한국ITS학회논문지
제12권, 제2호 (2013년4월)
pp.12~21
http://dx.doi.org/10.12815/kits.2013.12.2.012

최단경로 탐색영역 축소 알고리즘 개발

Development of Shortest Path Searching Network Reduction Algorithm

유 영 근*
(Ryu, Yeong-Geun)

요 약

본 연구에서는 최단경로 탐색 소요시간을 줄이기 위한 목적으로 탐색영역을 축소하는 알고리즘을 개발하였다. 개발된 알고리즘은 출발노드와 목적노드를 최소의 노드 수로 연결하면서 직선거리의 합이 최소인 임시경로를 구축하고, 구축된 임시경로의 부하량 보다 적은 부하량의 경로를 연결할 가능성이 있는 노드들을 찾는 것이다. 하나의 노드에서 출발노드까지의 직선거리와 목적노드까지의 직선거리 합이 임시경로의 부하량을 최소 가로 부하량 원단위로 나눈 값 보다 적을 경우, 그 노드는 임시경로 보다 더 적은 부하량을 가질 수 있는 경로를 구성할 가능성이 있는 노드가 된다. 이와 같은 노드들만을 탐색영역으로 하면 탐색영역이 축소됨에 따라 최단경로 탐색 소요시간을 줄일 수 있게 된다. 개발된 알고리즘은 큰 탐색영역에서 출발노드와 목적노드가 가까울 경우 더욱 효과적이다.

핵심어 : 최단경로탐색 알고리듬, 경로안내시스템, 탐색영역, 탐색시간, 다익스트라 알고리듬

Abstract

This study developed searching network reduction algorithm for reduce shortest path searching time. Developed algorithm is searching nodes that have the including possibility of less weights path than temporal path that consists minimum number of nodes and minimum sum of the straight line distances. The node that has the including possibility of shortest path is the node that the sum of straight line distance from start node and straight line distance to target node is less than the value that temporary path's weights divided by minimum weights units. If searching network reconstitutes only these nodes, the time of shortest path searching will be reduced. This developed algorithm has much effectiveness that start node and target node is close in large network.

Keywords : Shortest path algorithm, RGV, Searching area, Searching time, Dijkstra algorithm

Ⅰ. 서 론

최단경로 탐색에 관한 연구는 오랜 기간에 걸쳐 지속적으로 이루어지고 있다. 최단경로 탐색 알고리즘의 연구와 개발이 지속적으로 이루어지는 것은 물류를 포함하는 교통분야 만이 아니라 통신, 컴퓨터, 게임산업, 산업공정 등 많은 분야에서 요구되고 있기 때문이다.

특히 교통분야에서는 GPS(Global Positioning System)와 GIS(Geographic Information System)의 발달과 함께 RGS(In-Vehicle Route Guidance Systems)의 급속한 발달이 있었고, 지속적인 개선과정에서

* 주저자 : 영남교통정책연구원 원장
† 논문접수일 : 2013년 3월 00일

빠른 최단경로 탐색 알고리즘의 연구가 활발히 진행되고 있다[1,2].

RGS는 현재 차량의 위치에서 목적지까지 최소 통행시간 경로를 찾아 제시하는 시스템으로 Dijkstra (1959) 알고리즘을 기초로 하고 있다[1].

Dijkstra(1959) 알고리즘[3]은 정적인 부하량(통행시간, 통행비용 등)을 대상으로 하여 최단경로를 탐색하고 있다. 가로의 소요시간으로부터 최단경로를 탐색하는 RGS는 소요시간의 잦은 변동으로 Dijkstra 알고리즘을 적용하기에는 어려운 점이 있다.

Dijkstra 알고리즘을 일부분 수정하거나 응용하여 RGS 적용 가능성을 높인 연구들도 있다[4,5,6,7].

그러나, Hall(1986)은 가로 통행시간을 부하량으로 하여 최단경로 탐색을 처음으로 연구하였는데, 실패할 수도 있음을 밝혔다[8].

결국, RGS에서는 정확한 최단경로를 빠르게 탐색해야 성공적인 RGS가 될 수 있다. 물론 RGS에서만이 아니라 최단경로 탐색을 필요로 하는 모든 분야에서 빠른 탐색을 위한 알고리즘의 개발을 하고 있다.

네트워크 내에서 목적지까지의 최단경로를 정확하면서 빠르게 탐색하기 위해서는 네트워크 특성에 맞는 효율적인 탐색 알고리즘의 적용이 필요하다.

또한 네트워크를 출발지와 목적지를 중심으로 하여 최대한 줄일 수 있다면 빠른 최단경로 탐색이 가능하게 될 것이다.

예로, 서울특별시 전체 네트워크에서 서울시청을 출발지, 서울역을 목적지로 하여 최단경로 탐색을 한다면 탐색의 필요성이 전혀 없는 영역이 상당히 크다는 것을 알 수 있다. 만약 국토 전체 네트워크에서 찾는다면 불필요한 영역이 실제로 엄청나게 되고, 탐색을 위한 메모리의 양과 탐색시간이 상당히 길어지게 된다.

본 연구에서는 최단경로 탐색소요시간을 줄이기 위하여 출발지와 목적지에 따라 반드시 탐색이 필요한 노드(최단경로 구성 가능 노드)들 만을 추출하는 알고리즘을 개발하였다.

Ⅱ. 탐색영역 축소를 위한 방안

1. 기초 개념

탐색영역 축소를 위한 알고리즘을 개발하기 위하여 다음 항목들을 기본적으로 고려하였다.

- 직선거리와 부하량(실제거리, 통행소요시간, 통행비용 등)은 비례관계에 있다.
- 목적지와 같은 방향성을 가지는 노드가 최단경로에 포함될 가능성이 높으나, 소요시간 등의 부하량을 적용하는 경우에는 방향성이 맞지 않는 우회경로가 최단경로가 될 수도 있다.
- 노드수를 최소로 하는 경로탐색이 가장 빠르다. 이 탐색은 가로(Link) 부하량을 비교하지 않기 때문이다.

2. 탐색영역 축소를 위한 접근방법

출발노드에서 목적노드까지 간단하고 빠르게 하나의 경로(임시경로)가 구축된다면 그 구축된 경로는 최단경로일 수도 있고, 아닐 수도 있다.

임시경로가 가지는 부하량보다 더 적은 부하량을 가질 수 있는 경로에 포함될 가능성이 있는 노드들만으로 탐색영역을 구성한다면 탐색영역은 축소될 것이고, 최단경로 탐색 소요시간도 감소할 것이다.

탐색영역 축소 기준이 되는 임시경로는 다음 두 가지의 조건을 만족시킬 필요가 있다.

- 출발노드에서 목적노드까지 빠른 구축
- 실제 최단경로에 근접

임시경로를 빠르게 구축하기 위해서는 가로 부하량을 비교하지 않고 목적노드까지의 경로를 구축할 필요가 있다. 그리고 실제 최단경로에 근접시키기 위해서는 노드간의 직선거리를 이용할 필요가 있다.

이를 위해서는 출발노드에서 직접 연결된 노드들 중에서 목적노드까지의 직선거리가 가장 가까운 노드를 선택하고, 선택된 노드로부터 같은 방식으로 목적노드까지 진행하면 임시경로는 빠르게 구축할 수 있다.

최단경로는 구축된 임시경로의 부하량보다 적으면서 최소의 부하량을 가지는 경로가 되는데, 만약 존재하지 않는다면, 임시경로가 최단경로가 된다.

임시경로보다 적은 부하량을 가질 수 있는 경로를 구성할 수 있는 노드를 찾는 것은 네트워크에서 직선거리당 최소 가로 부하량(최소 가로 부하량 원단위)을 이용할 수 있다.

최소 가로 부하량 원단위는 출발노드와 목적노드를 포함하는 네트워크의 각 가로에서 그 가로의 부하량을 직선거리로 나눈 값들 중 최소의 값이 된다.

직선거리와 최소 가로 부하량 원단위를 이용하여 최단경로 구성 가능성이 있는 노드를 찾을 수 있는 예는 다음과 같다.

다른 교통수단을 이용하지 않는 통행시간 부하량으로 최단경로를 탐색할 때, 임시경로의 부하량이 15초였다면, 최단경로는 15초 이내가 되어야 할 것이다.

100m 거리를 최대한 빨리 뛰어 간다고 하여도 9초58(세계기록)이 한계가 되며, 이 기준으로 보면 15초를 달려서 갈 수 있는 최대거리는 156.6m가 되는 것이다. 그 이상의 거리는 15초 이상 달려야 도달 가능하다.

즉, 거리의 부하량과 연결상태에 관계없이 출발지와 도착지에서 직선거리 합이 156.6m 이내에 있는 노드는 임시경로보다 적은 부하량을 가질 가능성이 있는 경로에 포함될 가능성이 있는 노드가 되는 것이다.

임시경로의 부하량은 직선거리로 최대 어느정도로 떨어질 때까지 산출될 수 있는가를 알 수 있기 때문에 임시경로보다 부하량이 적을 수 있는 경로 구축의 가능성 있는 노드들을 찾을 수 있다.

Ⅲ. 탐색영역 축소 알고리즘

1. 네트워크 정비

1) 네트워크 전체집합 구축

탐색대상 네트워크에서 모든 노드를 포함하는 집합을 구축한다.

M = { …p, q … }

2) 직선거리 매트릭스 작성

집합내 노드 간(Link)의 직선거리(Euclidean Distance) 매트릭스(Ecu_Dis[p][q])를 작성한다.

Ecu_Dis[p][q]는 p 노드에서 q 노드까지의 직선거리로 노드좌표를 이용하면 쉽게 계산할 수 있다.

$$Ecu\_Dis[p][q] = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} \quad (1)$$

$p, q \in M$,
p $Node$좌표 : (x_p, y_p),
q $Node$좌표 : (x_q, y_q)

Ecu_Dis[p][q]는 네트워크에서 가로 연결상태와 관계없이 값을 가지며, 같은 노드가 아닐 경우, “0” 이상의 값을 가진다.

직선거리 매트릭스는 정적인 자료(Static Data)로 변화가 없게 된다.

3) 부하량 매트릭스 작성

탐색대상 네트워크의 노드간 부하량(실제거리, 소요시간, 통행비용 등)을 매트릭스(Weight_Dis[p][q])로 작성한다. 이 매트릭스는 노드간 연결가로의 부하량을 나타낸 것으로 직접 연결되지 않은 노드간의 부하량은 “0”로, 직접 연결된 노드간의 부하량은 “0” 이상의 값이 기입된다.

$$Weight\_Dis[p][q] = a \quad (2)$$

$Node\ p,\ q$ 가 직접 연결되어 있지 않으면 $a = 0$,
$Node\ p,\ q$ 가 직접 연결되어 있으면, $Weight = a$

부하량 매트릭스는 실제거리를 부하량으로 할 경우에는 변함이 없는 정적 자료가 되나, 소요시간 등은 수시로 값이 바뀌므로, 최소시간 단위에서 지속적인 변환을 해야 한다.

4) 최소 가로 부하량 원단위 산출

네트워크상에서 최소 가로 부하량 원단위

(Min_Weight_Unit)는 직접 연결된 노드간(Link) 부하량을 그 노드간의 직선거리로 나눈 값들 중에서 최소의 값으로, 나타낸 것으로 다음과 같이 계산한다.

$$Min\_Weight\_Unit = Min.Weight\_Dis[p][q] / Ecu\_Dis[p][q] \quad (3)$$

$$\text{단, } Weight\_Dis[p][q] > 0,\ Ecu\_Dis[p][q] > 0$$

이 원단위는 전체 네트워크에서 직선거리당 최소 부하량을 나타낸 것으로, 탐색영역을 축소하는데 직접 이용된다.

2. 탐색영역 축소

1) 출발노드와 목적노드

출발노드를 sn, 목적노드를 en으로 한다.

2) 임시경로 구축

출발노드(sn)와 목적노드(en)간의 최단경로 탐색에 우선하여 구축되는 임시경로(Temporal path)는 최소 가로 부하량 원단위와 함께 탐색영역의 축소에서 중요한 기능을 하게 된다. 임시경로는 출발노드와 목적노드간 하나의 경로로, 최단경로의 후보가 되며, 빠른 구축이 필요하다.

빠른 구축을 위해서는 A* 알고리즘을 이용할 수 있는데, 이 알고리즘은 휴리스틱 기법으로 경로를 탐색하며, 최적해가 아닌 적정해를 찾는다. 그러나 빠른 탐색은 가능하나, 적정해도 찾지 못할 수 있는 약점을 가진 최단경로 탐색 알고리즘이다.

A* 알고리즘보다 더욱 간단히 연결 노드간 직선거리를 이용하여 구축할 수도 있다(Case 1). 그리고 네트워크가 상당히 불규칙하여 경로구축이 간단하지 않을 것으로 판단되면 노드수를 최소화하는 알고리즘 즉, 노드간의 연결 상태만을 이용하여 구축하는 방법(Case 2)을 생각할 수도 있다[9].

< Case 1 >

출발노드(sn)에서 목적노드(en)까지 서로 직접 연결되는 노드들 중에서 직선거리를 최소화하는 노드들로 경로를 구축한다.

Step 1) 탐색 깊이 i = 0 , tp[i] = sn으로 초기화한다.
Step 2) i 를 1증가시킨다.
Step 3) tp[i-1]에서 직접 연결된 노드들의 f(n)값(f(n) = Ecu_Dis[sn][n]+Ecu_Dis[n][en])을 구하고, 최소 f(n) 값을 가지는 노드를 tp[i]에 삽입한다.
Step 4) tp[i]가 목적노드 en과 다르다면 Step 2로 간다.
Step 5) 임시경로의 부하량(TPRW)을 계산한다.

$$TPRW = \sum_{k=0}^{i-1} (Weight\_Dis[p_k][p_{k+1}]) \quad (4)$$

$$p_k = tp[k], \qquad p_{k+1} = tp[k+1]$$

< Case 2 >

노드간의 연결만을 이용하여 임시경로를 구축하는 것은 아주 간단하고, 탐색시간도 최소화 할 수 있는 방법이다. 출발노드에서 탐색을 시작하는 일방향 탐색으로 임시경로를 구축하거나, 출발노드 및 목적노드에서 동시에 탐색을 실시하는 양방향 탐색으로도 임시경로를 구축할 수 있다.

일방향 탐색 알고리즘은 논리가 간단하고, 변수를 줄일 수 있는 장점이 있다.

Step 1) 출발노드 sn을 S[0][0]와 임시경로 tp[0]에 삽입한다. 탐색깊이 i = 1로 초기화 한다.
Step 2) S[i-1][j]에서 직접 연결되는 노드를 S[i][j]에 저장한다(직접연결되는 노드가 k개라면 j값을 0에서부터 k-1까지, 1씩 증가시킴).
Step 3) S[i][j]가 목적노드 en과 같은 가를 검색하고, 같은 노드가 있으면 검색을 중단한다. k개 모두에서 목적노드가 없으면, i를 1증가시키고 Step 2로 간다.
Step 4) 임시경로 tp[i]에 목적노드 en을 삽입한다.
Step 5) tp[i-1]에는 S[i-1][j]중에서 tp[i]에 직접 연결된 노드를 대입한다.
Step 6) i 를 1감소시킨다.

Step 7) i 가 0이 아니면 Step 5로 간다.
Step 8) 임시경로의 부하량(TPRW)을 계산한다(식(4)를 이용).

네트워크의 구조적인 특성을 고려하여 Case 1이나 Case 2를 선택한다.

3) 직선거리 경계값 산정

직선거리 경계값은 임시경로에서 계산된 부하량(TPRW)을 최소 가로 부하량 원단위로 나누어 산출한다. 이 경계값(Weight_Bounds)은 임시경로의 부하량보다 적은 부하량을 가질 수 있는 경로에 포함될 수 있는 노드를 결정하는 기준 값이 된다.

$$Weight\_Bounds = TPRW / Min\_Weight\_Unit \quad (5)$$

4) 축소된 탐색영역

어떤 노드에서 출발노드까지의 직선거리와 목적노드까지의 직선거리 합이 직선거리 경계값 보다 크다면 그 노드는 임시경로 보다 최단경로를 구축할 가능성이 없게 된다. 즉 어떤 노드에서 출발노드와 목적노드까지의 직선거리 합이 직선거리 경계값 이하인 노드들 만으로 구성된 집합 NM을 구축한다.

NM ={ sn…np,…en }

여기서, NM집합내 노드 np는 다음 식(6)의 조건을 만족해야 한다.

$$(Ecu\_Dis[sn][np] + Ecu\_Dis[np][en]) \leq Weight\_Bounds \quad (6)$$

3. 축소 영역내 최단경로 탐색(최적해) 보장성

개발된 알고리즘은 출발노드와 목적노드간 직선거리가 짧은 임시경로를 구축하고, 구축된 임시경로의 부하량 이하의 경로 구축 가능성이 있는 노드를 찾아내는 것인데, 임시경로보다 부하량이 적은 경로가 없다면 임시경로가 최단경로가 된다.

축소 알고리즘이 네트워크 전체에서 최소 부하량 원단위를 이용하는 것은 최단경로를 구성할 가능성이 있는 노드를 절대로 놓치지 않기 위해서다.

예로, 네트워크에서 각 가로(Link) 부하량을 직선거리로 나눈 값들 중 최소값(최소 부하량 원단위)이 A이고, 임시경로의 부하량이 B이면 그 네트워크에서 B정도의 부하량을 가질 수 있는 최대 직선거리(x, 경계값)는 (5)식과 동일하며, 다음과 같다.

$$1 : A = x : B$$

$$\therefore x(Weight\_Bounds) = B(TPRW)/A(Min\_Weight\_Unit)$$

그리고, 최단경로를 위한 최상의 조건은 모든 노드가 다른 노드를 경유하지 않고 출발노드와 목적노드에 직접 직선으로 연결되고, 최소 부하량원단위와 같은 부하량 원단위를 가지는 것이다. 이러한 경우, 하나의 노드에서 출발노드와 목적노드의 직선거리 합이 x보다 클 경우, 그 노드는 임시경로보다 짧은 경로를 구축할 가능성이 전혀 없는 노드가 된다.

x보다 짧을 경우에는 임시경로보다 더 짧은 최단경로를 구축할 가능성이 있는 노드가 되는 것이다.

따라서 최단경로 탐색영역 축소를 위해 개발된 알고리즘은 최단경로에 포함되는 노드 탐색에 절대로 실패할 수 없기 때문에 알고리즘에서 도출되는 축소된 영역에서의 최단경로(최적해)의 탐색은 보장되는 것이다.

4. 축소 알고리즘의 장단점

최단경로 탐색영역(네트워크)의 축소는 최단경로 탐색소요시간의 단축을 위함이다. 본 연구에서 개발한 알고리즘은 네트워크의 복잡성과 크기, 그리고 출발노드와 목적노드간의 위치관계에 따라 효과정도가 달라진다.

네트워크가 크고, 출발노드와 목적노드가 가까울 경우, 개발된 알고리즘의 효과는 클 것이며, 특히 네트워크가 복잡할 경우, 기존 최단경로 탐색 알고

리즘에 의한 탐색 소요시간이 길어지기 때문에 더욱 큰 효과를 내게 된다.

반대의 경우, 즉 출발노드와 목적노드가 영역의 양측 경계 가까이 있어서 축소영역이 초기 탐색영역과 거의 동일하게 된다면 효과가 미미하게 된다.

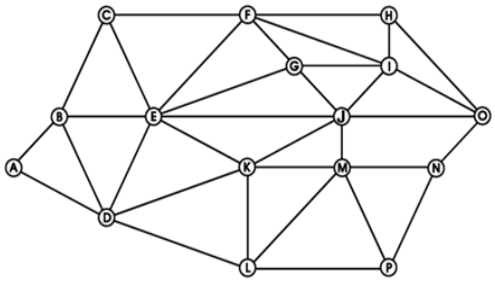
RGS(In-Vehicle Route Guidance Systems)경우는 초기 탐색영역도 넓고, 출발노드가 계속 목적노드로 가까이 진행하기 때문에, 개발된 알고리즘의 효과는 크게 될 것이다.

Ⅳ. 사례 연구

본 연구의 주 목적이 탐색영역 축소 알고리즘 개발에 있으므로, 가상 네트워크를 구축하고 개발된 탐색영역 축소 알고리즘을 네트워크에 적용하여 유용성을 검정하였다.

1. 가상 네트워크

알고리즘 적용을 위한 가상 네트워크는 <그림 1>에서 나타내었다.



〈그림 1〉 가상 네트워크
〈Fig. 1〉 Sample network

16개의 노드와 33개의 가로로 비교적 불규칙한 네트워크를 구축하였다.

2. 자료정비영역 축소 알고리즘 적용

1) 네트워크 전체 집합 구축

16개의 노드를 가지므로, 전체 집합 M은 다음과 같다.

$$M = \{A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P\}$$

2) 직선거리 매트릭스 작성

직선거리 매트릭스는 각 노드의 좌표를 이용하여 식(1)을 이용하여 계산하였으며, 계산결과는 <표 1>에서 나타내었다.

노드의 좌표는 상대좌표를 사용하고, 위치변화에 따른 작선거리의 변화가 없는 정적인 자료(Static data)이므로, 미리 구축해 둘 필요가 있다.

3) 부하량 매트릭스 작성

부하량 매트릭스는 <표 2>에서 작성하였는데, 부하량은 해당 가로의 소요시간 또는 통행비용 등으로 볼 수 있다.

노드간 직접 연결되어 있으면 “0” 이상의 값을 가지나, 직접 연결되어 있지 않으면 “0”이 된다.

4) 최소 부하량 원단위 산출

최소 부하량 원단위(Min_Weight_Unit)는 식(3)으로부터 계산하는데, <표 2>의 각 셀에서의 부하량 값을 <표 1>에 있는 해당 셀의 직선거리로 나눈 값들 중에서 최소값이 된다. 부하량이 “0”인 셀(직접 연결되지 않은 노드 간)은 제외된다.

계산된 값 중에서 최소 부하량 원단위는 “A”노드와 “D”노드를 연결하는 가로에서 1.7857로 산출되었다.

$$Min\_Weight\_Unit$$
$$= Weight\_Dis[A][D] / Ecu\_Dis[A][D]$$
$$= 4.0/2.24 = 1.7857$$

〈표 1〉 직선거리 매트릭스
〈Table 1〉 Euclidean distance matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	0	1.41	3.61	2.24	3.16	5.83	6.32	8.54	8.25	7.07	5.00	5.39	7.00	9.00	10.05	8.25
B	1.41	0	2.24	2.24	2.00	4.47	5.10	7.28	7.07	6.00	4.12	5.00	6.08	8.06	9.00	7.62
C	3.61	2.24	0	4.00	2.24	3.00	4.12	6.00	6.08	5.39	4.24	5.83	5.83	7.62	12.04	7.81
D	2.24	2.24	4.00	0	2.24	5.00	5.00	7.21	6.71	5.39	3.16	3.16	5.10	7.07	8.25	6.08
E	3.16	2.00	2.24	2.24	0	2.83	3.16	5.39	5.10	4.00	2.24	3.61	4.12	6.08	7.00	5.83
F	5.83	4.47	3.00	5.00	2.83	0	1.41	3.00	3.16	2.83	3.00	5.00	3.61	5.00	5.39	5.83
G	6.32	5.10	4.12	5.00	3.16	1.41	0	2.24	2.00	1.41	2.24	4.12	2.24	3.61	4.12	4.47
H	8.54	7.28	6.00	7.21	5.39	3.00	2.24	0	1.00	2.24	4.24	5.83	3.16	3.16	2.83	5.00
I	8.25	7.07	6.08	6.71	5.10	3.16	2.00	1.00	0	1.41	3.61	5.00	2.24	2.24	2.24	4.00
J	7.07	6.00	5.39	5.39	4.00	2.83	1.41	2.24	1.41	0	2.24	3.61	1.00	2.24	3.00	3.16
K	5.00	4.12	4.24	3.16	2.24	3.00	2.24	4.24	3.61	2.24	0	2.00	2.00	4.00	5.10	3.61
L	5.39	5.00	5.83	3.16	3.61	5.00	4.12	5.83	5.00	3.61	2.00	0	2.83	4.47	5.83	3.00
M	7.00	6.08	5.83	5.10	4.12	3.61	2.24	3.16	2.24	1.00	2.00	2.83	0	2.00	3.16	2.24
N	9.00	8.06	7.62	7.07	6.08	5.00	3.61	3.16	2.24	2.24	4.00	4.47	2.00	0	1.41	2.24
O	10.05	9.00	12.04	8.25	7.00	5.39	4.12	2.83	2.24	3.00	5.10	5.83	3.16	1.41	0	3.61
P	8.25	7.62	7.81	6.08	5.83	5.83	4.47	5.00	4.00	3.16	3.61	3.00	2.24	2.24	3.61	0

〈표 2〉 부하량 매트릭스
〈Table 2〉 Link weights matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	0	3.4	0	4.0	0	0	0	0	0	0	0	0	0	0	0	0
B	3.4	0	4.6	4.4	4.2	0	0	0	0	0	0	0	0	0	0	0
C	0	4.6	0	0	5.0	7.8	0	0	0	0	0	0	0	0	0	0
D	4.0	4.4	0	0	4.6	0	0	0	0	0	7.4	7.0	0	0	0	0
E	0	4.2	5.0	4.6	0	8.0	7.6	0	0	8.2	5.6	0	0	0	0	0
F	0	0	7.8	0	8.0	0	3.8	7.4	7.8	0	0	0	0	0	0	0
G	0	0	0	0	7.6	3.8	0	0	6.4	4.2	0	0	0	0	0	0
H	0	0	0	0	0	7.4	0	0	6.0	0	0	0	0	0	8.2	0
I	0	0	0	0	0	7.8	6.4	6.0	0	5.6	0	0	0	0	7.6	0
J	0	0	0	0	8.2	0	4.2	0	5.6	0	7.4	0	4.0	0	7.2	0
K	0	0	0	7.4	5.6	0	0	0	0	7.4	0	6.6	7.2	0	0	0
L	0	0	0	7.0	0	0	0	0	0	0	6.6	0	7.8	0	0	10.0
M	0	0	0	0	0	0	0	0	0	4.0	7.2	7.8	0	6.4	0	5.4
N	0	0	0	0	0	0	0	0	0	0	0	0	6.4	0	5.4	5.0
O	0	0	0	0	0	0	0	8.2	7.6	7.2	0	0	0	5.4	0	0
P	0	0	0	0	0	0	0	0	0	0	0	10.0	5.4	5.0	0	0

3. 영역 축소 알고리즘 적용

1) 출발노드와 목적노드

RGS에서의 적용 등을 고려하여 출발노드와 목적노드를 네트워크의 양측 끝에서 택하지 않고 안쪽에서 선택하였으며, 출발노드(sn)는 “E”, 목적노드(en)는 “I”로 하였다.

2) 임시경로 구축

임시경로 구축은 본 연구에서 Case 1과 Case 2 두 가지로 제안하였는데, 사례연구에서는 출발노드와 목적노드까지의 직선거리를 이용하여 임시경로를 구축하는 Case 1을 선택하였다.

STEP 1. tp[0]=노드 “E”, 그리고 “E”노드와 직접 연결된 노드집합은 {B, C, D, F, G, J, K}.

STEP 2. 직접 연결된 노드집합에서 출발노드에서의 직선거리와 목적노드까지의 직선거리 합이 가장 짧은 노드 선택.

〈표 3〉 임시 경로 노드 선택(1차)
〈Table 3〉 Temporary path node selection(1st)

Node	Euclidean Distance From Starting Node(E)	Euclidean Distance To Target Node(I)	sum	Selected Node
B	2.00	7.07	9.07	
C	2.24	6.08	8.32	
D	2.24	6.71	8.95	
F	2.83	3.16	5.99	
G	3.16	2.00	5.16	√
J	4.00	1.41	5.41	
K	2.24	3.61	5.85	

"G"노드가 가장 짧은 직선거리를 가지므로,

tp[1] = "G"

STEP 3. 선택된 노드"G"가 목적노드가 아니므로, "G"노드에서 직접 연결된 노드들로부터 출발노드 및 목적노드까지 최소 직선거리를 가지는 노드를 선택한다.

〈표 4〉 임시 경로 노드 선택(2차)
〈Table 4〉 Temporary path node selection(2nd)

Node	Euclidean Distance From Starting Node(E)	Euclidean Distance To Target Node(I)	sum	Selected Node
F	2.83	3.16	5.99	
J	4.00	1.41	5.41	
I	5.10	0.00	5.10	√

분석 결과, "I"노드가 선택되었으며, "I"노드는 목적노드이므로, 임시경로의 탐색은 종료된다.

tp[2]= "I"

STEP 4. 임시경로의 부하량(TPRW)을 계산한다.

$$TPRW=$$

$$Weight\_Dis[E][G]+Weight\_Dis[G][I]=$$

$$7.6+6.4=14.0$$

노드수를 최소화 하는 임시경로는 "E"-"G"-"I"가 되었고, 임시경로의 부하량은 14.0으로 산출되었다.

3) 직선거리 경계값 산정

탐색영역 축소를 위한 직선거리 경계값(Weight_Bounds)은 식(5)로부터 7.82로 산출되었다.

$$Weight\_Bounds = TPRW/Min\_Weight\_Unit = 14.0/1.79 = 7.82$$

4) 탐색영역 축소

임시경로 보다 적은 부하량의 최단경로 구성 가능성이 있는 노드들은 그 노드에서 출발노드까지의 직선거리와 목적노드까지의 직선거리 합이 직선거리 경계값(Weight_Bounds)보다 노드가 된다(식(6)).

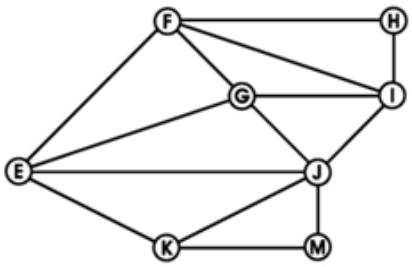
식(6)의 적용결과를 <표 5>에 나타내었다.

〈표 5〉 축소 네트워크 포함 노드 결정
〈Table 5〉 Node selection in reduced network

Node	Euclidian Distance From Starting Node(E)	Euclidian Distance To Target Node(I)	sum	Comparison with Weight_Bounds (7.82)
A	3.16	8.25	11.41	Over
B	2.00	7.07	9.07	Over
C	2.24	6.08	8.32	Over
D	2.24	6.71	8.95	Over
E	0.00	5.10	5.1	Under
F	2.83	3.16	5.99	Under
G	3.16	2.00	5.16	Under
H	5.39	1.00	6.39	Under
I	5.10	0.00	5.10	Under
J	4.00	1.41	5.41	Under
K	2.24	3.61	5.85	Under
L	3.61	5.00	8.61	Over
M	4.12	2.24	6.36	Under
N	6.08	2.24	8.32	Over
O	7.00	2.24	9.24	Over
P	5.83	4.00	9.83	Over

<그림 2>에서와 같이 전체 16개 노드에서 8개의 노드만이 남았으며, 축소된 네트워크인 NM집합은 다음과 같다.

$NM = \{E,\ F,\ G,\ H,\ I,\ J,\ K,\ M\}$



〈그림 2〉 축소된 네트워크
〈Fig. 2〉 Reduced network

4. 알고리즘의 효율성 검증

가상 네트워크에서 출발노드 “E”에서 목적노드 “I”의 최단경로는 구축된 임시경로와는 다르게 “E”-“J”-“I”이며, 최단경로의 부하량은 13.8이다.

노드수와 직선거리를 최소로 하는 임시경로는 실제 부하량을 최소로 하는 최단경로의 가장 가까운 후보가 되나, 최단경로로 단정지울 수는 없다.

사례연구에서 16개의 노드와 33개의 가로를 가지는 탐색영역(네트워크)이 탐색영역 축소 알고리즘을 적용한 결과, 기존 네트워크에서 반 정도 감소한 8개의 노드와 14개의 가로만이 탐색영역으로 되었다.

탐색영역 축소 알고리즘은 더욱 넓은 탐색영역(네트워크)일 수록, 그리고 출발지와 목적지가 가까울수록 효과는 더욱 커진다.

기존 타 알고리즘에서는 없는 임시경로 구축이 추가적인 시간소요를 가져오나, 구축과정에서 정적(Static)인 직선거리를 이용한다는 것과 타 경로와의 비교를 행하지 않고 구축한다는 점에서 큰 소요시간을 필요로 하지는 않는다.

Ⅴ. 결론 및 향후과제

최단경로 탐색 알고리즘에 관한 연구는 가장 기본적인 다익스트라(Dijkstra) 알고리즘이 1959년 발표된 이후부터 현재까지 지속적으로 진행되고 있다.

최단경로 탐색 알고리즘 연구는 대부분 정확도와 함께 빠른 탐색을 하는 알고리즘 개발에 목적을 두고 있다.

빠른 탐색은 효율적인 알고리즘에 의해서만 가능한 것이 아니라, 불필요한 영역을 탐색하지 않는 것도 가능하다.

본 연구에서 개발한 탐색영역 축소 알고리즘은 출발노드에서 목적노드까지 노드 수가 최소인 임시경로를 탐색하고 임시경로의 부하량 보다 적은 부하량을 가질 수 있는 경로에 포함 가능한 노드를 찾는 것이다.

전체 탐색영역(네트워크)에서 최소 가로 부하량 원단위(최소 가로 부하량/직선거리)를 구한 후, 출발노드와 목적노드간 최소의 노드수로 짧은 직선거리를 가지는 임시경로를 구축한다.

네트워크내 하나의 노드에서 출발노드로 부터의 직선거리와 목적노드까지의 직선거리 합이 임시 경로의 부하량을 최소 부하량 원단위로 나눈 값보다 적을 경우, 그 노드는 임시경로 보다 더 적은 부하량을 가질 수 있는 경로에 포함될 수 있는 노드가 된다. 이상의 조건을 만족하는 노드들만을 탐색영역으로 하면, 최단경로 탐색에서 실패 없이 탐색소요시간을 줄일 수 있게 된다.

사례연구에서 규명한 바와 같이 탐색영역 축소 알고리즘은 넓은 탐색영역에서 출발노드와 목적노드가 가까운 경우 더 효과가 있다.

연구가 지속되고 있는 RGS(In-Vehicle Route Guidance Systems)에서는 이동시 계속해서 출발노드가 목적노드에 가깝게 변하므로 개발된 알고리즘이 큰 효과를 가질 것으로 판단한다.

차후, 본 연구에서 적용한 최소 부하량 원단위를 이용하여 최단경로 탐색 알고리즘을 개발할 계획이다.

참고문헌

[1] Lipimg Fu and L. R. Rilett, "Expected shortest paths in dynamic and stochastic traffic networks", *Transportation Research Part B*, vol 32, no 7, pp.499-516, 1998.

[2] Taehyeong Kim, Taehyeong Kim, Bum-Jin Park, Hyoungsoo Kim, "Development of One-to-One Shortest Path Algorithm Based on Link Flow Speeds on Urban Networks ", *The Journal of The Korea Institute of Intelligent Transport Systems* , vol 11, no 5, pp.38-45, 2012.

[3] E.W. DIJKSTRA, "A Note on Two Problems in Connexion with Graphs", *Numerische Mathematik1*, pp.269-271, 1959.

[4] Orda, A. and Rom, R. , "Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length", *journal of the ACM 37*, pp.607-625, 1990.

[5] Kaufman, E. , Lee, J. and Smith, R. L. , "Fastest paths in time-dependent networks for intelligent vehicle highway systems application", *IVHS journal 1(1),* pp.1-11, 1993.

[6] Ziliaskopoulos, A.K. and Mahassani, H.S. ,"Time-dependent, shortest-path algorithms for real-time intelligent vehicle system applications", *Transportation Research Record 1408, TRB, National Research Council, Washington DC,* pp.94-100, 1993.

[7] Chabini, I. ,"A new algorithm for shortest-path in discrete dynamic networks", *Proceeding of the 8th International Federation of Automatic Control Symposium on transportation Systems*, Vol. 2, eds M. Papageorgiu and A. Pouliezos, pp.551-556, 1997.

[8] Hall, R. ,"The fastest path through a network with random time-dependent travel time", *Transportation Science 20(3)*, pp182-188, 1986.

[9] Ryu, Yeong-Geun, "A Study on the Forecasting of Bus Travel Demand and the Method of Determining Optimal Bus Network", Ph. D. Thesis, Yeong Nam University, pp.131-140, 1996.

저자소개

유 영 근 (Ryu, Yeong-Geun)

2011년 2월~현재: 영남교통정책연구원 원장
2001년 3월~2011년 2월: 영남대학교 겸임교수
1997년 2월: 영남대학교 도시공학과 박사
1987년 2월: 영남대학교 도시공학과 석사
1985년 2월: 영남대학교 도시공학과 학사
e-mail : ygryu@chol.com