Journal of KIIT. Vol. 24, No. 1, pp. 181-189, Jan. 31, 2026. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2026.24.1.181

레벨 목표물 탐색 방법을 적용한 최단경로 알고리즘

이 상 운*

A Shortest Path Search Algorithm using a Method of Level Target Search

Sang-Un Lee*

이 논문은 2025년도 강릉원주대학교 학술연구조성비 지원에 의하여 수행되었음

요 약

차량용 내비게이션에는 일반적으로 Dijkstra 알고리즘(DA)이 적용되고 있다. DA는 출발 지점 s에서 특정 목표 지점 t까지 최단경로로 도달하기 위해 전체 n개 정점 모두에 대한 최단거리를 결정하기 위해 m개 간선 각각의 경로 길이를 모두 계산하여야 한다. 이 논문은 1차적으로 출발 정점부터 목적지 정점까지 레벨을 설정하고 역-레벨 간선을 삭제하여 출발 정점에서 목적지 정점 방향 간선들만 선택하여 망을 축소 시켰다. 다음으로 모든 정점들의 유출 최소 가중치 간선을 2개씩 선택하고, 선택된 간선들 중 최단 경로에 기여하지 못하는 간선들을 3단계에 걸쳐 제거하여 망을 2차로 축소 시켰다. 마지막으로, 단순화된 망의 간선들만을 대상으로 출발 정점에서 목적지 정점으로 최단 길이를 갖는 경로를 선택하였다. 제안된 알고리즘을 8개의 양방향 또는 양방향과 단방향이 혼재된 8~30개 정점 망에 적용한 결과 모든 망에서 최단경로를 빠르면서도 쉽고 정확하게 탐색할 수 있음을 보였다.

Abstract

The Dijkstra's algorithm (DA) is generally applied to vehicle navigation. DA shall calculate the path length of each of the m edges to determine the shortest distance for all n vertices in order to reach the shortest path from the starting point s to the specific target point t. This paper reduced the network by primarily setting the level from the starting vertex to the destination vertex and deleting the reverse-level edge to select only the destination vertex direction edges from the starting vertex. Next, two outdegree minimum-weight edges of all vertices were selected, and the network was reduced to the second order by removing the edges that did not contribute to the shortest path among the selected edges through three stages. Finally, only the edges of the simplified network were selected for the shortest path from the starting vertex to the destination vertex. As a result of applying the level target search algorithm to eight bi-directional or bi-directional and unidirectional mixed 8~30 vertices networks, it was possible to quickly, easily, and correctly search for the shortest path in all networks.

Keywords

shortest path, dijkstra algorithm, point-to-point search, level, reduced network

* 강릉원주대학교 멀티미디어공학과 교수
- ORCID: https://orcid.org/0009-0007-4264-4647

· Received: Sep. 26, 2025, Revised: Nov. 25, 2025, Accepted: Nov. 28, 2025
· Corresponding Author: Sang-Un Lee
Dept. of Multimedia Eng., Gangnueng-Wonju National University, Korea
Tel.: +82-33-760-8688, E-mail: sulee@gwnu.ac.kr

Ⅰ. 서 론

최근 차량용 내비게이션에 최단경로(SP, Shortest Path)를 구하는 Dijkstra 알고리즘(DA)이 일반적으로 적용되고 있다[1]. 차량용 내비게이션은 운전자가 목적지를 결정하면 출발지에서 목적지까지의 최단경로(주행 시간 또는 거리)를 결정하여 화면상에 표시해주고 운전자가 길을 따라 가도록 유도하는 시스템이다. 최단경로를 찾는 알고리즘은 Dijkstra, Bellman Ford, Topological Ordering과 A-Star(A*) 등이 있다[1][2]. DA는 양의 가중치를 갖는 호들로 구성된 방향 망(Directed Network)의 단일 출발점 최단경로(Single- Source Shortest Path)를 찾는 알고리즘이다[3]-[5]. 차량용 내비게이션의 최단경로는 단일 출발점-단일 목적지의 최단경로를 찾는 점대점(P2P, Point-to-Point) 최단경로이다. 점대점 최단경로 탐색 문제도 DA에 기반을 두고 있다[1][6][7]. A* 알고리즘은 f=g+h의 f값이 가장 적은 경로를 선택하여 탐색하는 방법이다. 여기서 g는 출발지에서 현 지점까지의 실제거리이며, h는 현 지점부터 목적지까지의 예상거리이다. 이 방법은 DA에 비해 보다 빠른 방법으로 알려져 있다. 다만 경로 거리가 멀수록 최적경로를 선택하기 위해 탐색된 예비경로들의 집합인 Open List의 크기가 증가하여 예비경로들의 비교연산 또한 증가하는 단점을 갖고 있다. 이로 인해 Y. H. Lee and S. W. Kim[8]은 DA와 A*의 하이브리드 방법을 제안하기도 하였다.

도시의 도로 구조는 무 방향(양방향)과 단방향(일방 통행로)가 혼재되어 있다. 이러한 도로 구조에 대해 차량용 내비게이션으로 점대점 최단경로를 찾는데 있어 DA보다 메모리를 적게 요구하면서 빠른 알고리즘을 구현할 필요가 있다. 최단경로를 찾는 알고리즘에 관한 연구는 S. U. Lee가 있다[9]-[13]. S. U. Lee[9][10]는 주어진 망을 출발지에서 목적지로의 점대점 최단경로를 찾기 위해 출발지를 근 노드로 레벨 단위로 재배치하여 BFS(넓이우선탐색)로 목적지를 최단경로로 찾아가는 방법을 연구하였으며, S. U. Lee[11]는 MST(최소신장트리)를 구성하여 점대점으로 목적지를 찾아가는 방법을 제안하였다, S. U. Lee[12][13]는 특정 도로에서 병목현상이 발생하였다는 실시간 상황에서 목적지까지 최단시간 경로를 찾아가는 방법을 제안하였다.

본 논문은 메모리 용량을 최소로 하면서도 간단한 방법으로 점대점 최단경로를 찾을 수 있는 알고리즘을 제안한다. 2장에서는 DA의 한계를 고찰한다. 3장에서는 제안하는 레벨 단위 목표물 탐색 알고리즘의 상세 구조를 제시한다. 4장에서는 제안된 알고리즘을 다양한 망에 적용하여 성능 및 적합성을 검증한다.

Ⅱ. 관련 연구와 연구 배경

무 방향 망 $G=(V,E)$은 정점들(Vertices, V)과 간선들(Edges, E)로 구성되어 있으며, 정점 들(교차로, 지점 등)이 무 방향(양방향)의 간선(도로)들로 연결되어 있다. 따라서 두 정점 x와 y간에 연결된 간선은 $\{x,y\}=\{y,x\}$로 양방향 통행이 가능하다. 방향 망 $G=(N,A)$는 노드(Nodes, N)와 호 (Arcs, A)로 구성되어 있으며, 두 노드 x와 y를 연결하는 호는 방향성으로 $(x,y)\neq(y,x)$로 순서쌍으로 표기한다. 망에서 간선 또는 호는 가중치 (거리 또는 주행시간)를 갖고 있다.

방향 망의 단일 출발 노드 최단경로를 찾는 대표적인 알고리즘인 DA는 Lee[3]-[5]에서 인용되었다. DA는 “특정 노드로부터 시작하여 최소 경로 길이를 갖는 노드를 한 번에 하나씩 선택하는 방식”으로,특정 노드에 인접한 노드들과 이전에 계산된 노드들의 경로의 합이 최소가 되는 노드를 찾는다. 따라서 출발 노드로부터 시작하여 다른 모든 노드들을 방문하는 최단경로를 찾는데 $|n|-1$회 수행된다. DA를 무 방향 망에 적용하기 위해서는 노드를 정점으로, 호를 간선으로 변환하면 된다.

DA를 그림 1의 G_1 무 방향 망에 대해 적용하여 보자.

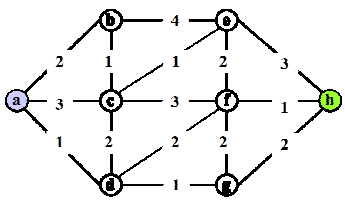


그림 1. G_1 망
Fig. 1. G_1 network

G_1 망은 Chen[14]에서 인용되었으며, ⓐ에서 출발하여 ⓗ에 도착하는 최단경로를 찾는 문제이다. 출발 정점에서 목적지 정점까지 최단거리를 갖는 경로를 DA로 찾는 과정은 그림 2와 같이 a→d→b→g→c→f→h→e의 경로길이가 최소인 다음 정점으로 여기저기 움직이면서 모든 정점의 최단경로를 갱신하는 전역탐색 법이라 할 수 있다. a로부터 다른 모든 정점까지의 최단경로가 결정된다. 따라서 모든 정점(노드)까지의 최단경로는 알고리즘이 종료된 시점에서만 알 수 있다.

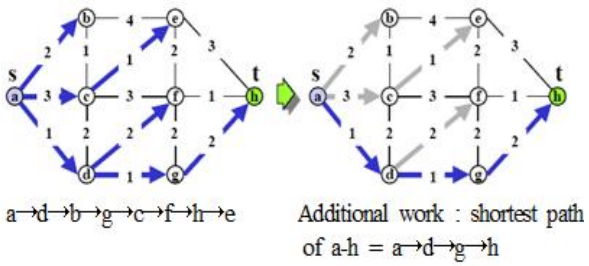


그림 2. G_1 망의 Dijkstra 알고리즘 적용
Fig. 2. Apply Dijkstra's algorithm to G_1 network

알고리즘이 종료된 이후 추가적으로 a-h의 최단경로를 결정한다. DA는 항상 목적지까지의 최단경로를 찾는데 성공할 수 있다. 그러나 단점은 모든 정점을 대상으로 해당 정점에 연결된 모든 정점들의 경로를 기존에 계산된 경로 길이와 비교하여 현재까지의 최단경로를 계산함으로 인해, 기존에 계산된 최단경로 정보를 모두 갖고 있어 많은 양의 메모리를 요구하며, 일반적으로 모든 노드의 최단경로 선택에 $|v|-1$회를 수행하기 때문에 망이 복잡해질수록 수행속도가 느리다. 즉, DA는 n개 정점 각각이 m개 간선 중에서 자신에게 부속된 간선들을 모두 저장하고, 각 간선들 까지의 경로길이 정보도 함께 보관한 상태에서 n개 정점 모두에 대해 최단경로를 결정하는 방식이다.

이러한 DA의 단점을 극복한 방법이 Lee의 레벨 단위로 노드(정점)를 BFS로 탐색하는 방법이다. 이 방법은 출발지를 근 노드로, 이웃들을 레벨로 설정하여 레벨 단위로 부 노드로 부터의 최단경로를 결정하고, 제노드 경로와 비교하여 해당 노드의 최단경로를 확정하는 지역탐색법이라 할 수 있다[9]-[13]. 따라서 특정 노드에 연결된 이웃 노드들의 경로 정보와 이웃들 간의 간선 거리 정보만 알고 있으면 최단경로를 확정할 수 있어 메모리를 획기적으로 축소시킬 수 있다. 반면에 이들 알고리즘의 단점은 출발지에서 목적지까지 도달 가능한 모든 경로를 대상으로 계산하여야만 한다는 점이다. 복잡한 망(도시)의 경우 이러한 알고리즘도 수행시간이 길어진다는 단점이 있다. 따라서 복잡한 망에서 목적지까지 최단경로를 빠르게 탐색하기 위해서는 수행횟수를 단축시킬 수 있는 알고리즘이 절실히 요구된다. 3장에서는 망의 점대점 최단거리를 찾기 위해 요구되는 메모리 크기도 줄이면서 수행속도를 월등히 향상시킬 수 있는 알고리즘을 제안한다.

III. 적응형 레벨 목표물 최단경로 알고리즘

임의의 출발지(s)에서 목적지(t)로의 최단경로를 찾는 방법은 어느 방향의 어디 쯤에 목적지가 있는지 모르는 상황에서 방향과 거리를 모두 찾아야만 한다. 따라서 주어진 망에서 목적지까지 찾기 위해서는 모든 방향의 정점들 까지의 최단경로를 찾아야만 하며, 이러한 최단경로들을 구하는 단순한 방법으로 본 장에서는 레벨 단위(인접 단계)로 찾으면서도 출발지에서 목적지로 도달 가능한 중간 경로 노드들 중 불필요한 노드들을 제거하고 필수적인 노드들만을 대상으로 최단경로를 찾는 방법을 제안한다. 제안된 방법은 현재 위치한 레벨 정점에서 다음 레벨로 2개의 최단거리 간선을 선택하고 이들이 경쟁하여 최단경로 길이 정점을 선택하는 방식으로 양자택일 방법이라고 할 수 있다. 따라서 메모리는 $2l$의 간선 경로 길이만 필요하다. 여기서 l은 주어진 망의 출발지에서 목적지까지의 경유 정점인 레벨 수이다.

망은 양방향(무 방향) 또는 양방향과 단방향이 혼재된 도로들로 구성되어 있다고 가정한다. 제안되는 알고리즘은 출발지에서 목적지로 레이더 전파를 발사하여 허상(목적지 정점까지 도달하는데 경유하지 않아도 되는 불필요한 정점들)을 제거하고 원하는 목표물들(목적지 정점까지 도달하는데 경유하는 정점들) 만을 탐색(선택)하는 방법과 유사하다. 레이더 전파의 탐지거리는 목적지 정점까지이며, 경로 정점들은 레벨 단위로 탐색한다. 따라서 제안되는

알고리즘을 그림 3과 같이 “레벨 목표물 탐색” 방법이라 하며 세부적으로는 그림 4와 같이 망의 단순화 과정이 수행된다.

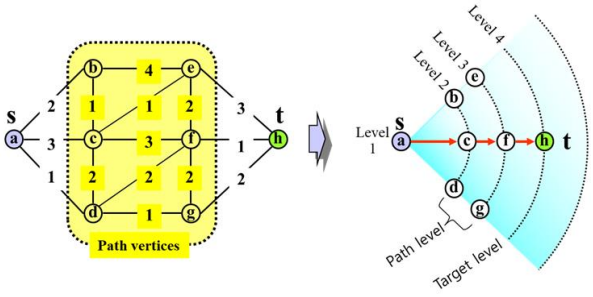


그림 3. G_1 망의 레벨 목표물 탐색
Fig. 3. Level target search for G_1 network

첫 번째로, 출발지 s에서 목적지 t까지 도달하기 위해 경유하는 정점들을 최소화 시키는 기법을 다음과 같이 제안한다. 이와 같이 하는 이유는 s에서 t까지 오로지 순방향(Feed forward)으로만 탐색하기 위함이다.

(1) 출발지 s에서 목적지 t로의 최단 경로는 경유하는 지점들을 최소화 시켜야 하며, 목적지 t에서 출발지 s로의 역주행 방향 도로들을 사전에 제거해야만 한다. 이 문제를 해결하는 가장 적합한 방법으로 망 레벨 방법을 적용한다. 만약, 목적지 정점이 다른 경로 정점들과 동일한 레벨에 존재하면 레벨을 하나 추가하여 목적지 정점을 마지막 레벨로 이동시킨다.

두 번째로, 한 정점에 인접한 많은 수의 간선들 중 최단경로로 선택될 가능성이 높은 간선을 선택하는 문제는 다음 방법을 적용한다. 이와 같이 하는 이유는 탐색 경로 상의 각 정점에서의 최단거리 간선들만으로 탐색하면 목적지까지의 거리가 오히려 길어질 수 있기 때문에 본 논문에서는 2개 최단거리 간선 간에 경쟁하는 양자택일 법을 적용하여 이러한 문제점을 해결하였다.

(2) i^{th}레벨에서 $i+1^{th}$레벨로 도달할 수 있는 최단경로는 i^{th}레벨의 한 정점 유출 간선 들 중 1^{st} MWE(Minimum Weight Edge)와 2^{nd} MWE 중 어느 하나로 결정된다. 따라서 모든 정점의 유출 간선들 중 1^{st} MWE와 2^{nd} MWE를 선택한다.

세 번째로, 출발지에서 목적지로 갈 수 있는 간선들이 충분히 선택된 상태이므로 불필요한 간선들을 삭제하여 망을 단순화하는 과정이 필요하다. 이는 레이더로 전파를 발사하여 탐색된 레벨 단위의 목표물 중 허상을 제거하는 과정으로 다음 방법을 적용한다. (2)와 (3)은 적응형 탐색법(ASM, Adaptive Search Method)이라 할 수 있다.

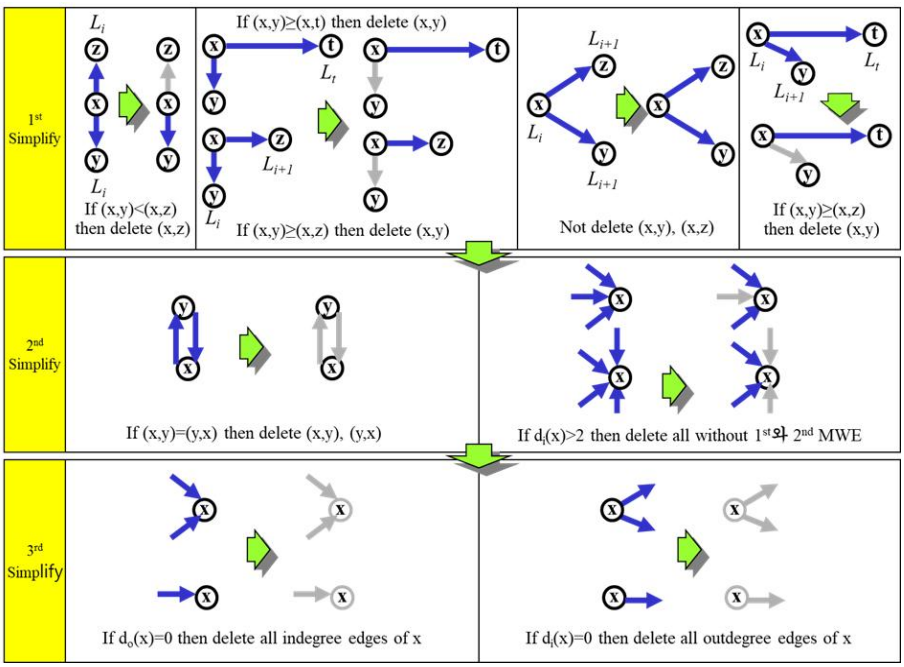


그림 4. 망 단순화 과정
Fig. 4. Simplifying process of network

(3) 한 정점에서 선택된 2개의 MWE 중 그림 4와 같이 3단계의 경우가 발생하면 최대 가중치 간선을 삭제한다. 예로, 한 정점 x에서 동일 레벨의 간선 $\{x,y\}$와 $\{x,z\}$가 있는 경우, $w\{x,y\}<w\{x,z\}$이면 $w\{x,z\}$는 최단경로에 기여하지 못하기 때문에 삭제가 가능하다.

마지막으로, 선택된 간선들을 대상으로 출발지 s에서 목적지 t까지 도달할 수 있는 경로들 중 최단경로를 다음과 같이 결정한다. 이와 같이 결정한 이유는 다양한 망에 적용한 실험 결과 경로 계산 우선순위를 준수하여야만 정확한 최소 경로 길이를 얻을 수 있기 때문이다.

(4) 유출 간선이 2개 이상인 정점($d_o(x)\geq 2$)에서 유입 간선이 2개 이상인 정점($d_i(i)\geq 2$) 간의 폐쇄경로(Closed Path)에 대해 길이의 합이 최소인 경로를 선택한다. 계산 우선순위는 {u,v} 간선에 대해 "{경로,경로}>{출발지,경로}>{경로,목적지}>{출발지,목적지}"이다.

위와 같이 방법을 적용한 "적응형 레벨 목표물 최단경로 알고리즘(Adaptive level target shortest path algorithm, ALTSPA)"은 그림 5에 제시되어 있다.

G_1 망에 레벨 목표물 탐색 방법을 적용한 과정은 그림 6에 제시되어 있다. 알고리즘 적용 결과, DA는 각 정점의 최단경로 선택에 $|v|-1=7$회 수행하였으며, 인접 정점들의 최단 길이 합을 반복적으로 갱신하고 최단 길이 정점을 선택하였다.

[Setting level from start point to target point]
Set the start point to level 1, then setting the network level neighboring vertices to next level until target point..
if ∃ target vertex in same level with path vertices then increasing level=level+1, move target point to final level.
Create a $v\times v$ adjacency square matrix in level order.
Delete reverse edges $l_1, l_2, \cdots, l_{i-1}$ in i^{th} level.

[Choose edges]
Select 1st MWE and 2nd MWE about start and path vertices.
if 1st MWE=multiple equal values then do not select 2nd MWE.
if 1st MWE=only 1, 2nd MWE= multiple then select all 2nd MWE.

[Simplify]
(1st simplify): for the $\{x,y\},\{x,z\}$ of path level $(l_i, i\geq 2)$
/* same level-same level
if $\{l_i(x),l_i(y)\},\{l_i(x),l_i(z)\}$ then
if $\{l_i(x),l_i(y)\}>\{l_i(x),l_i(z)\}$ then delete $\{l_i(x),l_i(y)\}$
/* same level-next level
else if $\{l_i(x),l_i(y)\},\{l_i(x),l_{i+1}(z)\}$ then
if $\{l_i(x),l_i(y)\}\geq\{l_i(x),l_{i+1}(z)\}$ then delete $\{l_i(x),l_i(y)\}$
/* same level-target level
else if $\{l_i(x),l_i(y)\},\{l_i(x),l_t(z)\}$ then
if $\{l_i(x),l_i(y)\}\geq\{l_i(x),l_t(z)\}$ then delete $\{l_i(x),l_i(y)\}$
/* next route level-next route level
else if $\{l_i(x),l_{i+1}(y)\},\{l_i(x),l_{i+1}(z)\}$ then Skip
/* next route level-target level
else if $\{l_i(x),l_{i+1}(y)\},\{l_i(x),l_t(z)\}$
if $\{l_i(x),l_{i+1}(y)\}\geq\{l_i(x),l_t(z)\}$ then delete $\{l_i(x),l_{i+1}(y)\}$.
(2nd simplify)
if $\exists(\{x,y\},\{y,x\})$ then delete $\{x,y\},\{y,x\}$.
if $\forall d_o(x)>2$ vertices then delete all edges without 1st와 2nd MWE.
However, if there are multiple 2nd MWEs, all are left.
(3rd simplify)
if vertex with no inflow or outflow edges then delete all vertex edges..
/* If this happens in succession, continue to delete.

[Determining the shortest path]
Select shortest path from a vertex with at least two outflow edges $(d_o(x)>2)$ to a vertex with at least two inflow edges$(d_i(y)>2)$. /* Path calculation priority: Route-Route>Start-Route>Route-Target>Start-Target path.

그림 5. 적응형 레벨 목표물 최단경로 알고리즘
Fig. 5. Adaptive level target shortest path algorithm(ALTSPA)

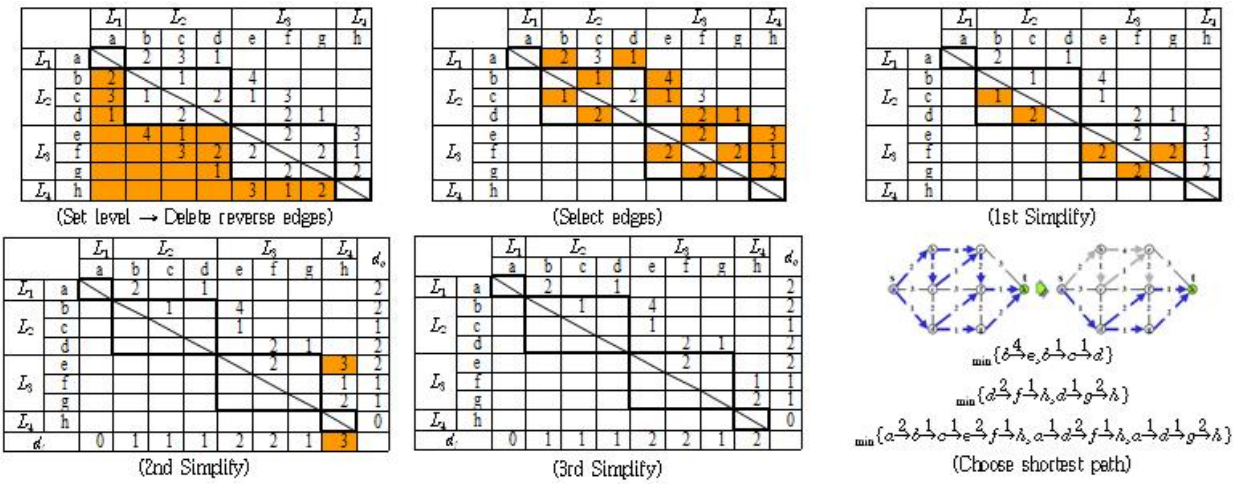


그림 6. G_1 망의 적응형 레벨 목표물 탐색 최단경로 알고리즘 적용
Fig. 6. Apply adaptive level target search shortest path algorithm to G_1 network

반면에 제안된 알고리즘은 단지 레벨 설정, 간선 선택과 3단계의 단순화 과정 등 5회 선택만 수행하면서도 정점들의 최단 경로 길이를 갱신하는 복잡한 계산을 수행하지 않는다. 제안된 알고리즘은 단지 최단경로 결정 단계에서만 경로 길이 합을 계산하는 단순한 알고리즘으로 DA에 비해 월등히 적은 메모리를 사용하며, 복잡한 계산과정을 수행하지 않는 장점이 있다.

Ⅳ. 알고리즘 적용성 평가

본 절에서는 그림 7의 7개 망을 대상으로 알고리즘의 적용성을 평가해 본다. 이들 망들은 또한 Lee[8-12] 논문에서도 벤치마킹 데이터로 인용되어 있다. G_2에서 G_7 망은 양방향 (무 방향) 망이며, G_8 망은 양방향과 단방향 (일방통행)이 혼합된 망이다. 각 망의 출발과 목적지 정점은 G_2 망은 a-k, G_3 망은 a-z, G_4 망은 1-6, G_5 망은 1-6, G_6 망은 a-l, G_7 망은 1-9, G_8 망은 1-30이라 가정한다.

제안된 알고리즘의 적합성 여부를 비교하기 위해, 그림 7의 7개 망에 DA를 적용하여 최단경로를 찾는 과정은 복잡하여 생략하였으며, 단지 결과는 그림 8에 제시하였으며, 본 논문에서 제안된 레벨 목표물 탐색 최단경로를 구한 결과는 그림 9에 제시하였다.

7개 망에 본 논문에서 제안된 레벨 목표물 탐색 최단경로 알고리즘을 적용한 결과 모든 망에서 최단경로를 1개 또는 2개를 찾는데 성공하였으며, DA에 비해 수행 횟수를 크게 줄이는 효과를 얻었다.

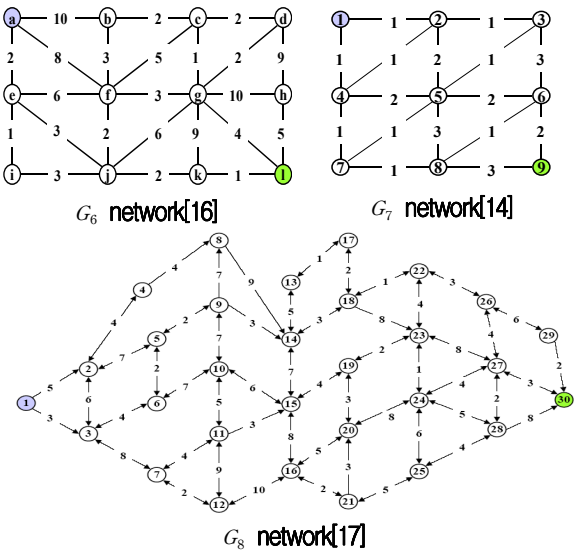


그림 7. 실험 벤치마킹 망
Fig. 7. Experimental benchmark network

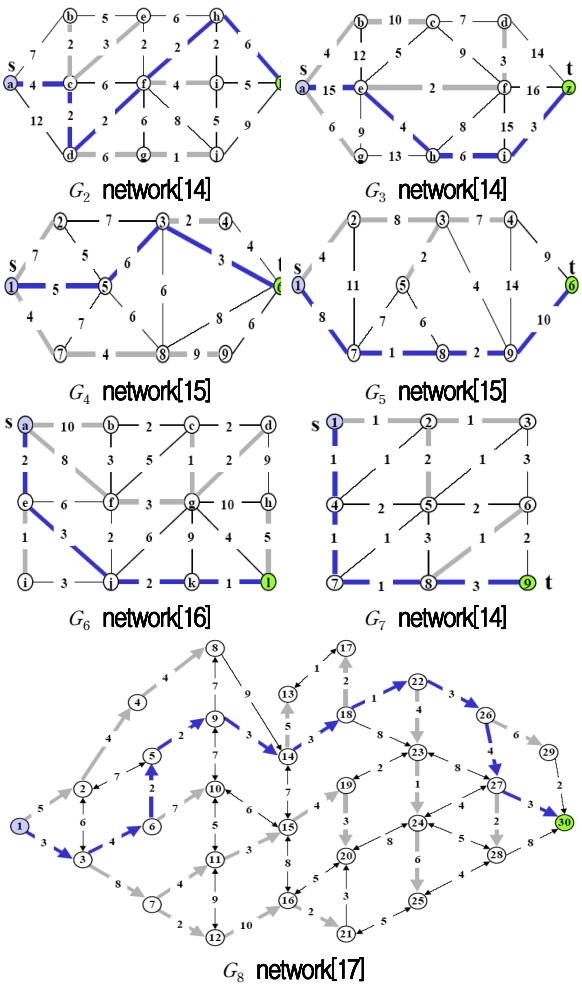
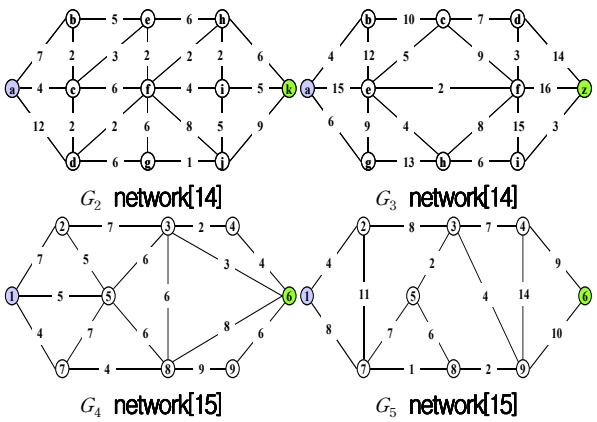
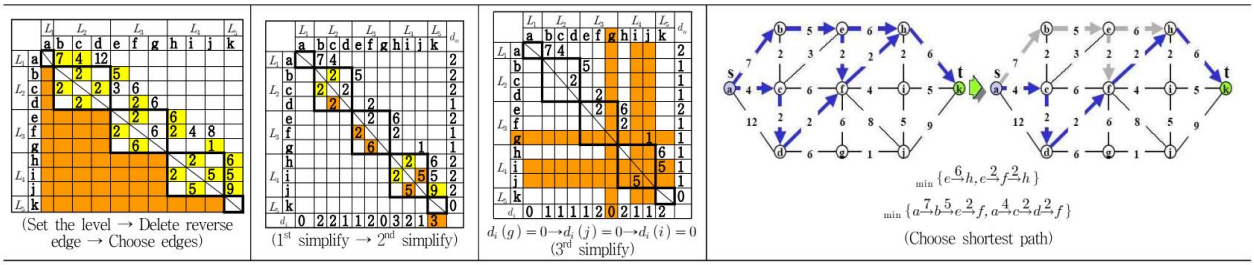
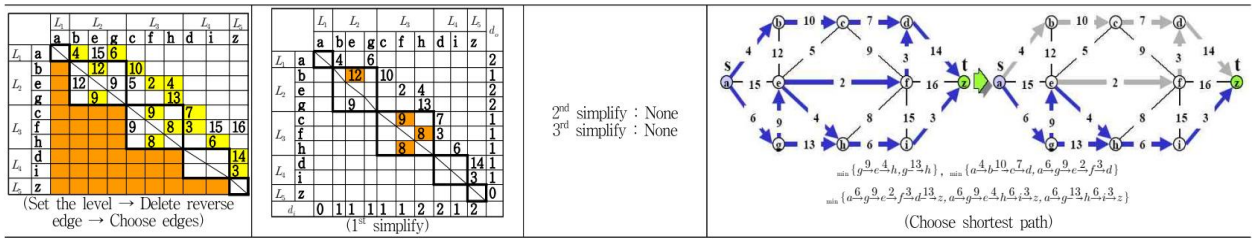


그림 8. Dijkstra 알고리즘의 최단경로
Fig. 8. Shortest path of Dijkstra's algorithm

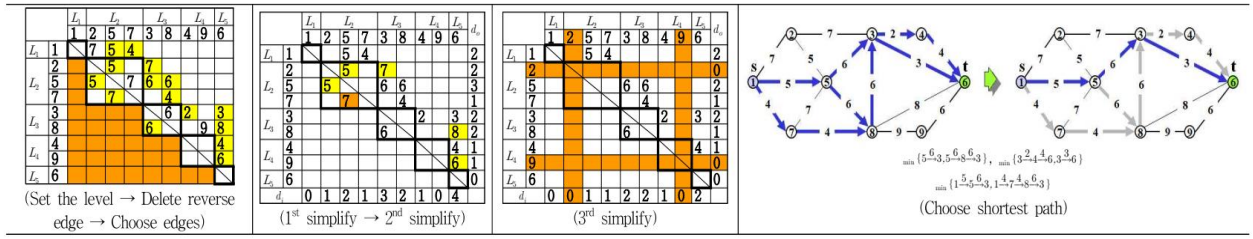




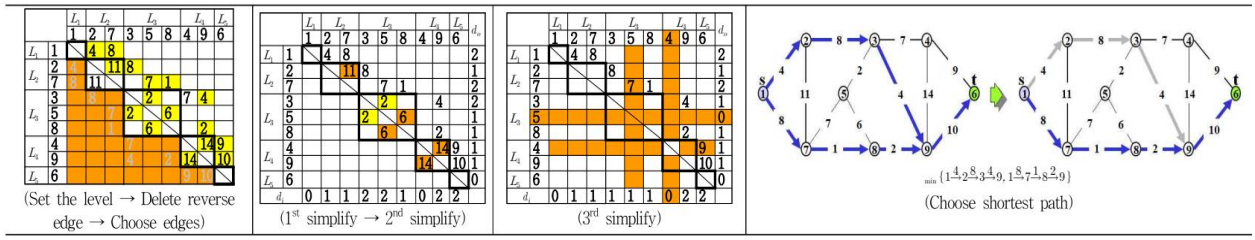
(a) G_2 network



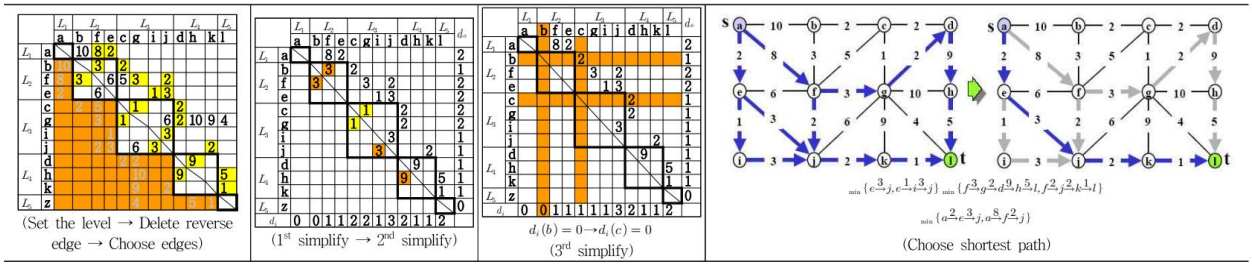
(b) G_3 network



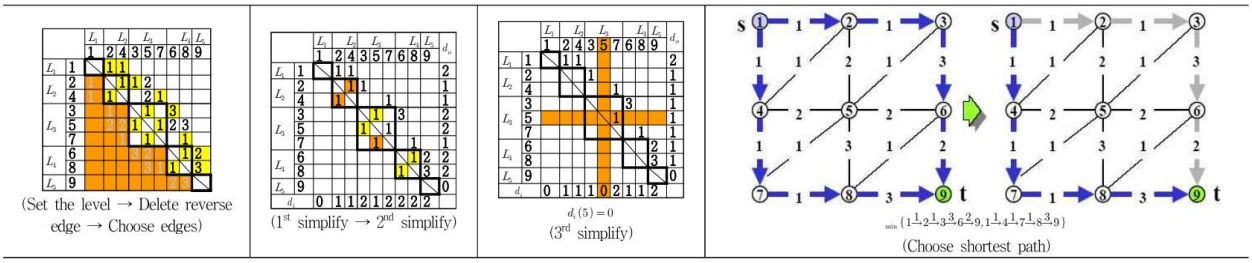
(c) G_4 network



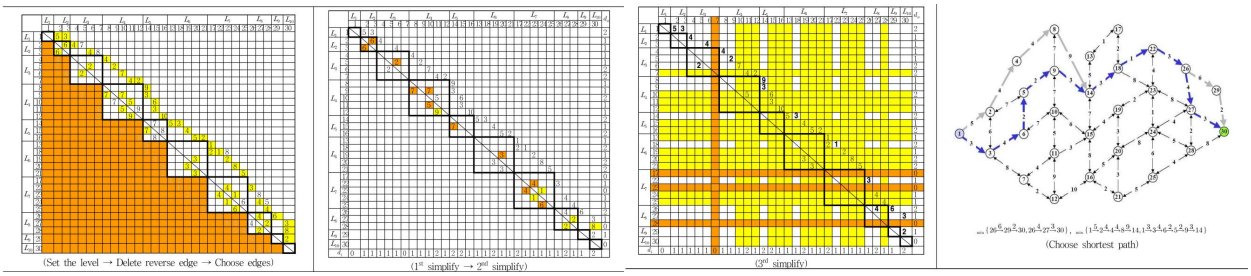
(d) G_5 network



(e) G_6 network



(f) G_7 network



(g) G_8 network

그림 9. 적응형 레벨 목표물 탐색 최단경로

Fig. 9. Shortest path of adaptive level target search

V. 결론 및 향후 과제

본 논문에서는 차량용 내비게이션의 출발지에서 목적지로의 점대점 최단경로를 찾는 DA의 문제점을 도출하여 이를 해결한 새로운 최단경로 탐색 알고리즘을 제안하였다. DA는 출발 정점에서 다른 모든 정점으로의 최단경로를 찾는다. 그러나 차량용 내비게이션에 적용되는 최단경로는 출발에서 목적지 정점까지의 점대점 최단경로 만을 대상으로 정보를 제공해야만 한다.

본 논문에서는 양방향(무방향)만으로 구성된 망과 양방향과 단방향(일방통행로)가 혼합된 망에 대해 최단경로를 쉽고, 빠르게 찾는 레벨 목표물 탐색 방법을 제안하였다. 제안된 알고리즘은 출발에서 목적지까지 망의 레벨을 설정하고, 역-레벨 간선들을 삭제한 후, 각 정점의 최소 가중치 간선을 2개씩 선택하고 불필요한 간선을 삭제하는 방법을 적용하였다. 이와 같이 적응형으로 단순화된 간선들을 대상으로 출발에서 목적지까지의 최단경로를 결정하였다. 8개의 다양한 망에 적용한 결과, 모든 망에서 최단경로를 빠르고, 간단하게 결정하는데 성공하였다. DA는 항상 $|v|-1$회 수행으로 모든 정점들의 최단경로를 결정하는데 반해 제안된 알고리즘은 레벨설정, 역-레벨 간선 삭제, 간선 선택, 3차의 단순화 과정만을 수행하면 단순화된 간선들을 얻을 수 있다. 따라서 제안된 알고리즘은 DA에 비해 메모리 크기도 적게 요구하면서, 수행속도도 월등히 빠른 효과를 얻었다. 또한, 제안된 알고리즘은 간선의 가중치를 거리 뿐 아니라 주행시간으로 변환하면 실시간 교통정보를 활용한 TPEG(Transport Protocol Expert Group) 내비게이션에도 활용이 가능하다.

본 논문에서는 최단경로 탐색 알고리즘을 가상의 망에 대해서만 이론적으로 적용성을 평가하는데 한정하였으며, 실제 차량용 내비게이션에 탑재하여 실용성을 검증하지는 못한 부분은 실제 차량용 내비게이션에 적용하여 실용성 여부를 검증하는 추후 연구할 예정이다.

References

[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", Information Processing Letters, Vol. 77, No. 2-4, pp. 197-200, Feb. 2001. https://doi.org/10.1016/S0020-0190(00)00202-7.

[6] Y. T. Lim and H. M. Kim, "A Shortest Path Algorithm for Real Road Network Based on Path Overlap", Journal of the Eastern Asia Society for Transportation Studies, Vol. 6, pp. 1426-1438, 2005. https://doi.org/10.11175/easts.6.1426.

[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] Y. H. Lee and S. W. Kim, "A Hybrid Search Method of A* and Dijkstra Algorithms to Find Minimal Path Lengths for Navigation Route Planning", Journal of the Institute of Electronics and Information Engineers, Vol. 51, No. 10, pp. 109-117, Oct. 2014. https://doi.org/10.5573/ieie.2014.51.10.109.

[9] S. U. Lee, "A Point-to-Point Shortest Path Search Algorithm for Digraph", Journal of the Korean Institute of Fuzzy and Intelligent Systems, Vol. 17, No. 7, pp. 893-900, Dec. 2007. https://doi.org/10.5391/JKIIS.2007.17.7.893.

[10] S. U. Lee, "A Point-to-Point Shortest Path Algorithm Based on Level Node Selection", Journal of the Institute of Internet, Broadcasting and Communication, Vol. 12, No. 1, pp. 133-140, Feb. 2012. https://doi.org/10.7236/JIWIT.2012.12.1.133.

[11] S. U. Lee, "A Point-to-Point Shortest Path Search Algorithm in an Undirected Graph Using Minimum Spanning Tree", Journal of The Korea Society of Computer and Information, Vol. 19, No. 7, pp. 103-111, Jul. 2014. https://doi.org/10.9708/jksci.2014.19.7.103.

[12] S. U. Lee, "A Real-time Point-to-Point Shortest Path Search Algorithm Based on Traveling Time", Journal of the Institute of Internet, Broadcasting and Communication, Vol. 12, No. 4, pp. 131-140, Sep. 2012. https://doi.org/10.7236/JIWIT.2012.12.4.131.

[13] S. U. Lee, "A Real-time Shortest Path Search for Navigation Based on Traveling Time Using the Minimum Spanning Tree", Journal of KIIT, Vol. 12, No. 8, pp. 1-8, Aug. 2014. https://doi.org/10.14801/kitr.2014.12.8.1.

[14] WWL. Chen, "Discrete Mathematics", Department of Mathematics, Division of ICS, Macquarie University, Australia, 2003.

[15] R. Wenger, "CIS 780: Analysis of Algorithms", 2004. http://www.cse.ohio-state.edu/~wenger/cis780/shortest_path.pdf. [accessed: Aug. 10, 2025]

[16] C. Peiper, "CS 400 - Data Structures for Non CS-Majors", 2005. http://www.cs.uiuc.edu/class/fa05/cs400/_labs/Lab12/suuri/. [accessed: Aug. 10, 2025]

[17] K. S. Lee, "Shortest Path Algorithm", 2008. http://www.geocities.com/leekinseng1/. [accessed: Aug. 10, 2025]

저자소개

이 상 운 (Sang-Un Lee)

1987년 2월 : 한국항공대학교 항공전자공학과(학사)

1997년 6월 : 경상대학교 컴퓨터과학과(석사)

2001년 2월 : 경상대학교 컴퓨터과학과(박사)

2003년 3월 : 강원도립대학 컴퓨터응용과 전임강사

2004년 3월 ~ 2007년 2월 : 국립 원주대학 여성교양과 조교수

2007년 3월 ~ 2015년 3월 : 강릉원주대학교 멀티미디어공학과 부교수

2015년 4월 ~ 현재 : 강릉원주대학교 멀티미디어공학과 정교수

관심분야 : 소프트웨어 프로젝트 관리, 소프트웨어 개발 방법론, 소프트웨어 분석과 설계 방법론, 소프트웨어 신뢰성, 인공지능, 빅데이터 분석, NP-완전 문제 최적화 알고리즘, 퍼즐