퍼지 및 지능시스템학회 논문지 2007, Vol. 17, No. 7, pp. 893-900

방향그래프의 점대점 최단경로 탐색 알고리즘

A Point-to-Point Shortest Path Search Algorithm for Digraph

이상운

Sang-Un, Lee

강릉대학교 컴퓨터정보공학부 멀티미디어정보공학 전공

요 약

본 논문은 실시간 GPS 항법시스템에서 최단 경로를 탐색하는데 일반적으로 적용되고 있는 Dijkstra 알고리즘의 문제점을 개선한 알고리즘을 제안하였다. Dijkstra 알고리즘은 출발 노드부터 시작하여 그래프의 모든 노드에 대한 최단 경로를 결정하기 때문에 일반적으로 노드의 수 - 1회를 수행해야 하며, 알고리즘 수행에 많은 메모리가 요구된다. 따라서 Dijkstra 알고리즘은 복잡한 도시의 도로에서 목적지 까지 최단 경로를 탐색하여 실시간으로 정보를 제공하지 못할 수도 있다. 이러한 문제점을 해결하고자, 본 논문에서는 먼저 출발과 목적지 노드를 제외한 경로 노드들의 최단 경로 (유입과 유출 최소 가중치 호 선택)를 결정하고, 출발 노드부터 시작하여 노드 유출 호들에 대해 경로 노드의 최단 경로와 일치하는 호들을 모두 선택하는 방식으로 한번에 다수의 노드들을 탐색하는 방법을 택하였다. 14개의 다양한 방향 그래프에 제안된 알고리즘을 적용한 결과 모두 최단 경로를 탐색하는데 성공하였다. 또한, 수행 속도 측면에서 Dijkstra 알고리즘보다 2배에서 3배 정도 빠른 결과를 얻었으며, 알고리즘 수행에 필요한 메모리도 적게 요구되었다.

키워드 : 방향 그래프, 최단 경로, Dijkstra 알고리즘, 최소 가중치 호, 점대점 최단경로 탐색

Abstract

This paper suggests an algorithm that improves the disadvantages of the Dijkstra algorithm that is commonly used in GPS navigation system, searching for the shortest path. Dijkstra algorithm, first of all, requires much memory for the performance of the algorithm. It has to carry out number of node minus 1, since it determines the shortest path from all the nodes in the graph, starting from the first node. Therefore, Dijkstra algorithm might not be able to provide the information on every second, searching for the shortest path between the roads of the congested city and the destination. In order to solve these problems, this paper chooses a method of searching a number of nodes at once by means of choosing the shortest path of all the path nodes (select of minimum weight arc in-degree and out-degree), excluding the departure and destination nodes, and of choosing all the arcs that coincide with the shortest path of the path nodes, from all the node outgoing arcs starting from the departure node. On applying the suggested algorithm to 14 various digraphs, we succeeded to search the shortest path. In addition, the result was obtained at the speed of 2 to 3 times faster than that of Dijkstra algorithm, and the memory required was less than that of Dijkstra algorithm.

Key words : Digraph or Directed Graph, Shortest Path, (Dijkstra's Algorithm, Minimum-weight Arc, Point-to-Point Shortest Path Search

1. 서 론

그래프(Graph)는 정점들(Vertices)과 간선들(Edges)로 구성되어 있으며, 정점들이 간선들로 연결되어 있고 (Connected), 간선들은 방향성(Directed)과 무방향성(Undirected)으로 구분된다. 또한 간선들은 가중치가 있는 경우(Weighted)와 없는 경우(Unweighted)로 구분된다. 무방향 그래프(Undirected Graph)는 $G=(V,E)$로 표기하며, 방향 그래프(Directed Graph)는 $G=(N,A)$는 로 표기한다. 노드(Nodes, N)는 정점 (Vertices, V)이라 하며, 호(Arcs, A)는 간선(Edges, E)또는 화살표(Arrows)라고도 한다. 방향 그래프는 최소 신장트리(MST, Minimum Spanning Tree), 최단 경로(Shortest Path)와 임계경로(CP, Critical Path) 등을 구하는데 활용된다. 본 논문은 최단 경로에 초점을 맞춘다. 최단경로를 구하는 알고리즘은 최근 들어 실시간 GPS 항법 시스템(Real Time GPS Navigation System)에 일반적으로 적용되고 있다[1]. 실시간 GPS 항법 시스템은 운전자에게 자신의 현재 위치를 화면상에 지정하고, 목적지를 결정하면 최단 경로(시간 또는 거리)를 결정하여 화면상에 표시해주고 운전자가 길을 따라 가도록 유도하는 시스템이다. 최단 경로를 찾는 알고리즘은 Dijkstra, Bellman Ford, Topological Ordering과 A-Star(A*)가 있다[1,2]. Dijkstra SP 알고리즘은 양의 가중치를 갖는 호들로 구성된 방향 그래프의 단일 출발점 최단 경로(Single-Source

접수일자 : 2007년 11월 15일
완료일자 : 2007년 12월 1일
본 논문은 2007년도 강릉대학교 학술연구조성비 지원에 의해 수행되었음."

Shortest Path)를 찾는 알고리즘이다[3-5]. GPS 시스템의 최단경로는 단일 출발점과 단일 목적지의 최단 경로를 찾는 점대점(P2P, Point-to-Point) 최단경로 알고리즘으로 Dijkstra SP 알고리즘의 특별한 경우이다. 점대점 최단경로 탐색 문제도 Dijkstra SP 알고리즘에 기반을 두고 있다[1,6,7].

GPS 최단거리 탐색 알고리즘에서, 그래프의 크기에 따라 Bellman Ford 알고리즘은 Dijkstra 알고리즘보다 많은 시간이 소요되지만 좋은 결과를 얻을 수 있다. Topological Ordering 알고리즘은 선형 시간대에 최단 경로를 찾을 수 있지만 최적의 경로를 찾지 못할 수 도 있다. A* 알고리즘은 Dijkstra 알고리즘과 같이 좋은 결과를 얻으며 메모리를 적게 요구되지만 최단 경로를 구하는 알고리즘을 구현하기가 보다 어렵다. 이와 같은 이유로 인해 최근 들어 GPS 시스템에는 Dijkstra SP 알고리즘이 가장 널리 활용되고 있다[1]. 그러나 Dijkstra SP 알고리즘은 알고리즘 수행에 많은 메모리를 요구하며, 일반적으로 "노드의 수 -1" 번을 수행해야 하기 때문에 그래프의 크기가 클 경우 실시간으로 정보를 제공하기가 어렵다.

본 논문은 그래프의 크기가 클 경우라도 적은 메모리 용량을 필요로 하면서도 간단한 수행 횟수로 최단 경로를 찾는 알고리즘을 제안한다. 2장에서는 Dijkstra SP 알고리즘의 최단 경로를 찾는 과정을 고찰해 보고 문제점을 살펴본다. 3장에서는 Dijkstra SP 알고리즘의 단점을 보완한 새로운 최단경로 탐색 알고리즘을 제안한다. 4장에서는 제안된 알고리즘을 다양한 방향 그래프에 적용하여 성능을 평가해 본다.

2. 관련 연구와 연구 배경

2.1 최단거리 탐색 알고리즘

무방향 그래프 $G=(V,E)$의 간선은 $e=\{x, y\}$로, 방향 그래프 $G=(N,A)$의 호는 방향성이므로 순서쌍 $a=(x, y)$로 표기한다. 여기서, x는 꼬리(Tail), y는 머리(Head)라 하며, 노드 n은 유출 차수와 유입 차수를 갖는다. 또한 간선과 호는 가중치를 갖고 있다. 방향그래프의 최단거리를 찾는 대표적인 알고리즘인 Dijkstra SP 알고리즘[5]은 단일 출발 노드 최단 경로를 찾는 알고리즘으로 그림 1과 같다.

Dijkstra SP 알고리즘은 "특정 노드로부터 시작하여 최소 경로 길이를 갖는 노드를 한번에 하나씩 선택하는 방식"으로, 특정 노드에 인접한 노드들과 이전에 계산된 노드들의 경로의 합이 최소가 되는 노드를 찾는다. 따라서 출발 노드로부터 시작하여 모든 노드들을 방문하는 최단 경로를 찾기 때문에 $n-1$번 알고리즘이 수행된다. 이는 Prim의 최소신장트리(MST, Minimum Spanning Tree)를 찾는 알고리즘과 유사한 방법이다. 참고로, Prim MST 알고리즘[8]은 무방향 그래프에서 모든 정점을 연결하는 최소 신장트리를 구성하는 것으로, 임의의 정점을 선택하고, 이에 연결된 간선들 중에서 최소 가중치 간선(MWE, Minimum-Weight Edge)을 가진 정점을 선택한다. 새로 선택된 정점의 간선들 가중치와 기존에 방문하였지만 선택되지 않은 간선들 중에서 MWE를 선택한다(단, 이 과정에서 사이클이 발생하는 간선은 무시한다). 모든 정점들이 선택될 때까지 이 과정을 반복적으로 수행한다.

2.2 알고리즘 적용 문제점과 연구 배경

그림 2의 G_1 그래프에 대해 Dijkstra SP 알고리즘[5]을 적용하여 보자. G_1 그래프는 Llewellyn[9]에서 인용되었으며, $n=10$, $a=14$인 그래프로 출발 노드는 ⓓ, 목적지 노드는 ⓖ이다. 출발 노드에서 목적지 노드까지의 최단거리를 갖는 경로를 Dijkstra SP 알고리즘으로 찾는 과정은 표 1에 제시되어 있다.

≪Dijkstra SP Algorithm ≫

방향 그래프 $D=(N,A)$와 가중치 함수 $w:A\rightarrow\vec{N}$에서, 출발 노드를 $s\in A$, $l(s)=0$로 설정하고 P에 저장

$x\neq s$인 모든 노드 $x\in N$에 대해 $l(x)=\infty$로 설정하고 Q에 저장

① Q에 있고, s에 인접한 모든 노드 $y\in A$ /* s의 유출 호

(1) $l(y)=\min[l(y), l(s)+w(s,y)]$로 치환(단, 동일한 경로 경우 $l(s)$ 호 선택, $l(s)+w(s,y)$ 경로 길이가 작을 경우 (s,y) 선택)

(2) $l(n_1)\leq l(y)$인 노드 n_1 선택(단, 최소 경로 길이 노드가 다수 존재시 하나만 선택)

(3) 호 (s,n_1)을 PSP (Partial Spanning Tree)에 저장, n_1을 Q에서 삭제하고 P에 저장

n_i, $i=1,2,\cdots$

② Q에 있고, n_i에 인접한 모든 노드 $y\in A$/* n_i의 유출 호 중 P에 존재하면 무시함

(1) $l(y)=\min[l(y), l(n_i)+w(n_i,y)]$로 치환(단, 동일한 경로길이 일 경우 $l(n_i)$호 선택, $l(n_i)+w(n_i,y)$ 경로 길이가 작을 경우 (n_i,y) 선택)

(2) $l(n_{i+1})\leq l(y)$인 노드 n_{i+1} 선택(단, 최소 경로 길이 노드가 다수 존재시 하나만 선택)

(3) 호 (n_i,n_{i+1})을 PSP(Partial Spanning Tree)에 저장, n_{i+1}을 Q에서 삭제하고 P에 저장

③ $i=i+1$, 그래프의 ST(Spanning Tree)를 얻을 때까지 ② 반복수행

④ PSP에 저장된 호들에 대해 출발에서 목적지 노드까지의 경로 설정, $l(s,d)$를 SP로 결정

그림 1. Dijkstra SP 알고리즘
Fig. 1. Dijkstra's SP Algorithm

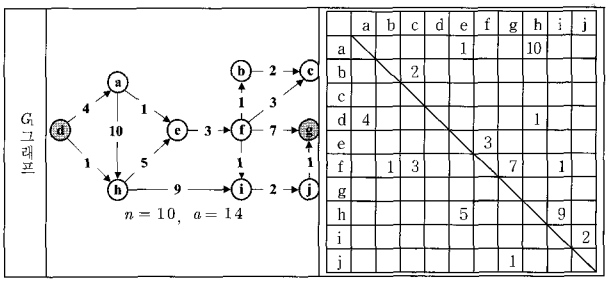


그림 2. G_1 그래프
Fig. 2. G_1 Graph

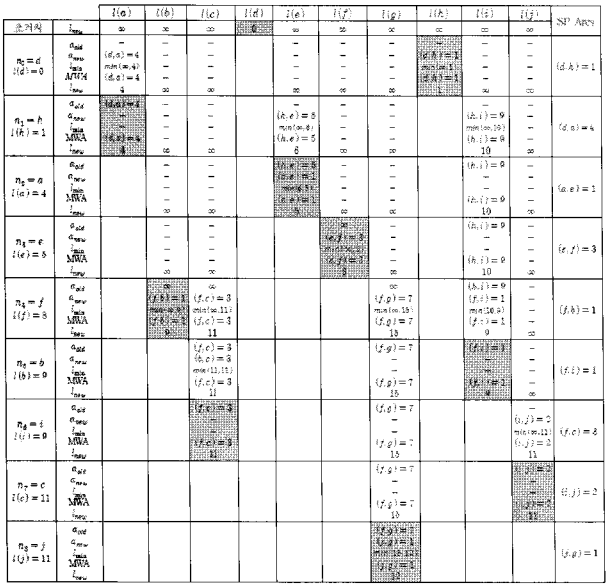
Dijkstra SP 알고리즘의 문제점은 많은 양의 메모리를 요구하며, 일반적으로 n-1회를 수행하기 때문에 수행속도가 느리다. 즉, 하나의 노드에서 경로의 합과 최소 가중치 호 저장에 각각 n개의 메모리가 요구되며, SP Arcs와 새로 탐색을 시작하는 노드의 경로 길이 $l(i)$이 1개씩 필요하다. 따라서 $n-2$개의 메모리가 $n-1$번 소요된다. 또한, 표 1에서 알 수 있듯이 $n_4=f$에서의 $l(b)=9$와 $l(i)=9$, $n_6=i$에

서의 $l(c)=11$과 $l(j)=11$이 동일한 경로 길이를 갖고 있음에도 불구하고 1개씩만 선택하기 때문에 $n-1$번 수행된다.

만약 복잡한 도시의 목적지까지 최단 경로를 찾는 경우, GPS 시스템에 Dijkstra SP 알고리즘을 적용하면 실시간으로 정보를 제공할 수 없는 경우도 발생할 수 있다. 따라서 실시간으로 정보를 제공하기 위해서는 수행 횟수를 단축시킬 수 있는 알고리즘이 요구된다. 3장에서는 점대점 최단거리를 찾기 위해 알고리즘 수행에 요구되는 메모리 크기도 줄이면서 수행속도를 월등히 향상시킬 수 있는 알고리즘을 제안한다.

표 1. G_1 그래프의 Dijkstra SP 알고리즘 적용

Table 1. Applying to Dijkstra's SP Algorithm to G_1 Graph



3. Sulee SP 알고리즘

3.1 알고리즘 제안

방향그래프의 점대점 최단거리를 빠르게 찾는 제안된 알고리즘을 "Sulee SP 알고리즘"이라 하자. 먼저, 그래프의 노드들을 그림 3과 같이 S, P, P_d와 D의 4 그룹으로 분류한다. 여기서 S는 출발 노드, P는 출발에서 목적지 까지의 경로 노드, P_d는 경로 노드들 중 목적지 노드에 인접한(목적지 노드로 유입되는) 노드, D는 목적지 노드이다.

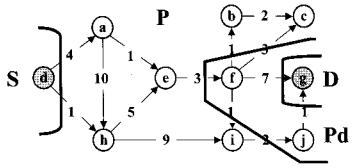


그림 3. G_1 그래프의 노드 분류

Fig. 3. Node Classification of G_1 Graph

Sulee SP 알고리즘은 "각 경로 노드의 유출과 유입 최소 가중치 호 선택"과 "노드 확장 방식"을 택한다. 먼저, 출발(S)과 목적지 노드(D)가 결정되었을 때, 경로 노드들($P \cup P_d$)의 유출 최소 가중치 호와 유입 최소 가중치 호들을 "노드 최소 가중치 호"로 사전에 선택한다. 최소 가중치 유출 호는 정방행렬의 행 노드에서, 최소 가중치 유입 호는 열 노드에서 선택된다.

노드 최소 가중치 호 선택"은 Borůvka MST 알고리즘[10,11]을 간선형한 형태를 적용한다. 이와 같이 하는 이유는 각 경로 노드들의 유입과 유출 최소 가중치 호들로 경로를 설정하여 출발에서 목적지 노드까지 경로 단절 없이 최단 경로를 갖도록 하기 위함이다. 참고로, Borůvka MST 알고리즘[10,11]은 모든 정점에 연결된 MWE를 선택하고 사이클(루프)이 발생하는 간선들을 삭제한다. 이렇게 얻은 간선들로 부분 신장트리(PST, Partial Spanning Tree)를 구성하여 PSP가 모든 정점들이 연결되어 있으면 알고리즘이 종료된다. 만약에 모든 정점들을 연결하지 못하면 PSP간 MWE를 선택하여 최종적으로 최소 가중치를 갖는 신장트리(MST)를 얻는다. Borůvka MST 알고리즘[10,11]은 정방행렬의 모든 행 정점에 대해서만 MWE를 선택(정점 최소 가중치 유출 간선)하는데 반해, Sulee SP 알고리즘은 경로노드에 대해서만 행 노드의 MWA(노드 최소 가중치 유출 호)와 열 노드의 MWA(노드 최소 가중치 유입 호)를 선택하는 방식이다.

다음으로, "노드 확장 방식"은 Prim MST 알고리즘[8]을 간소화한 형태를 적용한다. 출발 노드와 인접 노드 2개를 선택하여 최소 경로 길이를 가진 호를 선택하고, 이들 호와 관련된 노드들의 유출 호들에 대해 방문 노드로 유출되는 호를 제거 (사이클 검증)하고 방문하지 않은 노드로 유출되는 호들에 대해서만 "노드 최소 가중치 호와 일치하는 호들을 선택하고 일치하지 않는 호는 무시"하는 방식이다. 이렇게 선택된 호의 노드들을 계속적으로 확장해 나가면서 목적지 노드에 도달하면 알고리즘을 종료한다. Sulee SP 알고리즘의 상세한 내용은 그림 4에 제시되어 있다.

3.2 알고리즘 적용 방법

그림 2의 G_1 그래프에 Sulee SP 알고리즘을 적용한 과정은 표 2에, 적용 결과 얻은 SP는 그림 5에 제시되어 있다. 알고리즘 적용 결과, Sulee SP 알고리즘은 Dijkstra SP 알고리즘[5]과 동일한 SP 경로를 얻는데 성공하였다. 알고리즘 수행 횟수 측면에서 볼 때, Dijkstra SP 알고리즘[5]은 $n-1$=9회를 수행한데 비해 Sulee SP 알고리즘은 6회로 단축시킬 수 있었으며, 메모리 크기 측면에서도 Sulee SP 알고리즘이 월등히 적은 메모리를 사용함을 알 수 있다.

4. 알고리즘 적용성 평가

4.1 방향 그래프의 최단 경로 탐색

본 절에서는 그림 6과 같이 12개 그래프를 대상으로 Sulee SP 알고리즘의 적용성을 평가해 본다. G_2와 G_7 그래프는 Chen[12], G_3, G_5, G_9와 G_{13} 그래프는 Ikeda[13], G_4 그래프는 Wenger[14], G_6 그래프는 Llewellyn[9], G_8 그래프는 Boyd[15], G_{10} 그래프는 Orlin[16], G_{11} 그래프는 Coleman[17], G_{12} 그래프는 Walker[18]에서 인용되었다.

각 그래프에 대해 Dijkstra SP 알고리즘과 Sulee SP 알고리즘을 적용한 결과는 그림 7~그림 18에 제시되어 있다. 알고리즘 적용 결과 G_6와 G_9 그래프만이 다른 경로를 탐색하

였으며, 나머지 10개 그래프는 동일한 경로를 탐색하였다.

≪ Sulee SP Algorithm ≫

방향 그래프 $G=(N,A)$
N : 노드 (Nodes, n), A : 호 (Arcs, a), (i,j)
각 호들에 대해 $n\times n$ 인접행렬 작성
경로노드 최소 가중치 호 (MWA, a_{mwa}) 선택
1. 행 노드 /* 경로 노드 (P)와 목적지 연결 노드 (P_d)의 최소 가중치 유출 호 선택
출발 (S)과 목적지 모드 (D) : 선택 안함
1-1. 목적지 연결 노드 $(P_d(out), j\in D)$ 최소 가중치 유출 호
1-1-1. $(i,d)=MWA$이면 (i,d) 선택
1-1-2. $(i,d)\neq MWA$이면
1-1-2-1. $(i,j)+(j,d)$에서 $(j,d)\neq\varnothing$인 호가 존재하지 않으면 해당 노드의 유출 호는 선택 안함
1-1-2-2. $(i,j)+(j,d)$에서 $(j,d)\neq\varnothing$인 호가 존재하면 $\min(l[(i,j)+(j,d)])$ 선택
1-2. 경로 노드 $(P(out), j\in(P\cup P_D)$ 최소 가중치 유출 호
1-2-1. 경로 노드에서 MWA 선택(단, 동일한 가중치 호는 모두 선택)
2. 열 노드 /* 경로 노드 (P)와 목적지 연결 노드 (P_d)의 최소 가중치 유입 호 선택 $[P\cup P_d]$
출발 (S)과 목적지 노드 (D) : 선택 안함
2-1. 선택된 호가 MWA가 아닌 경우 또는 호가 선택되지 않은 경우 S, D, P_d를 제외한 경로 노드 (P)에서 MWA 선택
출발 노드를 SP Nodes (n_{sp})에 추가 /* 출발 노드부터 시작
① 출발 노드 (S) 행 : 최소 가중치 2개 호 $(s,i),(s,j)$ 선택 (만약 동일한 가중치이면 모두 선택)
$\min[(s,i),(s,j)+(j,i)]$와 $\min[(s,j),(s,i)+(i,j)]$ 계산 (만약 $(i,j),(j,i)$가 없으면 ∞로 놓음)
최소 가중치를 갖는 호들을 SP Arcs에 저장, 해당 노드를 SP Nodes (n_{sp})에 추가
② n_{sp}에 새로 추가된 노드 : 노드 유출 호들을 Candidate Arcs에 가 $(a_c=a_a)$
IF Candidate Arcs (a_a)의 $j\in n_{sp}$ THEN 해당 호 삭제 (a_d) /* 사이클 발생 호 삭제
ENDIF
③ $a_c(a_c=a_a-a_d)$와 a_{mwa} 비교, SP Arcs 선택
IF $a_c=a_{mwa}$ THEN SP Arcs (a_{sp})에 저장, a_c에서 삭제
ELSE IF $a_c\neq a_{mwa}$ THEN a_c에서 삭제
ENDIF
a_{sp}에 저장된 호의 "j" 노드를 n_{sp}에 저장
IF $a_{sp}(i,j)$의 $j=d$ THEN "④" 수행
ELSE "②"로 복귀
ENDIF
④ a_{sp}에서 얻은 호들을 사슬로 연결하여 목적지 까지 연결된 경로에 대해 $l(s,d)$가 최소로 되는 SP 선택

그림 4. Sulee SP 알고리즘
Fig. 4. Sulee's SP Algorithm

표 2. G_1 그래프의 Sulee SP 알고리즘 적용
Table 2. Applying to Sulee's SP Algorithm to G_1 Graph

a_{mwa} 선택	
경로 노드 최소 가중치 호 (MWA, a_{mwa}) 선택 1. 최소 가중치 유출 호 선택 (행 노드) 1-1. 목적지 연결 노드 $P_d(out)$ = {f,j} ⓕ : $(f,g)=7\neq MWA$ (1-1-2) $(f,b)=1+(b,g)=\varnothing$, $(f,c)=3+(c,g)=\varnothing$, $(f,j)=1+(j,g)=1\Rightarrow(f,j)=1$ 선택 (1-1-2-2) ⓙ : $(j,g)=1$ (MWA) $\Rightarrow(j,g)=1$ 선택 (1-1-1) 1-2. 경로 노드 $P(out)$ = {a,b,c,e,f,h,i} $(a,e)=1,(b,c)=2,(e,f)=3,(f,b)=1,(f,i)=1,$ $(h,e)=5,(i,j)=2$ 2. 최소 가중치 유입 호 선택 (열 노드) $[P\cup P_D(in)]$ = {a, b, c, e, f, h, i, j} ⓐ : $(d,a)=4$는 출발노드 유출 호로 선택 안함 b,c,e,f,i,j : 경로 노드에서 유입되는 MWA가 이미 선택되어 있어 선택 안함 ⓗ : $(d,h)=1$은 출발노드 유입으로 제외, 경로 노드 유입 $(a,h)=10$ 선택 (2-2)	(a,e)=1 (b,c)=2 (e,f)=3 (f,b)=1 (f,i)=1 (h,e)=5 (i,j)=2 (j,g)=1 (a,h)=10

N	SP Nodes	Candidate Arcs (a_c)			a_{mwa}	SP Arcs
		a_a	Cycle (a_d)	a_a-a_d		
{a,b,c,e,f,g,h,i,j}	{d}	(d,a)=4, (d,h)=1 (a,h)=10, (h,a)=∞	min[(d,a)=4, (d,h)+(h,a)=∞] min[(d,h)=1, (d,a)+(a,h)=14]			**(d,a)=4** **(d,h)=1**
{b,c,e,f,g,i,j}	{d,a,h}	(a,e)=**1**, **(a,h)=10** (h,e)=5, (h,i)=9	**(a,h)=10**	**(a,e)=1**, **(h,e)=5** (h,i)=9	(a,e)= 1 (b,c)= 2 (e,f)= 3 (f,b)= 1 (f,i)= 1 (h,e)= 5 (i,j)= 2 (j,g)= 1 (a,h)=10	**(a,e)=1** **(h,e)=5**
{b,c,f,g,i,j}	{d,a,h,e}	(e,f)=3	-	**(e,f)=3**		**(e,f)=3**
{b,c,g,i,j}	{d,a,h,e,f}	(f,b)=1, (f,c)=3 (f,g)=7, (f,i)=1	-	**(f,b)=1**, (f,c)=3 (f,g)=7, **(f,i)=1**		**(f,b)=1** **(f,i)=1**
{c,g,j}	{d,a,h,e,f,b,i}	(b,c)=2, (i,j)=2	-	**(b,c)=2**, **(i,j)=2**		**(b,c)=2** **(i,j)=2**
{g}	{d,a,h,e,f,b,i,c,j}	(j,g)=1	-	**(j,g)=1**		**(j,g)=1**

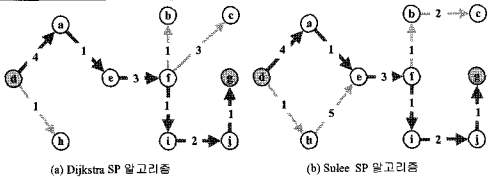


그림 5. G_1 그래프의 최단 경로
Fig. 5. Shortest Path of G_1 Graph

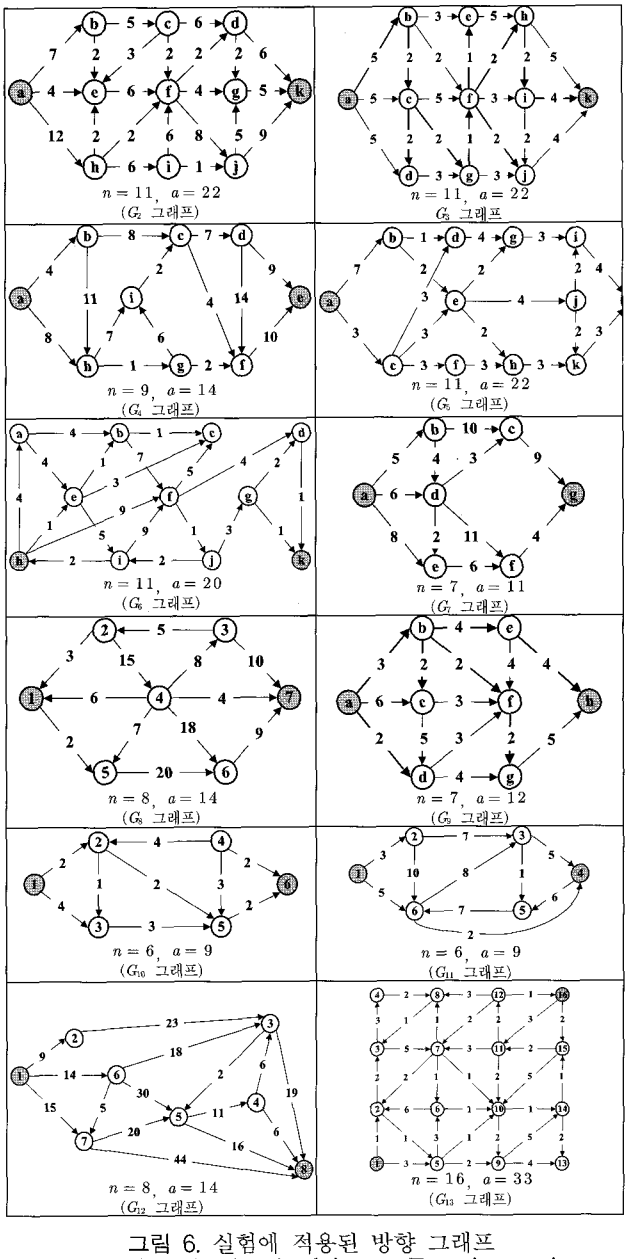


그림 6. 실험에 적용된 방향 그래프
Fig. 6. Digraphs for Applying to Experimentation

Dijkstra SP 알고리즘	SP Nodes	a	e	b	f	c	d	h	g	i	j
	SP Arcs	(a,e)=4	(a,b) =7	(e,f) =6	(b,c) =5	(f,d) =2	(a,h) =12	(f,g) =4	(h,i) =6	(f,j) =8	(d,k) =6
	SP	(a,e)=4→(e,f)=6→(f,d)=2→(d,k)=6 (4+6+2+6=18)									
Sulee SP 알고리즘	Node MWA	$P_d(out)=\{d,g,j\}$: (d,k)=6,(g,k)=5,(j,k)=9 $P(out)=\{b,c,e,f,h,i\}$: (h,e)=2,(c,f)=2,(e,f)=6,(h,e)=2,(h,f)=2,(i,j)=1 $P\cup P_d(in)=\{b,c,d,e,f,g,h,i,j\}$: (b,c)=5,(f,d)=2,(d,g)=2,(h,i)=6									
	SP Nodes	a			b,e			c,f		d	
	SP Arcs	(a,b)=7,(a,e)=4			(b,c)=5,(e,f)=6			(f,d)=2		(d,g)=2,(d,k)=6	
	SP	(a,e)=4→(e,f)=6→(f,d)=2→(d,k)=6 (4+6+2+6=18)									

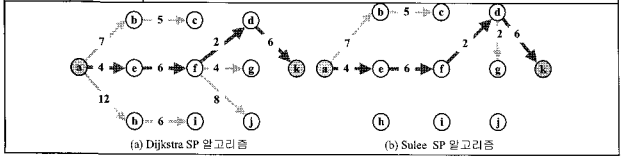


그림 7. G_2 그래프의 최단 경로
Fig. 7. Shortest Path for G_2 Graph

Dijkstra SP 알고리즘	SP Nodes	a	b	c	d	f	g	e	h	j	i
	SP Arcs	(a,b) =5	(a,c) =5	(a,d) =5	(b,f) =2	(c,g) =2	(b,e) =3	(f,h) =2	(f,j) =2	(f,i) =3	(j,k) =4
	SP	(a,b)=5→(b,f)=2→(f,j)=2→(j,k)=4 (5+2+2+4=13)									
Sulee SP 알고리즘	Node MWA	$P_d(out)=\{h,i,j\}$: (h,k)=5,(i,k)=4,(j,k)=4 $P(out)=\{b,c,d,e,f,g\}$: (b,c)=2,(b,f)=2,(c,d)=2,(c,g)=2,(d,g)=3,(e,h)=5,(f,e)=1,(g,f)=1 $P\cup P_d(in)=\{b,c,d,e,f,g,h,i,j\}$ (f,h)=2,(h,i)=2,(f,j)=2,(i,j)=2									
	SP Nodes	a			b,c,d			f,g		h,j	
	SP Arcs	(a,b)=5,(a,c)=5 (a,d)=5			(b,f)=2,(c,g)=2 (d,g)=3			(f,e)=1,(f,j)=2		(h,i)=2,(h,k)=5 (j,k)=4	
	SP	(a,b)=5→(b,f)=2→(f,j)=2→(j,k)=4 (5+2+2+4=13)									

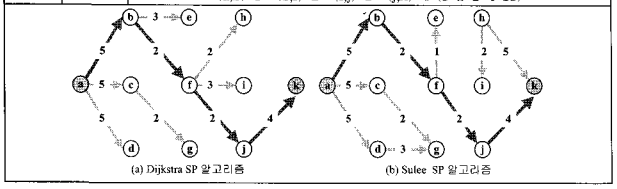


그림 8. G_3 그래프의 최단 경로
Fig. 8. Shortest Path for G_3 Graph

Dijkstra SP 알고리즘	SP Nodes	a	b	c	h	g	f	i	d
	SP Arcs	(a,b)=4	(b,c)=12	(a,h)=8	(h,g)=1	(g,f)=2	(h,i)=7	(c,d)=7	(f,e)=10
	SP	(a,h)=8→(h,g)=1→(g,f)=2→(f,e)=10 (8+1+2+10=21)							
Sulee SP 알고리즘	Node MWA	$P_d(out)=\{d,f\}$: (d,e)=8,(f,e)=10 $P(out)=\{b,c,d,f,g,h,i\}$: (b,c)=8,(c,f)=4,(g,f)=2,(h,g)=1,(i,c)=2 $P\cup P_d(in)=\{b,c,d,f,g,h,i\}$: (c,d)=7,(c,h)=11,(g,i)=6							
	SP Nodes	a		b,h		c,g		d,f,i	
	SP Arcs	(a,b)=4,(a,h)=8		(b,c)=8,(h,g)=1		(c,d)=7,(c,f)=4,(g,f)=2,(g,i)=6		(d,e)=9,(f,e)=10	
	SP	(a,h)=8→(h,g)=1→(g,f)=2→(f,e)=10 (8+1+2+10=21)							

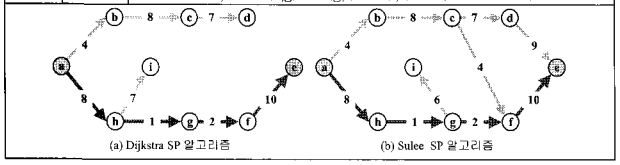


그림 9. G_4 그래프의 최단 경로
Fig. 9. Shortest Path for G_4 Graph

Dijkstra SP 알고리즘	SP Nodes	a	c	d	e	f	b	g	h	i	j	k
	SP Arcs	(a,c) =3	(c,d) =3	(c,e) =3	(c,f) =3	(a,b) =7	(e,g) =2	(e,h) =2	(e,j) =4	(g,j) =3	(h,k) =3	(k,l) =3
	SP	(a,c)=3→(c,e)=3→(e,h)=2→(h,k)=3→(k,l)=3 (3+3+2+3+3=14)										
Sulee SP 알고리즘	Node MWA	$P_d(out)=\{i,k\}$: (j,l)=4,(k,l)=3 $P(out)=\{b,c,d,e,f,g,h,j\}$: (b,d)=1,(c,d)=3,(c,e)=3,(c,f)=3,(d,g)=4,(e,g)=2,(e,h)=2 (f,h)=3,(g,j)=3,(h,k)=3,(j,l)=2,(j,k)=3 $P\cup P_d(in)=\{b,c,d,e,f,g,h,i,j,k\}$: (h,e)=2,(e,j)=4										
	SP Nodes	a			b,c			d,e,f			g,h,i	j,k
	SP Arcs	(a,b)=7,(a,c)=3			(b,d)=1,(b,e)=2 (c,d)=3,(c,e)=3 (c,f)=3			(d,g)=4,(e,g)=2 (e,h)=2,(e,j)=4 f,h)=3			(g,j)=3,(h,k)=3 (j,l)=2,(j,k)=2	(j,l)=4 (k,l)=3
	SP	(a,c)=3→(c,e)=3→(e,h)=4→(h,k)=3→(k,l)=3 (3+3+4+2+4=16)										

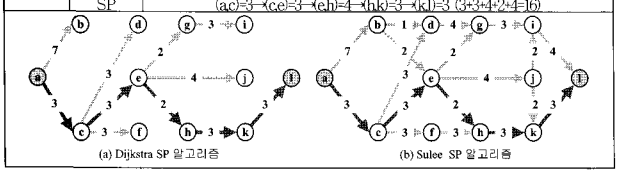


그림 10. G_5 그래프의 최단 경로
Fig. 10. Shortest Path for G_5 Graph

Dijkstra SP 알고리즘	SP Nodes	h	e	b	c	a	i	f	j	d	g
	SP Arcs	(h,e) =1	(e,b) =1	(b,c) =1	(h,a) =4	(e,i) =5	(h,f) =9	(f,j) =1	(f,d) =4	(j,g) =3	(d,k) =1
	SP	(h,f)=9→(f,d)=4→(d,k)=1 (9+4+1=14)									
Sulee SP 알고리즘	Node MWA	$P_d(out)=\{d,g\}$: (d,k)=1,(g,k)=1 $P(out)=\{a,b,c,e,f,i,j\}$: (a,b)=4,(a,e)=4,(b,c)=1,(e,b)=1,(f,j)=1 $P\cup P_d(in)=\{a,b,c,d,e,f,g,i,j\}$: (g,d)=2,(b,f)=7,(j,g)=3									
	SP Nodes	h		a,e		b		c,f		i	g,j
	SP Arcs	(h,a)=4 (h,e)=1		(a,b)=4 (e,b)=1		(b,c)=1 (b,f)=7		(f,j)=1		(j,g)=3 (i,j)=2	(g,d)=2 (g,k)=1
	SP	(h,e)=1→(e,b)=1→(b,f)=7→(f,j)=1→(j,g)=3→(g,k)=1 (1+1+7+1+3+1=14)									

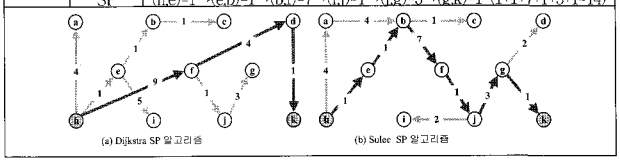


그림 11. G_6 그래프의 최단 경로
Fig. 11. Shortest Path for G_6 Graph

Dijkstra SP 알고리즘	SP Nodes	a	b	d	e	c	f
	SP Arcs	(a,b)=5	(a,d)=6	(a,e)=8	(d,c)=3	(e,f)=6	(c,g)=9
	SP	(a,d)=6→(d,c)=3→(c,g)=9 (6+3+9=18)					
Sulee SP 알고리즘	Node MWA	$P_d(out)=\{c,f\}$: (c,g)=9,(f,g)=4, $P(out)=\{b,d,e\}$: (b,d)=4,(d,e)=2,(e,f)=6 $P\cup P_d(in)=\{b,c,d,e,f\}$: (d,c)=3					
	SP Nodes	a		b,d		c,e	
	SP Arcs	(a,b)=5,(a,d)=6		(d,c)=3,(d,e)=2		(c,g)=9,(e,f)=6	
	SP	(a,d)=6→(d,c)=3→(c,g)=9 (6+3+9=18)					

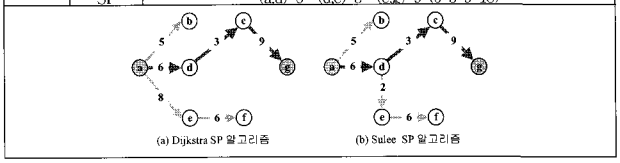


그림 12. G_7 그래프의 최단 경로
Fig. 12. Shortest Path for G_7 Graph

Dijkstr Dijkstra SP 알고리즘	SP Nodes	1	5	6
	SP Arcs	(1,5)=2	(5,6)=20	(6,7)=9
	SP	(1,5)=2→(5,6)=20→(6,7)=9 (2+20+9=31)		
Sulee SP 알고리즘	Node MWA	$P_d(out)=\{3,4,6\}$: (4,7)=4,(6,7)=9, $P(out)=\{2,5\}$: (2,4)=15,(5,6)=20 $P\cup P_d(in)=\{2,3,4,5,6\}$: (3,2)=5,(4,3)=8,(4,5)=7,(4,6)=18		
	SP Nodes	1	5	6
	SP Arcs	(1,5)=2	(5,6)=20	(6,7)=9
	SP	(1,5)=2→(5,6)=20→(6,7)=9 (2+20+9=31)		

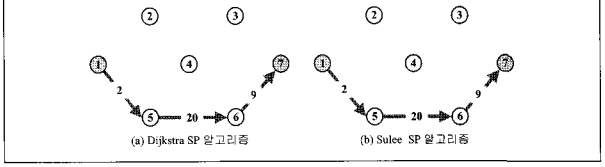


그림 13. G_8 그래프의 최단 경로
Fig. 13. Shortest Path for G_8 Graph

Dijkstra SP 알고리즘	SP Nodes	a	d	b	c	f	g	e
	SP Arcs	(a,d)=2	(a,b)=3	(b,c)=2	(d,f)=3	(d,g)=4	(b,e)=4	(g,h)=5
	SP	(a,d)=2→(d,g)=4→(g,h)=5 (2+4+5=11)						
Sulee Sulee SP 알고리즘	Node MWA	$P_d(out)=\{e,g\}$: (e,h)=4,(g,h)=5 $P(out)=\{b,c,d,f\}$: (b,c)=2,(b,f)=2,(c,f)=3,(d,f)=3,(f,g)=2 $P\cup P_d(in)=\{b,c,d,e,f,g\}$: (c,d)=5,(b,e)=4						
	SP Nodes	a		b,d				c,e,f
	SP Arcs	(a,b)=3,(a,d)=2		(b,c)=2,(b,e)=4,(b,f)=2,(d,f)=3				(e,h)=4,(f,g)=2
	SP	(a,b)=3→(b,e)=4→(e,h)=4 (3+4+4=11)						

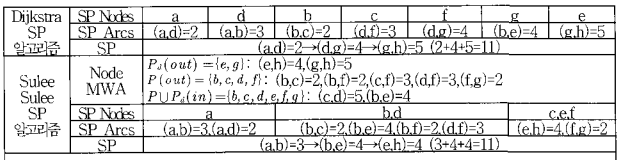


그림 14. G_9 그래프의 최단 경로
Fig. 14. Shortest Path for G_9 Graph

Dijkstra SP 알고리즘	SP Nodes	1	2	3	5	6
	SP Arcs	(1,2)=2	(1,3)=4	(2,5)=2	(5,6)=2	∞
	SP	(1,2)=2→(2,5)=2→(5,6)=2 (2+2+2=6)				
Sulee SP 알고리즘	Node MWA	$P_s(out)=\{4,5\}$: (4,6)=2,(5,6)=2, $P(out)=\{2,3\}$: (2,3)=1,(3,5)=3 $P\cup P_s(in)=2,3,4,5$: (4,2)=2,(2,5)=2				
	SP Nodes	1	2,3		5	
	SP Arcs	(1,2)=2,(2,3)=3	(3,5)=3,(2,5)=2		(5,6)=2	
	SP	(1,2)=2→(2,5)=2→(5,6)=2 (2+2+2=6)				

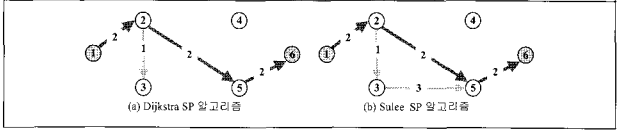


그림 15. G_{10} 그래프의 최단 경로
Fig. 15. Shortest Path for G_{10} Graph

Dijkstra SP 알고리즘	SP Nodes	1	2	6	4	3
	SP Arcs	(1,2)=3	(1,6)=5	(6,4)=2	(2,3)=7	(4,5)=6
	SP	(1,6)=5→(6,4)=2 (5+2=7)				
Sulee SP 알고리즘	Node MWA	$P_s(out)=\{3,6\}$: (6,4)=2, $P(out)=\{2,5\}$: (2,3)=7,(5,6)=7 $P\cup P_s(in)=\{2,3,5,6\}$: (3,5)=1				
	SP Nodes	1	2,6			
	SP Arcs	(1,2)=3,(1,6)=5	(2,3)=7,(6,4)=2			
	SP	(1,6)=5→(6,4)=2 (5+2=7)				

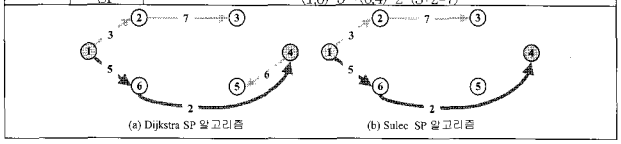


그림 16. G_{11} 그래프의 최단 경로
Fig. 16. Shortest Path for G_{11} Graph

Dijkstra SP 알고리즘	SP Nodes	1	2	6	7	3	5	4
	SP Arcs	(1,2)=9	(1,6)=14	(1,7)=15	(2,3)=23	(3,5)=2	(5,4)=11	(5,8)=16
	SP	(1,2)=9→(2,3)=23→(3,5)=2→(5,8)=16 (9+23+2+16=50)						
Sulee SP 알고리즘	Node MWA	$P_s(out)=\{3,4,5\}$: (3,5)=2,(4,8)=6,(5,8)=16 $P(out)=\{2,6,7\}$: (2,3)=23,(6,7)=5,(7,5)=20 $P\cup P_s(in)=\{2,3,4,5,6,7\}$: (4,3)=6						
	SP Nodes	1	2,6		3,7		5	
	SP Arcs	(1,2)=9,(1,6)=14	(2,3)=3,(6,7)=5		(3,5)=2,(7,5)=20		(5,4)=11,(5,8)=16	
	SP	(1,2)=9→(2,3)=3→(3,5)=2→(5,8)=16 (9+23+2+16=50)						

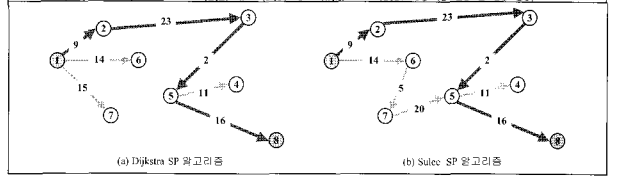


그림 17. G_{12} 그래프의 최단 경로
Fig. 17. Shortest Path for G_{12} Graph

Dijkstra SP 알고리즘	SP Nodes	1	2	5	3	10	9	14	4
	SP Arcs	(1,2)=1	(2,5)=1	(2,3)=2	(5,10)=1	(5,9)=2	(10,14)=1	(3,4)=3	(5,6)=3
	SP Nodes	6	15	13	7	8	11	12	
	SP Arcs	(14,15)=1	(14,13)=2	(3,7)=5	(4,8)=2	(15,11)=2	(11,12)=2	(12,16)=1	
	SP	(1,2)=1→(2,5)=1→(5,10)=1→(10,14)=1→(14,15)=1→ (15,11)=2→(11,12)=2→(12,16)=1 (1+1+1+1+1+2+2+1=10)							
Sulee SP 알고리즘	Node MWA	$P_s(out)=\{12\}$: (12,16)=1 $P(out)=\{2,3,\cdots,11,13,14,15\}$: (2,5)=1,(3,4)=3,(4,8)=2,(5,10)=1,(6,10)=1, (7,6)=1,(7,8)=1,(7,10)=1,(8,3)=1,(9,13)=4,(10,14)=1,(11,10)=2, (11,12)=2,(14,15)=1,(15,11)=2 $P\cup P_s(in)=\{2,3,\cdots,15\}$: (7,2)=2,(5,9=2,(10,9)=2,(14,13)=2							
	SP Nodes	1	2,5	9,10	13,14	15	11	12	
	SP Arcs	(1,2)=1 (2,5)=1	(5,9)=2 (5,10)=1	(9,13)=4 (10,14)=1	(14,15)=1	(15,11)=2	(11,12)=2	(12,7)=2 (12,16)=1	
	SP	(1,2)=1→(2,5)=1→(5,10)=1→(10,14)=1→(14,15)=1→ (15,11)=2→(11,12)=2→(12,16)=1 (1+1+1+1+1+2+2+1=10)							

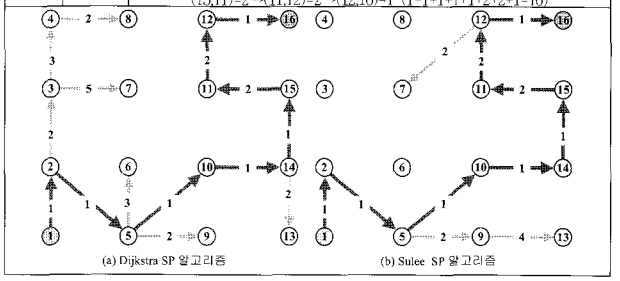


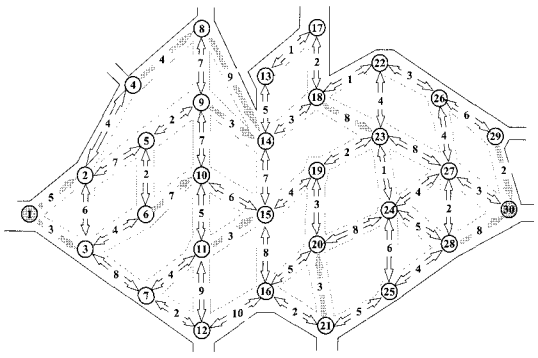
그림 18. G_{13} 그래프의 최단 경로
Fig. 18. Shortest Path for G_{13} Graph

G_7 그래프의 $(a,e)=8\rightarrow(e,f)=6\rightarrow(f,g)=4$ 경로도 $l(a,g)=18$로 동일한 최단 경로가 될 수 있다. 그러나 두 알고리즘 모두 이 경로는 선택하지 못하였다. G_8 그래프는 ② 노드가 출발 노드 ①로 유입되는 호만 존재하며, ④ 노드는 대부분이 유출되는 노드만 존재하여 ②,③,④ 노드를 방문하지 못하고 알고리즘이 빠르게 종료되었다. G_{10} 그래프에서, Dijkstra SP 알고리즘은 ④ 노드로 유입되는 호가 전혀 없기 때문에 5번째 수행에서 $l(4)=\infty$가 선택되었다. 이 경우 호를 선택하지 않는 것으로 하였다. G_{11} 그래프의 Dijkstra SP 알고리즘은 3번째 수행에서 $(6,4)=2$가 선택되어 목적지 노드 ④에 도달하였으나 모든 노드들을 방문해야 하기 때문에 불필요하게 2번이 추가로 수행되었다.

결론적으로, Sulee SP 알고리즘은 13개 그래프 모두에서 문제점 없이 SP를 얻는데 모두 성공하였다. 그러나 Dijkstra SP 알고리즘은 G_{10}과 G_{11} 그래프에서 알고리즘 적용에 약간의 문제가 발생하였다.

4.2 실제 도시의 최단 경로 탐색

본 절에서는 그림 19의 G_{14} 그래프를 대상으로 Sulee SP 알고리즘의 적용성을 평가해 본다. G_{14}그래프는 Lee[19]에서 인용되었으며, 출발 노드 "1"로부터 목적지 노드 "30" 까지 최단경로를 탐색하고자 한다. G_{14} 그래프는 대부분이 양방향 통행이 가능한 도로로 구성되어 있으며, 특정 도로는 일방통행만이 허용되는 경우이다. Dijkstra SP 알고리즘과 Sulee SP 알고리즘을 적용한 결과는 그림 20에 제시되어 있다. 알고리즘 적용 결과 동일한 경로를 탐색하였으며, Dijkstra SP 알고리즘은 29회 수행으로, Sulee SP 알고리즘은 단지 10회 수행으로 최단경로를 탐색하는데 성공하였다.



$n=30,\ a=91$

그림 19. G_{14} 그래프
Fig. 19. G_{14} Graph

4.3 알고리즘 성능 분석

알고리즘 적용에 활용된 14개 그래프에 대해 Dijkstra SP 알고리즘과 Sulee SP 알고리즘의 호 선택 수행횟수를 비교한 결과는 표 5에 제시하였다. Sulee SP 알고리즘은 Dijkstra SP 알고리즘에 비해 동일한 SP를 얻으면서, 평균적으로 알고리즘 수행 횟수를 약 2배 단축시키는 효과를 얻었다. 또한, 그래프의 노드수가 큰 G_{14}는 약 3배 정도 알고리즘이 빨리 수행됨을 알 수 있다.

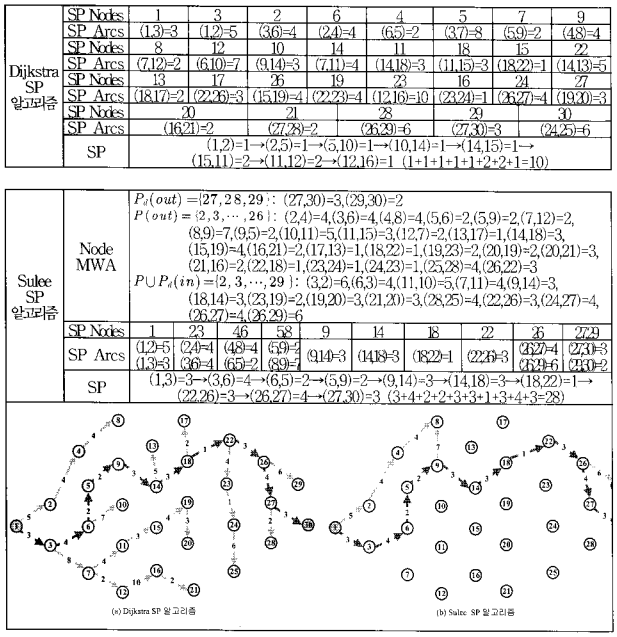


그림 20. G_{14} 그래프의 최단 경로
Fig. 20. Shortest Path for G_{14} Graph

표 5. 최단경로 탐색 알고리즘 성능 비교
Table 5. Performance Comparison of Shortest Path Search Algorithm

그래프	n	a	Dijkstra SP 알고리즘		Sulee SP 알고리즘		
			수행 횟수	$l(d)$	수행 횟수	$l(d)$	수행횟수 비율
G_1	**10**	14	**9**	12	**6**	12	66.7%
G_2	**11**	22	**10**	18	**4**	18	40.0%
G_3	**11**	22	**10**	13	**4**	13	40.0%
G_4	**9**	14	**8**	21	**4**	21	50.0%
G_5	**12**	22	**11**	14	**5**	14	45.0%
G_6	**11**	20	**10**	14	**6**	14	60.0%
G_7	**7**	11	**6**	18	**3**	18	50.0%
G_8	**8**	14	**3**	31	**3**	31	100%
G_9	**8**	14	**7**	11	**3**	11	42.9%
G_{10}	**6**	9	**5**	6	**3**	6	60.0%
G_{11}	**6**	9	**5**	7	**2**	7	40.0%
G_{12}	**8**	15	**7**	50	**4**	50	57.1%
G_{13}	**16**	33	**15**	10	**7**	10	46.7%
G_{14}	**30**	91	**29**	28	**10**	28	34.5%
평균							52.0%

5. 결론 및 향후 연구과제

본 논문에서는 실시간 GPS 항법 시스템에 일반적으로 적용되고 있는 Dijkstra SP 알고리즘의 문제점을 고찰해보고 새로운 최단 경로 탐색 알고리즘을 제안하였다. Dijkstra SP 알고리즘은 출발 노드부터 시작하여 모든 노드들을 한번에 하나씩 방문하는 방법으로 알고리즘 수행속도가 느린 단점이 있다. 따라서 복잡한 도시의 경우 실시간으로 최단 경로를 탐색하여 정보를 제공하지 못할 수도 있다. 이러한 단점을 보완하기 위해, 제안된 알고리즘은 한번에 하나의 노드를 탐색하지 않고 한번에 다수의 노드를 탐색하는 방법을 취하였다. 세부적으로는 출발과 목적지 노드가 설정되었을 때 경로 노드에 대해 유출과 유입되는 최소 가중치 호를 사전에 설정하고, 출발 노드부터 시작하여 연결된 호들에 대해 사전에 결정된 최소 가중치 호와 일치하는 호의 노드들을 모두 방문하는 방법을 택하였다. 14개의 다양한 방향 그래프에 대해 제안된 알고리즘을 적용한 결과 모두 최단경로를 탐색하는데 성공하였으며, Dijkstra SP 알고리즘에 비해 2배에서 3배 정도 알고리즘이 빨리 종료되는 효과를 얻었다. 따라서 제안된 알고리즘을 실시간 GPS 항법 시스템의 최단 경로 탐색에 적용하면 고객의 만족도를 향상시킬 수 있을 것이다.

본 논문에서는 방향 그래프의 최단 경로 탐색 알고리즘을 고찰하였다. 그러나 실제 도시의 도로는 모두 양방향 통행이 가능할 수도 있다. 양방향 통행이 가능한 도로만으로 되어 있는 경우 방향 그래프의 최단 경로 탐색 알고리즘이 적용되지 않을 수도 있다. 양방향 통행이 가능한 도로는 무방향 그래프로 생각할 수 있다. 따라서 추후 무방향 그래프의 최단 경로를 탐색하는 알고리즘을 연구할 계획이다.

참 고 문 헌

[1] M. Abboud, L. Mariya, A. Jaoude, and Z. Kerbage, "Real Time GPS Navigation System," 3rd FEA Student Conference, Department of Electrical and Computer Engineering, American University of Beirut, 2004.

[2] T. H. Cormen, C. E. Leserson, R. L. Rivest, and C. Stein, "*Introduction to Algorithms*," 2nd Edition, MIT Press and McGraw-Hill, 2001.

[3] E. W. Dijkstra, "A Note on Two Problems in Connection with Graphs," *Numerische Mathematik*, Vol. 1, pp. 269-271, 1959.

[4] Wikipedia, "Dijkstra's Algorithm,"http:// en. wikipedia.org/wiki/Dijkstra_algorithm, Wikimedia Foundation Inc., 2007.

[5] J. Misra, "A Walk Over the Shortest Path: Dijkstra's Algorithm Viewed as Fixed-Point Computation," Department of Computer Science, University of Texas at Austin, USA, 2000.

[6] Y. T. Lim and H. M. Kim, "A Shortest Path Algorithm for Real Road Network Based on Path Overlap," Department of Civil Engineering, Institute of Transportation Studies, University of California, Irvine, USA, http://www.its.uci.edu/~hyunmyuk/library/ (2005) 20EAST(k-path).pdf, 2005.

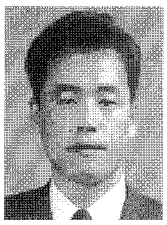
[7] F. B. Zhan, "Three Fatest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures," *Journal of Geographic Informaton and Decision Analysis*, Vol. 1, No. 1, pp. 69-82, 1997.

[8] R. C. Prim, "Shortest Connection Networks and Some Generalisations," *Bell System Technical Journal*, Vol. 36, pp. 1389-1401, 1957.

[9] M. Llewellyn, "COP 3503: Computer Science II - Introduction to Graphs," http://www.cs.ucf.edu/courses/cop3503/summer04, 2004.

[10] O. Borůvka, "O Jistem Problemu Minimalnim," Prace Mor. Prrodved. Spol. V Brne (Acta Societ. Natur. Moravicae), Vol. III, No. 3, pp. 37-58, 1926.
[11] J. Nešetřil, E. Milková, and H. Nešetřilová, "Otakar Borůvka on Minimum Spanning Tree Problem (Translation of the both 1926 Papers, Comments, History)," DMATH: Discrete Mathematics, Vol. 233, 2001.
[12] WWL. Chen, "Discrete Mathematics," Department of Mathematics, Division of ICS, Macquarie University, Australia, http://www. maths.mq.edu. au/~wchen/lndmfolder/lndm.html, 2003.
[13] K. Ikeda, "Mathematical Programming," Dept. Information Science and Intelligent Systems, The University of Tokushima, http:// www-b2.is.tokushima-u.ac.jp/~ike-da/suuri/maxflow/MaxflowApp. shtml. 2005.
[14] R. Wenger, "CIS 780: Analysis of Algorithms," http://www.cse. ohio-state.edu/~wenger/cis780/shortest_path.pdf, 2004.
[15] S. Boyd, "Applications of Combinational Optimization: Optimal Paths and Trees," http://www.site.uotta-wa.ca/~sylvia/csi5166web /5166tespg26to61.pdf, School of Information Technology and Engineering (SITE), University of Ottawa, Canada, 2005.
[16] J. B. Orlin, "Network Optimization: Dijkstra's Algorithm for the Shortest Path Problem," http://www.mit.edu/~jorlin/15.082/Lectures/ 05 Dijkstra.ppt, MIT, 2003.
[17] R. Coleman, "CS 221: Data Structures," Computer Science, University of Alabama in Huntsville, 2002.
[18] D. Walker, "COS 226: Algorithms and Data Structures," Department of Computer Science, Princeton University, http://www.cs.princeton.edu/ courses/archive/fall04/cos226/lectures/short-est-path.pdf, 2004.
[19] K. S. Lee, "Shortest Path Algorithm" http://www.geocities.com/ leekinseng1/.

저 자 소 개



이상운(Sang-Un, Lee)

1987년 : 한국항공대학교 항공전자공학과 (학사)
1997년 : 경상대학교 컴퓨터과학과(석사)
2001년 : 경상대학교 컴퓨터과학과(박사)
1992년~2002년 : 국방품질관리소 항공전자장비 소프트웨어 품질보증 담당
2003년 : 강원도립대학 컴퓨터응용과 전임강사
2004년~2007.2 : 원주대학 여성교양과 조교수
2007.3~현재 : 강릉대학교 컴퓨터공학부 멀티미디어정보공학 전공 조교수

관심분야 : 소프트웨어 프로젝트 관리, 소프트웨어 개발 방법론, 소프트웨어 척도, 분석과 설계 방법론, 소프트웨어 시험 및 품질보증, 소프트웨어 신뢰성, 신경망, 뉴로-퍼지, 알고리즘, 그래프 알고리즘
e-mail : sulee@kangnung.ac.kr