한국ITS학회논문지
제12권, 제6호 (2013년 12월)
pp.01~09
http://dx.doi.org/10.12815/kits.2013.12.6.001

A* 알고리즘의 최단경로 탐색 정확도 향상을 위한 역방향 적용방법에 관한 연구

A Study on A* Algorithm Applying Reversed Direction Method for High Accuracy of the Shortest Path Searching

유 영 근* 박 용 진**
(Ryu, Yeong-Geun) (Park, Yongjin)

요 약

Dijkstar 알고리즘에 기초하는 최단경로 탐색 알고리즘의 탐색속도 향상에 관한 많은 연구들이 지속되어 왔다. 그 대표적인 알고리즘이 A* 알고리즘이다. 빠른 탐색속도는 A* 알고리즘의 장점이지만, 복잡하고 불규칙한 가로 네트워크에서 실제의 최단경로 탐색이 실패할 확률이 높다. 탐색실패란 목적노드를 탐색하지 못한 경우와 최단경로가 아닌 경로를 구축하는 것을 의미한다. 본 연구는 A* 알고리즘의 최단경로 탐색 성공확률을 높이기 위한 방법으로 일차적으로 출발노드와 목적노드 간 연결 관계를 정리하고, 목적노드에서 출발노드까지 정리된 경로에 따라 A* 알고리즘을 역으로 적용한 것이다. 이 방법은 네트워크 및 경로 부하량 특성에 따라 실제의 최단경로가 아닌 경로를 최단경로로 구축하는 경우가 발생할 수는 있으나, 경로구축의 완전한 실패는 발생시키지 않는다. 이 방법을 실제 복잡한 네트워크에 적용하여 유효성을 검증한 결과, 통상적인 A* 알고리즘의 적용보다 탐색 소요시간은 약간 증가하나, 정확성은 상당히 높아지는 것으로 분석되었다.

핵심어 : 최단경로탐색 알고리즘, 경로안내시스템, 탐색영역, A* 알고리즘, 다익스트라 알고리즘

Abstract

The studies on the shortest path algorithms based on Dijkstra algorithm has been done continuously to decrease the time for searching. A* algorithm is the most represented one. Although fast searching speed is the major point of A* algorithm, there are high rates of failing in search of the shortest path, because of complex and irregular networks. The failure of the search means that it either did not find the target node, or found the shortest path, witch is not true.

This study proposed A* algorithm applying method that can reduce searching failure rates, preferentially organizing the relations between the starting node and the targeting node, and appling it in reverse according to the organized path.

This proposed method may not build exactly the shortest path, but the entire failure in search of th path would not occur. Following the developed algorithm tested in a real complex networks, it revealed that this algorithm increases the amount of time than the usual A* algorithm, but the accuracy rates of the shortest paths built is very high.

Keywords : Shortest path algorithm, RGV, Searching area, A* algorithm, Dijkstra algorithm

* 주저자 : 영남교통정책연구원 원장
** 교신저자 : 계명대학교 교통공학과 교수
† 논문접수일 : 2013년 11월 14일
† 논문심사일 : 2013년 12월 10
† 게재확정일 : 2013년 12월 23일

Ⅰ. 서 론

최단경로 탐색 알고리즘은 다양한 분야에서 긴요한 것으로, 효율적으로 최단경로를 구축하기 위한 연구가 지속적으로 진행되고 있다.

하나의 출발지에서 하나의 목적지를 탐색(One to One Search)하는 최단경로 탐색 알고리즘은 Dijkstra 알고리즘[1]에 기초를 두고 있으며, 현재까지도 많이 활용하고 있다.

Dijkstra 알고리즘은 최단경로를 정확히 찾아내는 장점을 가지고 있으나, 탐색 소요시간이 네트워크의 혼잡정도 및 출발지와 목적지의 거리에 비례하여 길어지는 단점을 가진다.

Dijkstra 알고리즘의 단점 해소, 즉 보다 빠른 탐색은 탐색 노드를 줄이는 방향에서 연구되고 있는데, 3가지 방법으로 분류가능하다[2,3].

주요 3가지 방법은 출발노드와 목적노드에서 양방향으로 최단경로를 탐색하는 양방향 탐색법(Bidirectional Search), 소요시간 단축을 위하여 정확한 최단경로의 구축이 아니라 적정 수준에서의 최단경로 구축을 목적으로 하는 휴리스틱 탐색법(Heuristic Search), 그리고 출발노드와 목적노드가 멀리 떨어진 경우 주로 사용하는 네트워크 계층구분법(Hierarchical Methods) 등 이다.

각 최단경로 탐색 방법들은 장점과 단점을 모두 가지고 있기 때문에 네트워크의 상태와 링크 부하량의 수준 및 변화 정도 등에 따라 선택하여 적용한다.

최단경로 탐색에서 소요시간이 최소가 될 수 있는 탐색은 휴리스틱 탐색이며, AI분야에서 개발되고 활성화된 A* 알고리즘이 그 대표적인 알고리즘이다.

빠른 탐색이 A* 알고리즘의 장점이나, 불규칙하고 복잡한 네트워크에서는 실패할 확률이 높아지는 단점을 가진다[4]. 본 연구에서 탐색실패의 의미는 목적노드를 탐색하지 못한 경우와 최단경로가 아닌 경로를 최단경로로 구축한 경우를 의미한다.

A* 알고리즘의 정확도를 높이기 위한 연구들은 수행되었고, 정확도도 높였다[4, 5]. 그러나 정확도가 높을수록 탐색 속도가 감소하는 결과를 초래하므로, 최단경로 정확도의 향상과 함께 탐색 소요시간을 최소화할 수 있는 알고리즘의 연구가 필요하다.

본 연구에서는 A*알고리즘 적용방법을 개선하여 탐색 실패확률을 감소시키고, 탐색 소요시간을 최소화 하는 방법 제시하는 것을 목적으로 한다.

Ⅱ. 기존 연구

1. A* 알고리즘

대표적인 휴리스틱 최단경로 탐색 알고리즘이 A* 알고리즘이다. A* 알고리즘은 깊이 우선 탐색(Depth First Search)으로 휴리스틱 추정치(Estimate)를 사용하는 평가함수(Guidance function)에 의해 산정된 값 중 최소 값을 가진 노드를 선택 한다[6].

탐색노드 n에서의 평가함수는 식 (1)과 같다.

$$f(n) = g(n) + h(n) \quad (1)$$

$g(n)$: 출발노드에서 n노드 까지의 부하량

$h(n)$: n노드에서 '목적노드까지의 '추정 부하량

$h(n)$은 주로 직선거리(Manhattan distance, Euclidean distance)를 이용한다.

출발노드에서부터 연결된 노드 중 최소 평가함수 값을 가지는 노드를 최단경로 구성 노드로 하면서 진행하는데, 목적노드가 선택되면 종료된다.

A* 알고리즘은 타 경로와의 비교없이 목적노드 방향의 노드를 찾아감에 따라 탐색 소요시간이 짧게 되는 장점을 가진다. 그러나 규칙적이지 않고 복잡한 네트워크에서는 최단경로 탐색에 실패할 확률이 높다[4].

즉, 휴리스틱의 성격에 따라 최단이 아닌 경로를 최단경로로 구축하거나, 탐색되지 않았던 노드로 다시 탐색을 반복해야 하기 때문에 탐색 소요시간이 크게 증가되는 단점을 가진다.

2. A* 알고리즘의 기존 개선 연구

1) A* 알고리즘의 양방향 탐색

양방향 탐색법은 출발노드와 목적노드 양측에서 A* 알고리즘으로 최단경로 탐색을 같이 행하는 것

이다. 기본적인 탐색의 성공조건은 출발노드에서의 탐색과 도착노드의 탐색이 중간에서 하나의 같은 노드를 탐색해야 한다는 것이다.

기본적인 조건 외에도 양방향 탐색방법이 성공하기 위해서는 하나의 탐색방향에서 반대방향 탐색으로 바꾸는 조건이다. 통상 양측의 반복 탐색을 편리한 것으로 판단하나, 단일방향 탐색에 비하여 효율적으로 보기 어렵다는 연구결과[5]가 있다.

그리고 양측 방향에서의 탐색을 중단하기 위한 최적조건의 설정이 과제가 되며, 이 과제의 해결과정에 따라 단일 방향 탐색보다 비효율적으로 된다는 것이 동일한 연구[5]에서 밝혀졌다.

2) 탐색영역의 제한

네트워크에서 탐색영역의 제한은 모든 노드를 탐색해야 하는 Dijkstra 알고리즘의 단점을 직접적으로 극복하기 위한 것으로, 검색의 필요성이 없는 노드를 사전에 차단하는 것이다.

Fu(2006)는 A* 알고리즘의 탐색과정 중에서 평가함수 값이 추정된 한계 값을 초과할 경우, 탐색하지 않는 방법을 식 (2), 식 (3)과 같이 제안하였다.

$$g(n) + h(n) \le E_{(o,d)} \quad (2)$$

$E_{(o,d)}$: 출발노드에서 목적노드까지의 최소 소요시간(부하량) 추정 한계값

$$E_{(o,d)} = K \cdot h(n) \quad (3)$$

K : 조정계수

식 (3)에서 조정계수 K는 네트워크 사전 조사로부터 산정하거나, 목적노드 탐색 실패시 K값을 증가시키면서 조정해 가는 것이다.

이 방법에서는 k값의 산정이 쉽지 않으며, 탐색 소요시간을 증대시킬 수도 있는 문제가 있다.

유영근(2013)은 전처리에서 탐색영역을 축소하는 방법을 제안하였다[7]. 출발노드에서 목적노드까지의 직접연결 노드수를 최소로하는 임시경로를 구축하고 임시경로의 부하량(거리, 소요시간, 비용 등)과 네트워크 전체에서 구축된 최소 가로 부하량 원단위를 이용하여 식 (4)와 같이 직선거리 경계 값을 산정한다.

$$Weight\_Bounds = TPRW / Min\_Weight\_Unit \quad (4)$$

$Weight\_Bounds$: 직선거리 경계값

$TPRW$: 임시경로의 부하량

Min_Weight_Unit : 최소 가로부하량 원단위

식 (5)와 같이 네트워크 전체 노드에서 출발노드까지의 직선거리와 목적노드까지의 직선거리 합이 직선거리 경계값 보다 적은 노드들 만으로 탐색 네트워크를 축소한다.

$$Ecu\_Dis[sn][p] + Ecu\_Dis[p][en] \le Weight\_Bounds \quad (5)$$

$Ecu\_Dis[sn][p]$: 출발노드에서 노드까지의 직선거리

$Ecu\_Dis[p][en]$: 노드에서 도착노드까지의 직선거리

$Weight\_Bounds$: 최소 가로 부하량 원단위

이 방법은 광역 네트워크에서는 효과를 얻을 수 있으나, 광역이 아닌 작은 규모의 네트워크에서는 소수의 노드만이 최단경로 탐색에서 제외될 수 있어 탐색 소요시간의 단축이 어려울 수도 있다.

3) 평가함수의 개선

일반적인 A*알고리즘에서는 탐색노드를 결정하는 평가함수에서의 추정 부하량($h(n)$)을 직선거리로 사용한다. Fu(1996)는 효율적인 탐색을 위하여 식 (6)과 같이 목적지까지의 직선거리를 네트워크에서의 평균속도로 나누는 것으로 제안하였다.

$$h_{(n)} = S_{(n)} / V \quad (6)$$

$S_{(n)}$: n탐색노드에서 목적노드까지의 직선거리

V : 네트워크에서의~평균속도

그러나, 이 방법은 평균속도 적용에 따라 탐색속도는 빨라질 수 있으나, 정확도는 크게 높일 수 없는 단점이 있다.

유영근(2013)은 추정 부하량($h(n)$)을 직선거리에 최소 가로부하량 원단위를 곱하여 산출하는 최소기대 부하량으로 바꾸어 적용하는 방법을 개발하였다[4].

$$emw(n) = Ecu\_Dis[n][en] \times Min\_Weight\_Unit \quad (7)$$

$emw(n)$: 탐색노드(n)에서~목적노드(en)까지

최소 기대 부하량
$Ecu\_Dis[n][en]$: 탐색노드(n)에서~목적노드(en)까지 직선거리
Min_Weight_Unit : 최소 가로부하량 원단위

이 방법은 최소가로 원단위를 적용하여 탐색함에 따라 최단경로 탐색 소요시간은 증가하나, 정확한 최단경로를 탐색할 수 있다.

Ⅲ. A* 알고리즘의 정확도 향상 방법

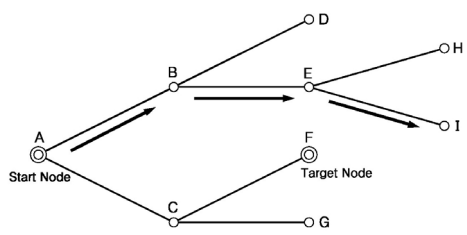
1. 개발방향

개발하는 방법은 A* 알고리즘의 장점인 짧은 시간의 탐색을 크게 후퇴시키지 않으면서 최단경로 탐색 실패확률을 줄이는 방법이다.

Dijkstra 알고리즘이 가장 안정적으로 최단경로를 구축하나, 구축과정에서 경로의 부하량 비교에 따라 소요시간이 길어지는 단점을 가진다.

그리고 A* 알고리즘은 깊이 우선탐색으로 타 경로와의 비교 없이 탐색함에 따라 실제 최단경로가 아닌 경로를 최단경로로 구축하는 경우나, 목적노드를 찾지 못하는 경우 등, 경로구축의 실패 확률이 높은 단점을 가지고 있다.

A* 알고리즘의 실패 사례는 <그림 1>에서 나타내었는데, 출발노드 A 에서 목적노드 F 까지의 최단경로 탐색시, 우선적으로 A 노드와 직접 연결된 B 노드와 C 노드를 탐색하게 된다. 만약 B 노드의 평가함수($f(B)$)가 C 노드의 평가함수($f(C)$)보다 낮은 값이 산출 될 경우, 최단경로에 B 노드를 포함하고, B 노드에서 D 노드와 E 노드의 평가함수를 비교하여 진행하게 된다.



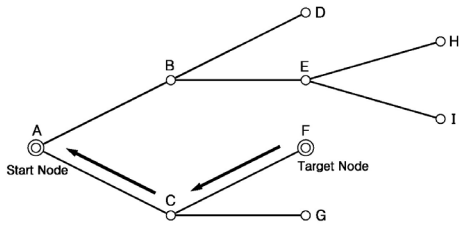
〈그림 1〉 A* 알고리즘의 실패적용 예
〈Fig. 1〉 Example of failed A* algorithm's application

A* 알고리즘은 후퇴의 기능이 없으므로 이와같이 규칙적이지 않은 네트워크에서는 실패를 하기 쉬운 것이다.

실패를 피하기 위해서 출발노드와 목적노드의 링크연결 상태를 우선적으로 파악할 수 있다면 효과적인 탐색이 가능하게 된다.

Dijkstra 알고리즘과 같이 모든 노드를 검색하여 최단경로를 구축하는 것이 아니라, 출발 노드(부(夫))에서 연결노드(자(子))를 찾고, 자(子)의 노드를 부(夫)의 노드로 바꾼 후, 다시 자(子)의 노드를 찾는다. 이 과정을 목적노드가 자(子)의 노드에 있을때까지 반복하는 것이다.

즉, 노드간의 연결 상태만을 정리하는 것이다. 정리된 연결상태에서 역으로(목적노드에서 출발노드) 최단경로를 탐색하면 탐색영역이 축소되는 효과와 함께 최단경로 구축의 실패확률도 줄일 수 있게 된다. 이 경우, 적정범위에서 최단경로를 구축하는 휴리스틱적 탐색결과는 실패없이 얻을 수 있다. A* 알고리즘의 역방향 적용 예는 <그림 2>에서 나타내었다. 목적노드 F와 연결된 노드는 C 노드이며, C 노드에서 연결된 노드는 출발노드인 A 노드이기 때문에 간단히 구축된다. 출발노드와 목적노드 사이 복수개의 경로가 존재하여도 평가함수에 의해 선정되는 노드들로 하나의 최단경로가 구축된다.



〈그림 2〉 역방향 적용 A* 알고리즘의 예
〈Fig. 2〉 Example of reverse direction application of A* algorithm

2. A* 알고리즘의 역방향 적용 방법

우선적으로 출발노드에서 부터 목적노드까지 진행단계별 연결노드를 찾아 저장한다. 이를 목적노드

가 있을때 까지 반복하는데, 목적노드가 있으면, 목적노드를 출발노드로 하고, 출발노드를 목적노드로 하여 A* 알고리즘을 적용한다.

노드간 연결관계는 연결된 노드를 이차원 배열로 하고 일차배열에 진행단계(i), 이차배열에 해당노드가 인식되도록 정리한다[8].

적용하는 A* 알고리즘의 평가함수($f(n) = g(n) + h(n)$)에서 $g(n)$은 출발노드에서 탐색노드(n 노드)까지의 부하량이 아니라 탐색노드(n노드)에서 목적노드까지의 부하량으로 바꾸어 적용한다.

그리고 통상 탐색노드에서 목적지까지의 직선거리를 적용하는 추정 부하량 $h(n)$은 탐색노드(n 노드)에서 출발노드까지의 추정부하량(직선거리 등)으로 바꾸어 적용한다.

역방향 적용 A* 알고리즘을 단계별로 정리하면 다음과 같다.

```
< 적용변수 >
i                  /*  진행단계                          */
sn                 /*  출발노드                          */
en                 /*  목적노드                          */
con_p[i][k]        /*  진행단계 i에서 전 단계(i-1단계)  */
                   /* 에서 탐색된 직접 연결된노드       */
shortest_path[i]   /* 최단경로의 i번째 노드             */
closed{  }         /* 검색된 노드집합                   */

rdis[k][l]         /* k노드에서 l노드까지의 부하량      */
                   /* rdis[k][l] ≠ 0 : k, l 직접연결됨  */
                   /* rdis[k][l] = 0 : k, l 직접연결안됨 */
sdis[k][l]         /* k노드에서 l노드까지의 직선거리    */
                   /* rdis[k][l] ≠ 0 : k, l 다른 노드   */
                   /* rdis[k][l] = 0 : k, l 같은 노드   */

< STEP 0 >
i ⇐ 0
con_p[0][0] ⇐ sn

< STEP 1 >
진행단계 i를 1증가(i ⇐ i+1) 시키고, 탐색노드 고유번호 k를 초기화(k ⇐ 0).

< STEP 2 >
탐색된 노드 집합 closed{ }에 포함되어 있지 않으면서
연결노드 con_p[i-1][k]에 직접 연결된 노드(l)를 탐색.
연결노드를 con_p[i][k]와 closed{ }에 저장.
연결노드가 없으면 STEP 4로 이동.

< STEP 3 >
k ⇐ k+1, STEP 2로 이동.

< STEP 4 >
연결노드 con_p[i][l], (l = 0~k)중에서 목적노드 en과 같은 노드가 없으면 STEP 1로 이동.

< STEP 5 >
현 진행단계 최단경로에 목적노드 저장(shortest_path[i] ⇐ en).

< STEP 6 >
최단경로 구성노드(shortest_path[i])와 전 단계에서 직접 연결된 노드들 중, 최소 평가함수를 가진 노드를 선택(rdis[l][shortest_path[i]] ≠ 0, min. rdis[l][shortest_path[i]] + sdis[shortest_path[i]][en])하고 최단경로 및 검색된 노드집합에 포함.shortest_path[i-1] ⇐ l, closed { } ⇐ l).

< STEP 7 >
i 를 1감소(i ⇐ i-1)시킴. i가 0이 아니면 STEP 6으로 이동.

< STEP 8 >
최당경로(shortest_path[n], n = 0~i) 출력 후, 종료.
```

Ⅵ. 효율성 검증

1. 효율성 검증방법

개발한 방법의 효율성 검증은 최단경로 탐색의 정확성에 초점을 두는데, 정확성의 향상이 너무 많은 탐색 시간을 요하게 되면 이 방법의 적용 효과가 반감될 수 있으므로 탐색시간도 검증항목으로 하였다.

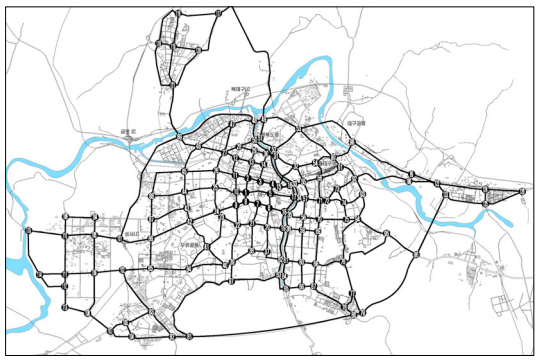
효율성 검증은 정확성과 소요시간, 두 개의 항목에 대해 기존 알고리즘과의 탐색결과를 비교하는 것으로 하였다. 비교대상 알고리즘은 A* 알고리즘과 최근 100% 정확성을 가지는 것으로 개발된 최소 기대 부하량을 이용한 알고리즘[4]으로 하였다.

본 연구에서 개발한 방법의 효율성은 기존 A* 알고리즘보다 정확도가 높으면서, 정확도를 목표로 한 최소 기대 부하량을 이용한 알고리즘보다 빠른 탐

색이 가능하다면 효율성을 가지는 것으로 판단할 수 있다.

대상 네트워크는 결과비교를 보다 객관적으로 하기 위하여 최소 기대 부하량을 이용한 알고리즘 개발시 적용한 네트워크와 부하량을 그대로 하였다.

네트워크: 대구시 간선가로 네트워크
126개 노드, 213개 링크
부하량: 가로 통행시간
2013년 6월 13일 18시 15분 통행시간



〈그림 3〉 대구시 네트워크
〈Fig. 3〉 Daegu-city Network

2. 효율성 검증

효율성 검증을 위한 최단경로 탐색 수는 전체 노드간의 최단경로 15,750(126 × 126 −126)개로 하였다. 그리고 출발노드와 목적노드까지 2㎞의 직선거리 단위로 구분하여 비교하였다.

1) 최단경로의 정확성

효율성 평가에서 정확성은 실제 최단경로를 정확히 구축한 경우를 성공으로 하고, 완벽한 최단경로가 아니거나, 아예 경로를 구축하지 못한 경우 모두를 실패로 보았다.

3개 알고리즘에 대한 정확성의 비교결과는 <표 1>에서 나타내었다.

〈표 1〉 최단경로 탐색 정확성 비율 결과
〈Table 1〉 Success rate of Minimum path searching

Range of Straight Distance(km,X) (Number of Routes)	Algorithm		
	A* (%)	M.E.W[1] (%)	Rev. A*[2] (%)
X < 2 (1,430)	59.23	100.00	77.83
2 ≤ X < 4 (3,198)	27.33	100.00	44.28
4 ≤ X < 6 (3,214)	16.46	100.00	30.06
6 ≤ X < 8 (2,570)	9.61	100.00	27.04
8 ≤ X < 10 (2,124)	5.79	100.00	22.46
10≤ X < 12 (1,626)	1.48	100.00	19.74
12≤ X < 14 (898)	1.22	100.00	18.37
14≤ X< 16 (412)	0.97	100.00	22.33
16 ≤ X (278)	0.36	100.00	31.65
Average	16.88	100.00	33.86

1) Minimum expected weights algorithm
2) Reverse direction application A* algorithm

대상 네트워크가 규칙적이지 않으면서 복잡하고, 부하량인 소요시간도 가로 길이에 비례적이지 못한 것 등의 요인으로 인해 A* 알고리즘의 완벽한 성공확률은 출발노드와 목적노드간의 길이가 멀면 멀수록 저하하는 것으로 나타났다.

최소 기대 부하량을 이용한 알고리즘은 최소 가로 부하량 원단위를 이용하고 반복탐색의 기능을 가짐에 따라 완벽하게 최단경로를 구축한다.

본 연구에서 개발한 A* 알고리즘의 역방향 적용방법은 A* 알고리즘의 정확성보다 월등히 높은 정확도를 가지는 것으로 분석되었다.

2) 최단경로 탐색 소요시간

탐색 소요시간은 출발노드에서 최단경로 탐색을 시작하여 목적노드를 찾아 종료할 때까지로 하였으며, 소요시간의 측정은 CPU 3.09GHz, RAM 2.99GB 성능의 PC에서 하였다. 측정결과는 <표 2>에서 나타내었다.

<표 2>에서 나타낸 평균 탐색 소요시간은 최단경로 탐색에 성공한 경우의 소요시간만을 집계하여 평균한 값이다.

노드간 직선거리가 길어질수록 탐색소요시간은 길어지는 경향을 나타내는데, A* 알고리즘과 본 연구에서 개발한 방법의 연구결과에서는 직선거리와 소요시간이 비례하지 않은 결과도 나타났다. 이는 소수의 성공 경로를 가진 경우, 그리고 대상 네트워크가 그리드 형태의 규칙적 가로가 아니고 복잡하며, 거리는 가깝지 않으나 부하량이 적은 순환선 등의 영향이 크기 때문이다. 그리고, Link와 부하량(소요시간)이 상당히 비례적이라고 볼 수 없는 것 등에 의한 영향도 있다.

15,750개의 최단경로를 찾는 전체 소요시간을 보면 A* 알고리즘이 2.00E-04 ms, 최소 기대 부하량을 이용한 알고리즘은 6.00E-06 ms로 측정된 반면 본 연구에서 개발한 A* 알고리즘의 역방향 적용 방법은 3.29E-03 ms로 측정되었다.

하나의 경로탐색에 대한 평균 소요시간은 A* 알고리즘이 1.09E-08 ms, 최소 기대 부하량을 이용한 알고리즘이 3.64E-06 ms로 측정되었고, 본 연구에서 개발한 방법은 2.09E-07 ms로 측정되었다.

본 연구에서 개발한 A* 알고리즘의 역방향 적용 방법은 A* 알고리즘에 비하여 약간의 탐색 소요시간이 증가하나, 높은 정확성을 가지는 것으로 확인되었으므로 효율성을 가지는 것으로 판단된다.

Ⅴ. 결 론

최단경로 탐색 알고리즘은 다방면에서 다양하게 이용되고 있는 알고리즘이다. 교통분야에서는 차내 RGS(Route Guidance Systems)의 발달과 연계되어 더욱 빠르고 정확한 목적지 탐색을 위한 개선 알고리즘 개발이 지속적으로 진행되고 있다.

Dijkstra 알고리즘에서 출발하는 최단경로 탐색 알고리즘은 Dijkstra 알고리즘이 정확한 최단경로 구축을 보장하고 있음에 따라 속도 향상 측면에서 연구가 많이 되었으며, 그 대표적인 결과가 A* 알고리즘이다. 인공지능연구 분야에서 개발된 A* 알고리즘은 휴리스틱 탐색으로 적정 범위 내에서 빠른 탐색을 수행한다.

그러나 A* 알고리즘은 빠른 탐색만을 주목적으로 설정한 결과, 복잡한 네트워크에서는 최단경로가 아닌 경로를 최단경로로 결정하거나 최단경로를 전혀 구축하지 못하게 되는 경우가 많다. 그리고 이와 같은 탐색 실패를 억제하기 위한 반복탐색에서는 소요시간이 크게 증가하는 단점이 있다.

본 연구에서는 이와 같은 A* 알고리즘의 낮은 정확도를 높이기 위하여 A* 알고리즘을 역방향으로 적용하는 방법을 제시하였다.

출발노드에서 목적노드까지 노드간 연계를 2차원 배열로 설정해 두고, 목적노드에서 역방향으로 출발노드까지 최단경로를 탐색하는 알고리즘이다.

목적노드에서 출발노드까지 직접 연결되는 노드들만을 탐색 대상으로 하므로, 탐색영역의 축소효과

〈표 2〉 최단경로 탐색 소요시간 측정 결과
〈Table 2〉 Results of Minimum path running time

Range of Straight Distance(km,X) (Number of Routes)	Algorithm		
	A* (Aver. ms) (Number of Success)	M.E.W[1] (Aver. ms) (Number of Success)	Rev. A*[2] (Aver. ms) (Number of Success)
X < 2 (1,430)	1.05E-08 (847)	3.5E-08 (1,430)	4.41E-08 (1,113)
2 ≤ X < 4 (3,198)	4.69E-09 (874)	9.69E-08 (3,198)	7.32E-08 (1,416)
4 ≤ X < 6 (3,214)	9.96E-09 (529)	2.89E-07 (3,214)	1.16E-07 (966)
6 ≤ X < 8 (2,570)	1.79E-08 (247)	1.13E-06 (2,570)	1.70E-07 (695)
8 ≤ X < 10 (2,124)	1.51E-08 (123)	2.97E-06 (2,124)	2.81E-07 (477)
10≤ X < 12 (1,626)	9.84E-09 (24)	6.97E-06 (1,626)	4.69E-07 (321)
12≤ X < 14 (898)	1.0E-30 (11)	1.37E-05 (898)	5.39E-07 (165)
14≤ X< 16 (412)	1.0E-35 (4)	2.42E-05 (412)	5.34E-07 (92)
16 ≤ X (278)	5.76E-08 (1)	4.76E-05 (278)	4.46E-07 (88)
Total	2.00E-04	6.00E-02	3.29E-03
Average	1.09E-08	3.64E-06	2.09E-07

1) Minimum expected weights algorithm
2) Reverse direction application A* algorithm

가 있고, 경로구축을 완전히 실패하는 경우는 절대로 발생하지 않으며, 적정범위에서의 최단경로를 구축하게 된다.

규칙적이지 못하고 복잡한 대구 간선가로 네트워크를 이용하여 효율성을 검증한 결과, 통상의 A* 알고리즘 보다 월등히 높은 정확도를 가지는 것으로 분석되었다.

최단경로 탐색소요시간은 통상의 A* 알고리즘 보다 약간 증가하였으나, 100% 정확도를 위해 개발된 알고리즘에 비해서는 상당히 빠른 탐색소요시간을 가지는 것으로 분석되었다.

결과적으로 높은 정확도를 비교적 빠른 탐색속도로 탐색하기 위해서는 A* 알고리즘을 그대로 적용하기 보다 본 연구에서 개발한 A* 알고리즘의 역방향 적용방법을 이용하는 것이 효과적일 것으로 판단한다.

개발한 방법 또한 통상의 A* 알고리즘의 적용결과와 같이 복수개의 경로 중에서 부하량이 높아도 노드수가 적은 경로를 최단경로로 선택할 확률은 가지고 있다. 차후, 이와 같은 약점을 극복할 수 있는 연구가 계속되어야 할 것이다.

참고문헌

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

[2] L. Fu and D. Sun and L. R. Rilett, “Heuristic shortest path algorithms for transportation applications: State of the art”, *Computers & Operations vol. 33,* no. 11, pp.3324-3343, 2006.

[3] W. Dorothea and W. Thomas, “Speed-Up Techniques for Shortest-Path Computations”, *STACS 2007, Lecture Notes in Computer Science*, vol. 4393, pp.22-36, 2007.

[4] Y. G. Ryu, “Development of a shortest path searching algorithm using minimum expected weights”, *The journal of The Korea Institute of Intelligent Transport Systems,* vol. 12, no 5, pp.36-45, 2013.

[5] S. E. Dreyfus, “An appraisal of some shortest path algorithms.”, *Operations Research,* vol. 17, no. 3, pp.395-412, 1969.

[6] P. E. Hart and N. J. Nilsson and B. Raphael, ″A Formal Basis for the Heuristic Determination of Minimum Cost Paths″, *IEEE Transactions on Systems Science and Cybernetics,* vol. 4, no. 2, pp.100 – 107, 1968.

[7] Y. G. Ryu, “Development of shortest path searching network reduction algorithm”, *The journal of The Korea Institute of Intelligent Transport Systems,* vol. 12, no. 2, pp.12-21, 2013.

[8] Y. G. Ryu, “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
연락처 : 010-7109-6986

박 용 진 (Park, Yongjin)
1992년 9월 ~ 현재: 계명대학교 교통공학과 교수
2011년 3월 ~ 2013년 2월: 대한교통학회 대구경북지회 지회장
1992년 6월 : University of Cincinnati 토목환경공학과 교통전공 박사
1988년 8월 : University of Cincinnati 토목환경공학과 교통전공 석사
1983년 8월 : 중앙대학교 토목공학과 학사
e-mail : ypark@kmu.ac.kr
연락처 : 053-580-5587