

<http://dx.doi.org/10.7236/JIWIT.2012.12.1.133>

JIWIT 2012-1-17

## 레벨 노드 선택 기반 점대점 최단경로 알고리즘

### A Point-to-Point Shortest Path Algorithm Based on Level Node Selection

이상운\*

Sang-Un Lee\*

**요 약** 본 논문은 실시간 GPS 항법시스템에서 최단경로 탐색에 일반적으로 적용되고 있는 Dijkstra 알고리즘의 수행 복잡도  $O(n^2)$ 을 선형인  $O(n)$ 으로 단축시킬 수 있는 알고리즘을 제안하였다. Dijkstra 알고리즘은 출발 노드부터 시작하여 모든 노드를 방문하여 최소 경로 길이를 계산한다. 따라서 “노드 수 -1”회를 수행해야 하기 때문에 복잡한 도로로 구성된 도시에서 실시간으로 최단경로 정보를 제공할 수 없는 경우도 발생한다. 제안된 알고리즘은 먼저, 그 래프를 트리로, 출발 노드를 근 노드로 치환하여 트리의 각 레벨에 해당하는 외부근방 (Out-Neighbourhood) 노드 집합을 구성하고, 외부근방과 외부근방 내부의 최소 경로 길이를 계산하는 방법을 적용하였다. 제안된 알고리즘을 양 방향과 일방통행으로 구성된 복잡한 2개 그래프에 대해 알고리즘을 적용한 결과 Dijkstra 알고리즘과 동일하게 모든 노드의 최소 경로 길이를 얻는데 성공하였다. 또한, 알고리즘 수행속도를 “노드 수 -1”회에서 “레벨 수 -1”회로 약 4 배 정도 단축시키는 효과를 얻었다. 제안된 알고리즘을 GPS 실시간 시스템에 적용하여 러시아워나 차량 사고로 인한 병목현상이 발생하였을 때, 최단 경로 우회 도로 정보를 실시간으로 제공할 수 있다면 운전자의 만족도를 크게 향상 시킬 수 있을 것이다.

**Abstract** This paper suggests an algorithm that can shorten the complexity  $O(n^2)$  of Dijkstra algorithm that is applied to the shortest path searching in real-time GPS Navigation System into an up-to-date  $O(n)$ . Dijkstra algorithm manipulates the distance of the minimum length path by visiting all the nodes from the starting node. Hence, it has one disadvantage of not being able to provide the information on the shortest path every second, in a city that consists of sophisticated roads, since it has to execute number of node minus 1. The suggested algorithm, firstly, runs by means of organizing the set of out-neighbourhood nodes at each level of the tree, and root node for departure node. It also uses a method of manipulating the distance of the minimum path of all out-neighborhoods and interior of the out-neighborhoods. On applying the suggested algorithm to two sophisticated graphs consisted of bi-direction and uni-direction, we have succeeded to obtain the distance of the minimum length path, just as same as Dijkstra algorithm. In addition, it has an effect of shortening the time taken 4 times from number of node minus1 to number of level minus 1. The satisfaction of the drivers can be increased by providing the information on shortest path of detour, every second, when occurs any rush hour or any traffic congestion due to car accident, by applying this suggested algorithm to the real-time GPS system.

**Key Words :** 최단경로 (Shortest Path), Dijkstra 알고리즘 (Dijkstra Algorithm), GPS 항법 시스템 (GPS Navigation System), 외부근방 (Out- Neighbourhood), 트리 레벨 (Tree Level)

\*정회원, 강릉원주대학교 멀티미디어공학과  
접수일자 2012.1.2, 수정완료 2012.2.4  
제재확정일자 2012.2.10

Received: 2 January 2012 / Revised: 4 February 2012 /  
Accepted: 10 February 2012  
\*Corresponding Author: sulee@gwnu.ac.kr  
Dept. of Multimedia Engineering, Science-Technology,  
Gangneung-Wonju National University, Korea

## I. 서 론

최근 실시간 GPS 항법 시스템 (Real Time GPS Navigation System)에 방향 그래프 (Digraph)의 최단경로 (SP, Shortest Path)를 구하는 Dijkstra 알고리즘이 일 반적으로 적용되고 있다.<sup>[1]</sup> 실시간 GPS 항법 시스템은 운전자에게 자신의 현재 위치를 화면상에 지정하고, 목적지를 결정하면 최단경로 (시간 또는 경로)를 결정하여 화면상에 표시해주고 운전자가 길을 따라 가도록 유도하는 시스템이다. 최단경로를 찾는 알고리즘은 Dijkstra, Bellman Ford, Topological Ordering과 A-Star(A\*) 등이 있다.<sup>[1,2]</sup> Dijkstra 알고리즘은 양의 가중치를 갖는 호들로 구성된 방향 그래프의 단일 출발점 최단경로 (Single-Source Shortest Path)를 찾는 알고리즈다.<sup>[3-5]</sup> GPS 시스템의 최단경로는 단일 출발점과 단일 목적지의 최단경로를 찾는 점대점 (P2P, Point-to-Point) 최단경로 알고리즘으로 Dijkstra 알고리즘의 특별한 경우이다. 점대점 최단경로 탐색 문제도 Dijkstra 알고리즘에 기본을 두고 있다.<sup>[1,6,7]</sup>

실제 양방향과 일방통행로가 혼합되어 구성된 복잡한 도시에서 러시아워 또는 차량 사고로 인해 정체될 때 GPS 항법시스템이 최단 경로 (시간, 운행비용 등)를 실시간으로 제공할 수 있다면 고객의 만족도를 크게 향상 시킬 수 있다. Dijkstra 알고리즘은 알고리즘 수행 횟수가 “노드 수 -1”로 과다한 시간이 소요되며, 수행 복잡도는  $O(n^2)$ 으로 알려져 있다. 지금까지 최단 경로 탐색 알고리즘의 수행 복잡도를 선형인  $O(n)$ 으로 할 수 있는 알고리즘은 미해결 과제로 남아 있다.<sup>[8]</sup>

본 논문은 수행 복잡도를 선형에 가까운 최단 경로를 탐색하는 알고리즘을 제안한다. 제안된 알고리즘은 그래프를 트리로, 출발 노드를 근 노드로 하여, 인접한 외부근방 (Out-Neighbourhood) 노드 집합을 레벨로 구성하고 레벨과 레벨간 (Inter-level) 노드의 최소 경로 길이와 동일 레벨에 존재하는 노드들 간 (Intra-level) 의 최소 경로 길이를 계산하는 방식으로 알고리즘 수행 복잡도는  $O(l \times n)$ 으로 단축시킬 수 있었다. II장에서는 Dijkstra 알고리즘의 최단경로를 찾는 과정을 고찰해 보고 문제점을 살펴본다. III장에서는 Dijkstra 알고리즘의 단점을 보완한 레벨 단위 SP 알고리즘을 제안하고 알고리즘 적용성을 평가해 본다.

## II. 관련연구와 연구배경

### 1. 최단경로 탐색 알고리즘

그래프는 정점들 (Vertices)과 간선들(Edges)로 구성되어 있으며, 정점들이 간선들로 연결되어 있고 (Connected), 간선들은 방향성 (Directed)과 무방향성 (Undirected)으로 구분된다. 무방향 그래프 (Undirected Graph)는  $G = (V, E)$ 로 표기하며, 방향 그래프 (Digraph)는  $D = (N, A)$ 로 표기한다. 노드 (Nodes,  $N$ )는 정점 (Vertices,  $V$ )이라 하며, 호 (Arcs,  $A$ )는 간선 (Edges,  $E$ ) 또는 화살표 (Arrows)라고도 한다. 무방향 그래프의 간선  $e = w\{x, y\}$ 로, 방향 그래프의 호는 방향성이므로 순서쌍  $a = w(x, y)$ 로 표기한다. 여기서,  $x$ 는 꼬리 (Tail),  $y$ 는 머리 (Head)라 하며, 노드  $n$ 은 유출 차수와 유입 차수를 갖는다. 또한 간선과 호는 가중치를 갖고 있다.<sup>[8,9]</sup>

방향그래프의 최단경로를 찾는 대표적인 Dijkstra 알고리즘<sup>[3-5]</sup>은 단일 출발 노드 최단경로를 찾는 알고리즘으로 그림 1과 같다.<sup>[10]</sup>

```

방향 그래프  $D = (N, A)$  와 가중치 함수  $w: A \rightarrow \mathbb{N}$ .
출발 노드를  $s \in N$ ,  $l(s) = 0$ 로 설정하고  $P$ 에 저장.
 $x \neq s$ 인 모든 노드  $x \in N \setminus s$ 에 대해  $l(x) = \infty$ 로 설정,  $Q$ 에 저장.
 $n_1 = s$ .
for  $i = 1$  to  $n$ 
begin
    for  $i = 1$  to  $n$  /*  $n_i$  인접 모든 노드  $y \in (N \cap Q)$ .
 $n_i$ 에 인접한 모든 노드  $y \in (N \cap Q)$ 에 대해
begin
     $l(y) = \min[l(y), \{l(s) + w(s, y)\}]$ .
     $l(n_{i+1}) \leq l(y)$ 인 노드  $n_{i+1}$  선택.
    PSP (Partial Spanning Tree)  $\leftarrow (n_i, n_{i+1})$ .
     $n_{i+1}$ 을  $Q$ 에서 삭제,  $P$ 에 저장.
end
end
PSP에 저장된 노드들에 대해 출발에서 목적지 노드까지의 경로 설정,  $l(s, d)$ 를
SP로 결정.

```

그림 1. Dijkstra 알고리즘

Fig. 1. Dijkstra algorithm

Dijkstra 알고리즘은 “특정 노드로부터 시작하여 인접 노드의 최소 경로 길이를 계산하고 최소 경로 길이를 갖는 노드를 한번에 하나씩 선택하는 방식”으로, 특정 노드에 인접한 노드들과 이전에 계산된 노드들의 경로의 합이 최소가 되는 노드를 찾는다. 이 과정을 출발 노드로부터 시작하여 모든 노드들을 방문하는 최단경로를 찾기 때문에  $n-1$ 번 수행된다. 이는 Prim의 최소신장트리 (MST, Minimum Spanning Tree)를 찾는 알고리즘[11]

과 유사한 방법이다. 참고로, Prim MST 알고리즘[11]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. Dijkstra 알고리즘을 무방향 그래프에 적용하기 위해서는 노드를 정점으로, 호를 간선으로 변환하면 된다.

## 2. 알고리즘 적용 문제점과 연구 배경

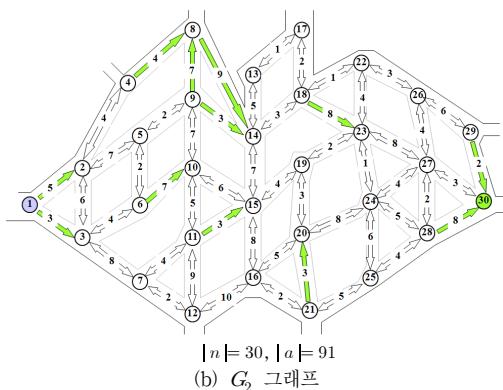
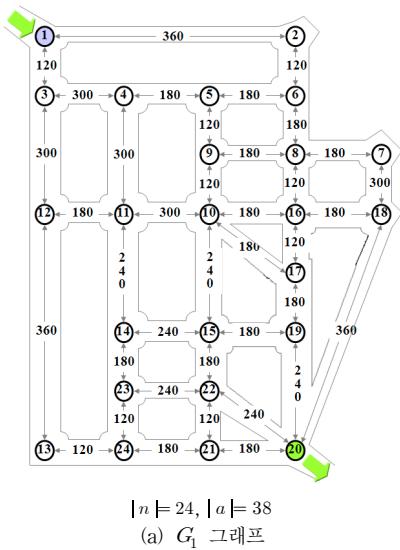


그림 2. 실험에 적용된 그래프

Fig. 2. Experimental Graphs

Dijkstra 알고리즘을 그림 2의  $G_1$ 과  $G_2$  그래프에 대해 적용하여 보자.  $G_1$  그래프는 Lim과 Kim[6]에서 인용되었으며, 미국 South Dakota 소재의 Sioux Fall 대학의 네트워크이다.

모든 호들이 양방향으로 구성되어 있고  $|n|=24, |a|=38$ 이다. 노드 “1”에서 출발하여 노드 “20”에 도착하는 최단경로를 찾고자 한다.  $G_2$  그래프는 Lee[12]에서 인용되었으며,  $|n|=30, |a|=91$ 로 구성되어 있으며, 양방향과 단방향 (일방통행)이 혼합된 그래프로 노드 “1”에서 출발하여 노드 “30”에 도착하는 최단경로를 찾는 문제이다.

$G_1$ 과  $G_2$  그래프의 Dijkstra 알고리즘을 적용한 결과는 각각 표 1과 표 2에 제시되어 있다.

$G_1$  그래프의 경우, 24개 노드의 최단 경로를 찾기 위해 23번의 알고리즘이 수행되었으며, 노드 1에서 출발하여 노드 20까지 도착하는 최단 경로 길이  $l(1, 20) = 1260$ 을 얻고, 경로는  $(1, 3), (3, 12), (12, 13), (13, 24), (24, 21), (21, 20)$ 이다.

$G_2$  그래프의 경우, 30개 노드의 최단 경로를 찾기 위해 29번의 알고리즘이 수행되었고, 노드 1에서 출발하여 노드 30까지 도착하는 최단 경로 길이  $l(1, 30) = 28$ 을 얻었으며,  $(1, 3), (3, 6), (6, 5), (5, 9), (9, 14), (14, 18), (18, 22), (22, 26), (26, 27), (27, 30)$  경로이다.

Dijkstra 알고리즘의 수행 복잡도는 최소 경로 길이 계산에  $O(n)$ , 최소 경로 길이 노드 선택에  $O(n)$ 으로  $O(n^2)$ 이 된다.

최단경로를 찾는 문제에서, 양의 가중치를 갖는 방향 그래프에서 하나의 노드에서 다른 모든 노드로의 경로를 계산하는 선형 알고리즘을 찾을 수 있는지가 문제점으로 남아 있다.[8] 만약 실시간으로 복잡한 도시의 목적지까지 최단경로를 찾는 경우, GPS 시스템에 Dijkstra 알고리즘을 적용하면 실시간으로 정보를 제공할 수 없는 경우도 발생할 수 있다. 따라서 실시간으로 정보를 제공하기 위해서는 수행 횟수를 선형으로 단축시킬 수 있는 알고리즘이 요구된다. III장에서는 점대점 최단경로를 찾기 위해 알고리즘 수행 복잡도를 선형으로 계산할 수 있는 알고리즘을 제안한다.

표 1.  $G_1$  그래프의 Dijkstra 알고리즘 적용

Table 1. Applying to Dijkstra algorithm for  $G_1$  graph

표 2.  $G_2$  그래프의 Dijkstra 알고리즘 적용

Table 2. Applying to Dijkstra algorithm for  $G_2$  graph

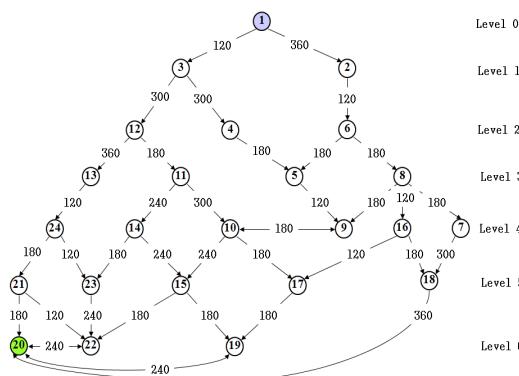
### III. 레벨단위 최단경로 알고리즘

#### 1. 알고리즘 제안

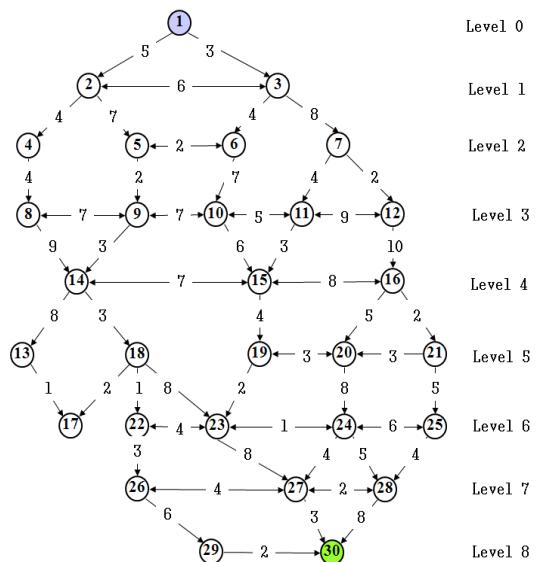
$D = (N, A)$ 의 하나의 노드  $s$ 에 대해,  $s$ 의 근방 (Neighborhood)을  $N(s) = N^+(s) \cup N^-(s)$ 라 한다. 여기서  $N^+(s) = \{u \in N - s : su \in A\}$ 를  $s$ 의 외부근방 (Out-neighbourhood),  $N^-(s) = \{w \in N - s : ws \in A\}$ 를 내부근방 (In-neighbourhood)이라 한다. 방향 그래프의 점대점 최단경로 탐색을 위해서는 외부 근방을 고려 한다. 출발 노드  $s$ 의 외부근방은  $N^{+p}(s) = \bigcup_{i=0}^p N^{+i}(s)$ 로 표현된다.<sup>[8]</sup>

최단 경로를 계산하기 위해 그래프를 트리로 생각하고, 출발 노드  $s$ 를 근 노드로 하여  $s$ 에 인접한 모든 노드를 방문하여  $dist(s, x) := 1$  ( $1^{\text{st}}$  외부근방)로 설정 한다.  $s$ 에서 경로가 1인 노드들  $x$ 에 인접하고 아직 방문하지 않은 모든 노드들  $y$ 를 방문하여  $dist(s, y) := 2$  ( $2^{\text{nd}}$  외부근방)로 설정한다.  $s$ 로부터 도달할 수 있는 모든 노드들을 방문할 때까지 반복 수행한다. 이 알고리즘을 인접리스트 (Adjacency List)로 구현하면 알고리즘의 복잡도는  $O(n)$ 이 된다.

$G_1$  그래프에 대해 트리 레벨을 얻은 결과는 그림 3과 같으며, 외부근방은  $N^0(s) = \{1\}$ ,  $N^{+1}(s) = \{2, 3\}$ ,  $N^{+2}(s) = \{4, 6, 12\}$ ,  $N^{+3}(s) = \{5, 8, 11, 13\}$ ,  $N^{+4}(s) = \{7, 9, 10, 14, 16, 24\}$ ,  $N^{+5}(s) = \{15, 17, 18, 21, 23\}$ ,  $N^{+6}(s) = \{19, 20, 22\}$ 이 된다.

그림 3.  $G_1$  그래프의 외부근방Fig. 3. Out-neighbourhood of  $G_1$  graph

$G_2$  그래프에 대해 트리 레벨을 얻은 결과는 그림 4에 제시되어 있으며, 외부근방은  $N^0(s) = \{1\}$ ,  $N^{+1}(s) = \{2, 3\}$ ,  $N^{+2}(s) = \{4, 5, 6, 7\}$ ,  $N^{+3}(s) = \{8, 9, 10, 11, 12\}$ ,  $N^{+4}(s) = \{14, 15, 16\}$ ,  $N^{+5}(s) = \{13, 18, 19, 20, 21\}$ ,  $N^{+6}(s) = \{17, 22, 23, 24, 25\}$ ,  $N^{+7}(s) = \{26, 27, 28\}$ ,  $N^{+8}(s) = \{29, 30\}$ 을 얻을 수 있다.

그림 4.  $G_2$  그래프의 외부근방Fig. 4. Out-neighbourhood of  $G_2$  graph

두 번째로  $p^{\text{th}}$  외부근방 집합에 대해 레벨 간 (Inter-Level  $N^{+p}(s) \rightarrow N^{+(p+1)}$ ) 최소 경로 길이를 계산한 후, 레벨 내부 (Intra-Level,  $N^{+(p+1)} \rightarrow N^{+(p+1)}$ )에 대한 최소 경로 길이를 계산하여 경로 길이를 갱신한다.

이 과정을 트리의 모든 레벨 ( $l$ )에 도달할 때까지 수행 한다. 이 과정의 알고리즘 복잡도는  $O(l \times n) \simeq O(n)$ 이 된다. 결국 제안된 알고리즘의 복잡도는  $O(n) + O(n) = O(n)$ 으로 선형이 됨을 알 수 있다. 제안된 알고리즘은 그림 5에 제시되어 있다.

방향 그래프  $D = (N, A)$  와 가중치 함수  $w: A \rightarrow \mathbb{N}$ .  
 방향 그래프를 트리로 가정, 출발 노드를  $s \in N$ 를 근 (Root)으로 치환  
 $p = 0, l(s) = 0, x \neq s$ 인 모든 노드  $x \in N \setminus s$ 에 대해  $l(x) = \infty$ .  
 $N^0(s) = s, N^{+p}(s), p = 0, 1, 2, \dots$ .  
 $n_1 = s$ .  
 for  $i = 1$  to  $n$  /\* 트리 레벨 설정  
 begin  
 $N^{+p}$  집합 노드 인접 모든 노드  $y \in (N \cap Q)$ 에 대해  
 $N^{+(p+1)} \leftarrow y$ .  
 $p = p + 1$ .  
 end  
 for  $p = 0$  to  $l$  /\* 레벨 단위 최단경로 계산  
 begin  
 $P \leftarrow N^{+(p)}(s), Q \leftarrow N^{+(p+1)}$ .  
 $n_i \in P$ 의 모든 인접 노드  $n_j \in Q$ 에 대해  
 $l(n_j) = \min[l(n_j), \{l(n_i) + w(n_i, n_j)\}]$ . /\* Inter-level 최단 경  
 로길이 계산.  
 $n_j \in Q$ 의 모든 인접 노드  $n_k \in Q$ 에 대해  
 $l(n_j) = \min[l(n_j), \{l(n_k) + w(n_k, n_j)\}]$ . /\* Intra-level 최단 경  
 로길이 계산.  
 $l(n_{i+1}) \leq l(n_j)$ 인 노드  $n_{i+1}$  선택.  
 PSP (Partial Spanning Tree)  $\leftarrow (n_i, n_{i+1})$ .  
 end  
 PSP에 저장된 노드들에 대해 목적지부터 출발까지의 경로를 역으로 설정,  $l(s, d)$ 를 SP  
 로 결정.

### 그림 5. 레벨 단위 최단경로 알고리즘

Fig. 5. Level based Shortest Path Algorithm

## 2. 알고리즘 적용 방법

$G_1$  과  $G_2$  그래프에 레벨 단위 최단경로 알고리즘을 적용한 과정은 각각 표 3과 표 4에 제시되어 있다. 2개 그래프 모두에서 레벨 단위 최단경로 알고리즘은 Dijkstra 알고리즘과 동일하게 모든 노드의 최소 경로 길이와 PSP를 얻는데 성공하였다. 그러나 알고리즘 수행 복잡도는  $O(n^2)$ 에서  $O(l \times n)$ 으로 단축시킬 수 있었으며, 수행속도는 Dijkstra 알고리즘은  $G_1$ 과  $G_2$ 에서 각각 “노드 수 -1 ( $n-1$ ) 회”인 23회와 29회를 수행하는데 반해, 레벨 단위 최단경로 알고리즘은 각각 “레벨 수 -1 ( $l-1$ ) 회”인 6회와 8회로 단축시킬 수 있었다.

표 3.  $G_1$  그래프의 레벨 단위 최단경로 알고리즘 적용

Table 3. Level based shortest path algorithm for  $G_1$  graph

Out-Neighborhood		Candidate Arcs		
$N^0(s) = \{1\}$	( $p, q$ )	(1,2)=360, (1,3)=120		
	( $r, q$ )	-		
$N^{+1}(s) = \{2, 3\}$	( $p, q$ )	(2,6)=120, (3,4)=300, (3,12)=300		
	( $r, q$ )	-		
$N^{+2}(s) = \{4, 6, 12\}$	( $p, q$ )	(4,5)=180, (4,11)=300, (6,5)=180, (6,8)=180, (12,11)=180, (12,13)=360		
	( $r, q$ )	-		
$N^{+3}(s) = \{5, 8, 11, 13\}$	( $p, q$ )	(5,9)=120, (8,7)=180, (8,9)=180, (8,16)=120, (11,10)=300, (11,14)=240, (13,24)=120		
	( $r, q$ )	(9,10)=120, (10,9)=120, (10,16)=180, (16,10)=180		
$N^{+4}(s) = \{7, 9, 10, 14, 16, 24\}$	( $p, q$ )	(7,8)=300, (10,15)=240, (14,15)=240, (10,17)=180, (14,23)=180, (16,17)=120, (16,18)=180, (24,21)=180, (24,23)=120		
	( $r, q$ )	-		
$N^{+5}(s) = \{15, 17, 18, 21, 23\}$	( $p, q$ )	(15,19)=180, (15,22)=180, (17,19)=180, (18,20)=360, (21,20)=180, (21,22)=120, (23,22)=240		
	( $r, q$ )	(19,20)=240, (20,19)=240, (20,22)=240, (22,20)=240		

$P$	$Q$	$\bar{x}$	Distance and PSP			
			Calculate of Length		Length	PSP
초기화					$l(1) = 0$	-
$N^0(s)$	$N^{+1}(s)$	( $p, q$ )	2 : $\min(\infty, 0+(1,2)=360)=360$		$l(2) = 360$	(1,2)=360
		( $r, q$ )	3 : $\min(\infty, 0+(1,3)=120)=120$		$l(3) = 120$	(1,3)=120
		-	-		-	-
$N^{+1}(s)$	$N^{+2}(s)$	( $p, q$ )	4 : $\min(\infty, 120+(3,4)=300)=420$		$l(4) = 420$	(3,4)=300
		( $r, q$ )	6 : $\min(\infty, 300+(2,6)=120)=480$		$l(6) = 480$	(2,6)=120
		( $p, q$ )	12 : $\min(\infty, 120+(3,12)=300)=420$		$l(12) = 420$	(3,12)=300
		( $r, q$ )	-		-	-
		( $p, q$ )	5 : $\min(\infty, 420+(4,5)=180, 480+(6,5)=180)=600$		$l(5) = 600$	(4,5)=180
$N^{+2}(s)$	$N^{+3}(s)$	( $p, q$ )	8 : $\min(\infty, 480+(6,8)=180)=660$		$l(8) = 660$	(6,8)=180
		( $r, q$ )	11 : $\min(\infty, 420+(4,11)=300, 420+(12,11)=180)=600$		$l(11) = 600$	(12,11)=180
		( $p, q$ )	13 : $\min(\infty, 420+(12,13)=360)=780$		$l(13) = 780$	(12,13)=360
		( $r, q$ )	-		-	-
		( $p, q$ )	7 : $\min(\infty, 600+(8,7)=180)=840$		$l(7) = 840$	(8,7)=180
$N^{+3}(s)$	$N^{+4}(s)$	( $p, q$ )	9 : $\min(\infty, 600+(5,9)=120, 600+(8,9)=180)=720$		$l(9) = 720$	(5,9)=120
		( $r, q$ )	10 : $\min(\infty, 600+(11,10)=300)=900$		$l(10) = 900$	(11,10)=300
		( $p, q$ )	14 : $\min(\infty, 600+(11,14)=240)=840$		$l(14) = 840$	(11,14)=240
		( $r, q$ )	16 : $\min(\infty, 600+(8,16)=120)=780$		$l(16) = 780$	(8,16)=120
		( $p, q$ )	24 : $\min(\infty, 780+(13,24)=120)=900$		$l(24) = 900$	(13,24)=120
$N^{+4}(s)$	$N^{+5}(s)$	( $p, q$ )	9 : $\min(20, 900+(10,9)=120)=720$		-	-
		( $r, q$ )	10 : $\min(900, 720+(9,10)=120, 780+(16,10)=180)=180=840$		$l(10) = 840$	(9,10)=120
		( $p, q$ )	16 : $\min(780, 900+(10,16)=180)=780$		-	-
		( $r, q$ )	15 : $\min(\infty, 840+(10,15)=240, 840+(14,15)=240)=1080$		$l(15) = 1080$	(10,15)=240
		( $p, q$ )	17 : $\min(\infty, 840+(10,17)=180, 780+(16,17)=120)=900$		$l(17) = 900$	(16,17)=120
$N^{+5}(s)$	$N^{+6}(s)$	( $p, q$ )	18 : $\min(\infty, 840+(7,18)=300, 780+(16,18)=180)=900$		$l(18) = 900$	(16,18)=180
		( $r, q$ )	21 : $\min(\infty, 900+(24,21)=180)=1080$		$l(21) = 1080$	(24,21)=180
		( $p, q$ )	23 : $\min(\infty, 840+(14,23)=180, 900+(24,23)=120)=1020$		$l(23) = 1020$	(14,23)=180
		( $r, q$ )	-		-	-
		( $p, q$ )	19 : $\min(\infty, 1080+(15,19)=180, 900+(17,19)=180)=1080$		$l(19) = 1080$	(17,19)=180
$N^{+6}(s)$		( $p, q$ )	20 : $\min(\infty, 960+(18,20)=360, 1080+(21,20)=180)=1200$		$l(20) = 1200$	(21,20)=180
		( $r, q$ )	22 : $\min(\infty, 1080+(15,22)=180, 1080+(21,22)=120)=1020$		$l(22) = 1020$	(21,22)=120
		( $p, q$ )	19 : $\min(1080, 1200+(20,19)=240)=1080$		-	-
		( $r, q$ )	20 : $\min(1200, 1080+(19,20)=240, 1200+(22,20)=240)=1200$		-	-
		( $p, q$ )	22 : $\min(1200, 1200+(20,22)=240)=1200$		-	-

$$\text{SP} : l(1,20) = \Sigma \{(1,3) = 120 + (3,12) = 300 + (12,13) = 360 + (13,24) = 120 + (24,21) = 180 + (21,20) = 180\} \Rightarrow 1260$$

표 4.  $G_2$  그래프의 레벨 단위 최단경로 알고리즘 적용  
Table 4. Level based shortest path algorithm for  $G_2$  graph

Out-Neighborhood	Candidate Arcs
$N^0(s) = \{1\}$	( $p, q$ ) (1,2)=5, (1,3)=3
$N^{+1}(s) = \{2, 3\}$	( $r, q$ ) (3,2)=6 ( $p, q$ ) (2,4)=4, (2,5)=7, (3,6)=4, (3,7)=8
$N^{+2}(s) = \{4, 5, 6, 7\}$	( $r, q$ ) (5,6)=2, (6,5)=2 ( $p, q$ ) (4,8)=4, (5,9)=2, (6,10)=7, (7,11)=4, (7,12)=2
$N^{+3}(s) = \{8, 9, 10, 11, 12\}$	( $r, q$ ) (9,8)=7, (9,10)=7, (10,9)=7, (10,11)=5, (11,10)=5 ( $p, q$ ) (8,14)=9, (9,14)=3, (10,15)=6, (11,15)=3, (12,16)=10
$N^{+4}(s) = \{14, 15, 16\}$	( $r, q$ ) (14,15)=7, (15,14)=7, (15,16)=8, (16,15)=8 ( $p, q$ ) (14,13)=5, (14,18)=3, (15,19)=4, (16,20)=5, (16,21)=2
$N^{+5}(s) = \{13, 18, 19, 20, 21\}$	( $r, q$ ) (19,20)=3, (20,19)=3, (21,20)=3 ( $p, q$ ) (13,17)=1, (18,17)=2, (18,22)=1, (18,23)=8, (19,23)=2, (20,24)=8, (21,25)=5
$N^{+6}(s) = \{17, 22, 23, 24, 25\}$	( $r, q$ ) (22,23)=4, (23,22)=4, (23,24)=1, (24,23)=1, (24,25)=6, (25,24)=6 ( $p, q$ ) (22,26)=3, (23,27)=8, (24,27)=4, (24,28)=5, (25,28)=4
$N^{+7}(s) = \{26, 27, 28\}$	( $r, q$ ) (26,27)=4, (27,26)=4, (27,28)=2, (28,27)=2 ( $p, q$ ) (26,29)=6, (27,30)=3, (28,30)=8
$N^{+8}(s) = \{29, 30\}$	( $r, q$ ) (29,30)=2

P	Q	호	Distance and PSP		
			Calculate of Length	Length	PSP
초기화					
$N^0(s)$	$N^{+1}(s)$	( $p, q$ )	2 : min( $\infty$ , 0+(1,2)=5)=5	$l(1) = 0$	-
		( $r, q$ )	3 : min( $\infty$ , 0+(1,3)=3)=3	$l(2) = 5$	(1,2)=5
		( $p, q$ )	2 : min( $\infty$ , 3+(2,3)=6)=5	$l(3) = 3$	(1,3)=3
		( $r, q$ )	4 : min( $\infty$ , 5+(2,4)=4)=9	$l(4) = 9$	(2,4)=4
		( $p, q$ )	5 : min( $\infty$ , 5+(2,5)=7)=12	$l(5) = 12$	(2,5)=7
		( $r, q$ )	6 : min( $\infty$ , 3+(3,6)=4)=7	$l(6) = 7$	(3,6)=4
$N^{+1}(s)$	$N^{+2}(s)$	( $p, q$ )	7 : min( $\infty$ , 3+(3,7)=8)=11	$l(7) = 11$	(3,7)=8
		( $r, q$ )	5 : min(12, 7+(6,5)=9)=9	$l(5) = 9$	(6,5)=9
		( $p, q$ )	6 : min(7, 12+(5,6)=7)=7	-	-
		( $r, q$ )	8 : min( $\infty$ , 9+(4,8)=13)=13	$l(8) = 13$	(4,8)=4
		( $p, q$ )	9 : min( $\infty$ , 9+(5,9)=2)=11	$l(9) = 11$	(5,9)=2
		( $r, q$ )	10 : min( $\infty$ , 7+(6,10)=7)=14	$l(10) = 14$	(6,10)=7
$N^{+2}(s)$	$N^{+3}(s)$	( $p, q$ )	11 : min( $\infty$ , 11-(7,11)=4)=15	$l(11) = 15$	(7,11)=4
		( $r, q$ )	12 : min( $\infty$ , 11-(7,12)=2)=13	$l(12) = 13$	(7,12)=2
		( $p, q$ )	8 : min(13, 11-(9,8)=7)=13	-	-
		( $r, q$ )	9 : min(11, 14-(10,9)=7)=11	-	-
		( $p, q$ )	10 : min(14, 11-(9,10)=7, 15+(11,10)=5)=14	-	-
		( $r, q$ )	11 : min(14, 15-(10,11)=5, 13+(12,11)=9)=15	-	-
$N^{+3}(s)$	$N^{+4}(s)$	( $p, q$ )	12 : min(13, 15-(11,12)=9)=13	-	-
		( $r, q$ )	14 : min( $\infty$ , 13-(8,14)=9, 9+(9,14)=3)=14	$l(14) = 14$	(9,14)=3
		( $p, q$ )	15 : min( $\infty$ , 14-(10,15)=6, 15+(11,15)=3)=18	$l(15) = 18$	(11,15)=3
		( $r, q$ )	16 : min( $\infty$ , 13-(12,16)=10)=23	$l(16) = 23$	(12,16)=10
		( $p, q$ )	14 : min(14, 18-(15,14)=7)=14	-	-
		( $r, q$ )	15 : min(18, 14-(14,15)=7, 23+(16,15)=8)=18	-	-
$N^{+4}(s)$	$N^{+5}(s)$	( $p, q$ )	16 : min(23, 18-(15,16)=8)=23	-	-
		( $r, q$ )	13 : min( $\infty$ , 14-(14,13)=5)=19	$l(13) = 19$	(14,13)=5
		( $p, q$ )	18 : min( $\infty$ , 14-(10,18)=3)=17	$l(18) = 17$	(14,18)=3
		( $r, q$ )	19 : min( $\infty$ , 18-(15,19)=4)=22	$l(19) = 22$	(15,19)=4
		( $p, q$ )	20 : min( $\infty$ , 23-(16,20)=5)=28	$l(20) = 28$	(16,29)=5
		( $r, q$ )	21 : min( $\infty$ , 23-(16,21)=2)=25	$l(21) = 25$	(16,21)=2
$N^{+5}(s)$	$N^{+6}(s)$	( $p, q$ )	19 : min(22, 28-(20,19)=3)=22	-	-
		( $r, q$ )	20 : min(28, 22+(19,20)=3)-25=(21,20)=3=25	$l(20) = 25$	(19,20)=3
		( $p, q$ )	17 : min( $\infty$ , 19-(13,17)=1, 17+(18,17)=2)=19	$l(17) = 19$	(18,17)=2
		( $r, q$ )	22 : min( $\infty$ , 17-(18,22)=1)=18	$l(22) = 18$	(18,22)=1
		( $p, q$ )	23 : min( $\infty$ , 17-(18,23)=8, 22+(19,23)=2)=24	$l(23) = 24$	(19,23)=2
		( $r, q$ )	24 : min( $\infty$ , 25-(20,24)=8)=33	$l(24) = 33$	(20,24)=8
$N^{+6}(s)$	$N^{+7}(s)$	( $p, q$ )	25 : min( $\infty$ , 25-(21,25)=5)=30	$l(25) = 30$	(21,25)=5
		( $r, q$ )	22 : min(18, 24-(23,22)=4)=18	-	-
		( $p, q$ )	23 : min(24, 22+(22,23)=4, 33-(24,23)=1)=22	$l(23) = 22$	(22,23)=4
		( $r, q$ )	24 : min(33, 24-(23,24)=1, 30+(25,24)=6)=25	$l(24) = 25$	(23,24)=1
		( $p, q$ )	25 : min(30, 33-(24,25)=6)=30	-	-
		( $r, q$ )	26 : min(21, 29-(27,26)=4)=21	-	-
$N^{+7}(s)$	$N^{+8}(s)$	( $p, q$ )	27 : min( $\infty$ , 22+(23,27)=8, 25-(24,27)=4)=29	$l(27) = 29$	(24,27)=4
		( $r, q$ )	28 : min( $\infty$ , 25+(24,28)=5, 30-(25,28)=4)=30	$l(28) = 30$	(24,28)=5
		( $p, q$ )	29 : min( $\infty$ , 21-(23,29)=6)=27	$l(29) = 27$	(26,29)=6
		( $r, q$ )	30 : min( $\infty$ , 25+(27,30)=3, 30-(28,30)=8)=28	$l(30) = 28$	(27,30)=3
		( $p, q$ )	30 : min(28, 27-(29,30)=2)=28	-	-
		( $r, q$ )	30 : min( $\infty$ , 27-(29,30)=2)=28	-	-

$$\text{SP} : l(1,30) = \Sigma \{(1,3) = 3 + (3,6) = 4 + (6,5) = 2 + (5,9) = 2 + (9,14) = 3 \\ + (14,18) = 17 + (18,22) = 1 + (22,26) = 3 + (26,27) = 4 + (27,30) = 3\} = 28$$

이와 같은 결과가 나타난 이유는 Dijkstra 알고리즘은 목적지까지 도달하기 위해 중간 과정에서 경유하는 모든 노드에 대한 최단경로길이를 찾는 방법인데 반해, 제안된 방법은 레벨 단위로 해당 레벨에서의 최단경로 한 지점만에 연결된 다음 레벨의 노드들만을 탐색하여 중간 목적지로 결정하기 때문이다. 결국, 실시간으로 정보를 제공해야 하는 GPS 항법 시스템일 경우 목적지까지의 최단경로를 찾는 시간은 제안된 알고리즘이 Dijkstra 알고리즘에 비해 엄청나게 빠른 결과를 나타내 사용자의 만족도를 크게 향상시킬 수 있을 것이다.

#### IV. 결론 및 향후 연구과제

본 논문에서는 실시간 GPS 항법 시스템에 일반적으로 적용되고 있는 Dijkstra 알고리즘의 문제점을 고찰해보고 새로운 최단경로 탐색 알고리즘을 제안하였다. Dijkstra 알고리즘은 방향 그래프에 대해 출발 노드에서 다른 모든 노드로의 최단경로를 찾는데 성공할 수 있으나 알고리즘이 복잡하고 수행속도가 느린 것이 단점으로 지적되고 있다. 지금까지는 최단 경로 탐색 복잡도를 선형으로 할 수 있는 알고리즘 개발이 미해결 과제로 남아 있다. 본 논문은 이 문제를 해결하는 알고리즘을 제안하였다.

제안된 알고리즘은 먼저 방향그래프의 모든 노드에 대해 외부근방 집합을 탐색하고 이를 레벨 집합간 (Inter-level) 인접 호와 집합 내부 (Intra-level)의 인접 호들에 대해 최소 경로 길이를 계산하는 방식을 택하여 알고리즘 복잡도를 선형에 가깝도록 단축시킬 수 있었다. 제안된 알고리즘을 2개 그래프에 적용한 결과 Dijkstra 알고리즘과 동일하게 출발 노드에서 모든 노드로의 최소 경로 길이와 호들을 선택할 수 있었으며, 수행 횟수를 약 4배 정도 단축시킬 수 있었다.

본 논문에서는 2개의 실제 그래프에 대한 최단경로 탐색을 이론적으로 고찰하였으며, 실제 GPS 항법 시스템에 적용하여 실시간으로 탐색 정보를 제공할 수 있는 실용성을 검증하지는 못하였다. 따라서 추후 실제 GPS 시스템에 적용하여 얼마나 빨리 최단경로를 탐색하여 고객 만족도를 향상시킬 수 있는지 검증하고자 한다.

## 참고문헌

- [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](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](http://www.its.uci.edu/~hyunmyuk/library/(2005)_20EAST(k-path).pdf), 2005.
- [7] F. B. Zhan, "Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures," Journal of Geographic Information and Decision Analysis, Vol. 1, No. 1, pp. 69–82, 1997.
- [8] J. Bang-Jensen and G. Gutin, "Digraphs: Theory, Algorithms and Applications," Springer-Verlag, London, 2006.
- [9] Wikipedia, "Glossary of Graph Theory," [http://en.wikipedia.org/wiki/Glossary\\_of\\_graph\\_theory](http://en.wikipedia.org/wiki/Glossary_of_graph_theory), Wikimedia Foundation Inc., 2007.
- [10] WWL. Chen, "Discrete Mathematics," Department of Mathematics, Division of ICS, Macquarie University, Australia,
- <http://www.maths.mq.edu.au/~wchen/lndmfolder/lndm.html>, 2003.
- [11] R. C. Prim, "Shortest Connection Networks and Some Generalisations," Bell System Technical Journal, Vol. 36, pp. 1389–1401, 1957.
- [12] K. S. Lee, "Shortest Path Algorithm," <http://www.geocities.com/leekinseng1/>, 2008.

## 저자 소개

### 이상운



- 1987년: 한국항공대학교 항공전자공학과 (학사)
- 1997년: 경상대학교 컴퓨터과학과 (석사)
- 2001년 : 경상대학교 컴퓨터과학과 (박사)
- 2003년 : 강원도립대학 컴퓨터응용과 전임강사
- 2004년 ~ 2007.2 : 국립 원주대학 여성교양과 조교수
- 2007.3 ~ 현재 : 강릉원주대학교 멀티미디어공학과 부교수  
<관심분야 : 소프트웨어 프로젝트 관리, 개발 방법론, 소프트웨어 척도, 분석과 설계 방법론, 시험 및 품질보증, 소프트웨어 신뢰성, 신경망, 뉴로-퍼지, 그래프 알고리즘>
- e-mail : [sulee@gwnu.ac.kr](mailto:sulee@gwnu.ac.kr)