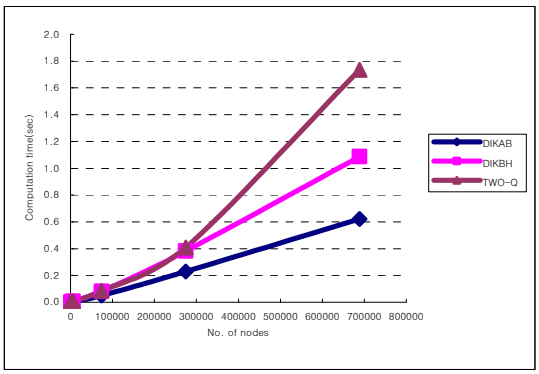

그림 2. 네트워크 크기변화에 따른 one-to-all 알고리듬 계산시간 변화 추이

또한 some-to-some으로 개발된 DLU 알고리듬의 성능은 작은 네트워크에서 작은 수의 OD 쌍을 구할 경우에만 Floyd-Warshall (FW) all-to-all 알고리듬보다 좋은 성능을 보일 뿐,

일반적인 some-to-some 문제에서는 좋지 않았다. All-to-all 알고리듬은 some-to-some 문제에서 TWO-Q, DIKAB 등에 미치지 못하는 성능을 보였다. 더 나아가 all-to-all 문제(예. 표2와 표3의 마지막 열)에서도 one-to-all 알고리듬을 반복하여 사용하는 것이 FW 알고리듬보다 계산시간이 빠름을 확인할 수 있었다. One-to-one 알고리듬인 A* 알고리듬은 계산시간 측면에서 매우 좋지 않은 결과를 내었다.

표1. 도로네트워크 데이터 셑

Data Set	Node 수	Arc 수	Arc/Node 비율	비 고
Network1	612	1685	2.76	Columbia, SC highway
Network2	3285	7755	2.36	LA Suburb
Network3	6305	17315	2.75	South Carolina highway
Network4	74180	167686	2.26	Long Islands Area
Network5	274464	600142	2.19	Atlanta Area
Network6	688489	1578980	2.29	Houston Area

표2. Network1상의 OD Matrix를 위한 알고리듬의 계산시간(초)

Stop 수	OD 쌍 수	DIKAB	DIKBH	TWO-Q	A*	FW	DLU
3	9	0.00	0.00	0.00	0.00	1.53	0.58
6	36	0.00	0.00	0.00	0.02	1.53	0.61
30	900	0.03	0.03	0.02	0.31	1.53	1.83
61	3721	0.03	0.03	0.02	1.38	1.55	5.11
122	14884	0.08	0.03	0.03	5.34	1.53	21.61
306	93636	0.11	0.09	0.12	33.53	1.55	120.80
612	374544	0.39	0.31	0.22	142.73	1.55	505.49

표3. Network2상의 OD Matrix를 위한 알고리듬의 계산시간(초)

Stop 수	OD 쌍 수	DIKAB	DIKBH	TWO-Q	A*	FW	DLU
3	9	0.00	0.03	0.00	0.03	250.22	86.36
16	256	0.05	0.06	0.02	2.80	250.11	104.22
32	1024	0.05	0.08	0.06	87.97	250.19	141.25
164	26896	0.30	0.45	0.27	336.72	250.27	1504.50
328	107584	0.59	0.89	0.61	1340.58	256.45	5887.13
657	431649	1.25	1.85	1.15	8415.06	250.41	
1642	2696164	3.30	4.72	3.10		250.13	
3285	10791225	7.65	11.04	7.06		250.53	

표4. Network3상의 OD Matrix를 위한 알고리듬의 계산시간(초)

Stop 수	OD 쌍 수	DIKAB	DIKBH	TWO-Q	A*	FW	DLU
6	36	0.02	0.03	0.03	0.16	1743.28	613.52
31	961	0.13	0.19	0.13	3.64	1731.11	869.48
63	3969	0.22	0.38	0.27	17.19	1720.86	1537.53
315	99225	1.02	1.78	1.36	438.71	1721.50	23129.41
630	396900	2.21	3.61	2.58		1720.95	
1261	1590121	4.67	7.68	5.53		1721.00	
3152	9935104	13.63	20.12	15.78		1720.94	

표5. Network4상의 OD Matrix를 위한 알고리듬의 계산시간(초)

Stop 수	OD 쌍 수	DIKAB	DIKBH	TWO-Q	A*
74	5476	3.53	5.81	6.55	736.08
370	136900	18.30	29.17	31.52	
741	549081	35.02	58.54	61.86	
3709	13756681	180.66	297.19	311.60	
7418	55026724	375.29	604.83	644.08	

표6. Network5상의 OD Matrix를 위한 알고리듬의 계산시간(초)

Stop 수	OD 쌍 수	DIKAB	DIKBH	TWO-Q	A*
274	75076	63.03	104.39	110.58	20852.80
1372	1882384	314.80	522.91	568.24	
2744	7529536	628.53	1047.09	1119.78	

표7. Network6상의 OD Matrix를 위한 알고리듬의 계산시간(초)

Stop 수	OD 쌍 수	DIKAB	DIKBH	TWO-Q
688	473344	425.91	725.75	1181.19
3442	11847364	2138.65	3865.65	5974.53
6884	47389456	4282.73	7491.48	11938.83

표8. one-to-all 최단경로 평균계산시간(초)

Data Set	Node 수	DIKAB	DIKBH	TWO-Q
Network1	612	0.00064	0.00051	0.00036
Network2	3285	0.00233	0.00336	0.00215
Network3	6305	0.00432	0.00638	0.00501
Network4	74180	0.00432	0.00638	0.00501
Network5	274464	0.22906	0.38159	0.40808
Network6	688489	0.62213	1.08825	1.73429

4. 결론

본 연구에서는 차량운행경로문제(Vehicle Routing Problem)의 기본적인 정보가 되는 Origin-Destination (OD) Matrix 의 생성에 사용되는 some-to-some 최단경로문제에 대해 one-to-all, one-to-one, all-to-all, some-to-some 알고리듬들이 어떤 성능을 보이는지에 대해 고찰하였다.

실험결과 node 수가 6300 개 이하인 작은 크기의 도로네트워크에서는 label correcting 알고리듬인 TWO-Q 알고리듬이 우수하였고 큰 도로네트워크에서는 label setting 알고리듬인 DIKAB 가 우수하였다. 네트워크 크기가 증가함에 따라 늘어나는 계산시간의 비율에서 DIKAB 가 TWO-Q 보다 우수하기 때문에 이런 현상이 일어남을 본 연구에서 확인하였다. 또한 all-to-all 최단경로 문제에서도 도로 네트워크에서는 일반적으로 효율적일 것이라고 알려진 Floyd-Warshall 알고리듬보다도 one-to-all 알고리듬을 반복하여 사용하는 것이 훨씬 효과적임을 알 수 있었다. 또한 네트워크가 큰 문제의 경우 Floyd-Warshall 이나 Wang 등(2005)의 some-to-some 알고리듬은 과다한 컴퓨터 메모리를 요구하기 때문에 계산을 수행 할 수 없는 경우가 발생하기도 하였다.

본 연구에서는 각기 다른 목적으로 개발된 기존의 여러 알고리듬들을 some-to-some 최단경로 문제에 적용하여 성능을 비교하였다. 비교한 알고리듬 중 가장 빠른 알고리듬도 큰 크기의 네트워크에서는 1시간 이상의 계산 시간을 필요로 한다. 향후 연구방향으로 도로 네트워크의 특성과 some-to-some 문제의 특성을 이용한 보다 효과적인 알고리듬의 개발을 고려할 수 있다.

참고문헌

[1] Carre, B. A., "An Algebra for Network Routing Problems," *IMA Journal of Applied Mathematics*, Vol. 7 (1971), pp. 273-294

[2] Cherkassky, B. V., Goldberg, A. V., and Radzik, T., "Shortest Paths Algorithms: Theory and Experimental Evaluations," Mathematical Programming, Vol. 73 (1996), pp. 129-174

[3] Floyd, L. R., Jr., "Algorithm 97, shortest path," *Communications of the ACM*, Vol. 5, No. 6 (1962), pp. 345.

[4] Goldberg, A. V. and Harrelson, C., "Computing the Shortest Path: A* Search Meets Graph Theory," *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, Vancouver, British Columbia, pp. 156-165, 2005.

[5] Ikeda, T., Hsu, M.-Y., and Imai, H., "A Fast Algorithm for Finding Better Routes by AI Search Techniques," *Proceedings of Vehicle Navigation and Information Systems Conference*, pp. 291-296, 1994.

[6] Pallottino, S., "Shortest-Path Methods: Complexity, Interrelations, and New Propositions," *Networks*, Vol. 14 (1984), pp. 257-267.

[7] Radin, R. L., *Optimization in Operations Research*, Prentice Hall: New Jersey, 1998.

[8] Sahoo S., Kim, S., Kim, B.-I., Kraas, B., and Popov, A., Jr., "Routing Optimization for Waste Management," *Interfaces*, Vol. 35, No. 1 (2005), pp. 24-36.

[9] Sanders, P. and Schultes, D., "Highway Hierarchies Hasten Exact Shortest Path Queries," *ESA 2005, Lecture Notes in Computer Sciences (LNCS)* 3669 (2005), pp. 568-579.

[10] Schulz, F., Wagner, D., and Zaroliagis, C., "Using Multi-level Graphs for Timetable Information in Railway Systems," ALENEX, *Lecture Notes in Computer Sciences (LNCS)*, 2409 (2002), pp. 43-59.

[11] Wang, I-L., Johnson, E. L., and Sokol, J. S., "A Multiple Pairs Shortest Path Algorithm," *Transportation Science*, Vol. 39, No. 4(2005), pp. 465-476.

[12] Weigel, D., and Cao, B., "Applying GIS and OR techniques to solve Sears technician-dispatching and home-delivery problems," *Interfaces*, Vol. 29, No.1(1999), pp. 112-130.

[13] Zhan, F. B. and Noon, C. E., "Shortest Path Algorithms: An Evaluation using Real Road Networks," *Transportation Science*, Vol. 32, No. 1 (1998), pp. 65-73.

[14] Zhan, F. B. and Noon, C. E., "A Comparison Between Label-Setting and Label-Correcting Algorithms for Computing One-to-One Shortest Paths," *Journal of Geographic Information and Decision Analysis*, Vol. 4, No. 2 (2000), pp. 1-13.