최적 경로를 보장하는 효율적인 양방향 탐색 알고리즘

황보 택근†

요 약

도로에서의 최적 경로 탐색은 출발지와 목적지의 위치를 알고 있는 경우로서 탐색에 대한 일종의 사전 지식을 가진 탐색으로 A* 알고리즘이 널리 사용되고 있다. 단방향 A* 알고리즘은 최적의 경로를 보장해 주는 반면 탐색 시간이 많이 소요되고 양방향 A* 알고리즘은 최적 경로를 보장해 주지 못하거나 최적 경로 보장을 위해서는 오히려 단방향 A* 보다 탐색 시간이 더 많이 소요될 수도 있다. 본 논문에서는 탐색 시간이 우수하며 최적 경로를 보장하는 새로운 양방향 A* 알고리즘을 제안한다. 본 논문에서 제안하는 알고리즘의 효용성을 확인하기 위하여 실제 도로에 적용한 결과 정확한 최적 경로를 탐색하고 탐색 시간도 매우 우수한 것으로 확인되었다.

Efficient Bidirectional Search Algorithm for Optimal Route

Taegkeun Whangbo†

ABSTRACT

A* algorithm is widely used in optimal car route search which is a kind of informed search, since the locations of starting and ending points are known a priori. Unidirectional A* algorithm requires considerable search time but guarantees a optimal path, bidirectional A* algorithm does not guarantee a optimal path and takes even longer search time than unidirectional search to guarantee a optimal path. In this paper, a new bidirectional A* algorithm which requires less search time and guarantees a optimal path is proposed. To evaluate the efficiency of the proposed algorithm, several experiments are conducted in real road map and the results show that the algorithm is very effective in terms of finding a optimal path and search time.

Key words: 탐색 알고리즘, 최적 경로, A* 알고리즘, 양방향 탐색, GIS

1. 서 론

탐색(searching) 문제는 다양한 분야에서 제기된 문제로서 알고리즘, 인공 지능 등에서 오랫동안 연구를 진행하여 많은 탐색 방법이 발표되었다. 탐색에 대한 사전 지식이 없을 경우 넓이 우선 검색(breadth first search), 깊이 우선 검색 (depth first search) 등이 널리 사용되고, 사전 지식이 주어졌을 경우에는 최선 우선 검색 (best first search), Hill-climbing, Simulated Annealing 등의 방법이 사용되고 있다 [1,2].

인터넷 통신의 발달과 더불어 웹을 이용한 응용 분야 중 도로에 대한 정보를 제공하고, 목적지와 출발지간의 최적 경로를 탐색해 주는 서비스가 활성화 되고 있다. 이는 최근 많은 차량의 증가로 인하여 교통 상황이 열악해 지고 있는 상황에서 일반 사용자에게 많은 호응을 얻고 있으며 CNS(car navigation system) 등에서 핵심 기능으로 제공되고 있으며 향후 그 활용성은 더욱 증가할 것으로 기대되고 있다 [3,4].

도로에서의 최적 경로 탐색은 출발지와 목적지의 위치 정보를 알고 있는 상황에서 최적 경로를 탐색하는 경우로서 사전에 어느 정도의 지식이 주어진 탐색에 해당하며 최선 우선 탐색 방법의 일종인 단방향 A* 알고리즘과 양방향 A*알고리즘이 널리 사용되고 있다[5,6]. 그러나 단방향 A* 알고리즘은 최적 경

† 종신회원, 경원대학교 소프트웨어대학 소프트웨어학부 조교수

로 탐색을 보장해 주는 반면 탐색 시간이 많이 소요되며 양방향 A* 알고리즘은 전방 탐색과 후방 탐색이 만났을 때 탐색을 종료하는 것으로 한다면 탐색 시간은 단방향에 비하여 적게 소요되나 최적 경로에 대한 보장을 할 수 없다[7]. 이 경우(전방 탐색과 후방 탐색이 만난 경우) 탐색된 경로의 비용은 최적 경로의 비용과 평균적으로 큰 차이는 없으나, 만일 양방향에서 최적 경로를 탐색하고자 한다면 일반적으로 탐색 시간이 단방향에 비하여 오히려 증가하며 최악의 경우 탐색 시간이 2배 정도 소요될 수도 있다[6].

본 논문에서는 양방향 A*알고리즘이 왜 최적 경로를 보장해 주지 못하는 가에 대하여 고찰해 보고 최적 경로를 보장하며 탐색 시간이 적게 소요되는 새로운 양방향 A* 알고리즘을 제안한다. 또한 제안된 양방향 A*알고리즘이 단방향 A* 알고리즘에 비하여 어느 정도의 탐색 시간 절약 효과가 있는지를 실험을 통하여 고찰해 보았다.

본 논문은 2장에서는 최적 경로 탐색에 대한 관련 연구를 살펴보고 3장에서는 최적 경로를 보장하는 양방향 A*알고리즘을 제안하고 그의 타당성을 증명한다. 4장에서는 제안된 알고리즘의 효용성을 확인하기 위하여 실제 도로에 적용한 결과를 보여주고 있다. 마지막으로 5장에서는 결론과 함께 향후 진행될 연구 방향에 대하여 논의하였다.

2. 관련 연구

각각의 점(node)들이 서로 연결된 그래프에서 출발점(s)과 도착점(t) 사이의 최적 경로의 탐색은 전통적인 그래프 탐색 문제로서 매우 다양한 방법이 제안되었으며, 각각의 방법은 탐색 소요 시간, 메모리 사용 등에서 많은 차이를 보이기도 한다. 출발점에서 도착점의 방향으로 진행하는 경우 단방향 탐색이라고 하고, 도착점이 하나일 경우 출발점에서 도착점으로 탐색하는 전방 탐색과 도착점에서 출발점으로 탐색을 진행하는 후방 탐색을 함께 진행하는 방법을 양방향 탐색이라고 한다[10]. 이 때 양방향 탐색에서 후방 탐색의 경우 목적지에서 출발지 방향으로의 경로를 사용하는 것이 아니라 출발지에서 목적지 방향으로의 경로를 사용한다.

출발점과 도착점의 위치에 대한 정보가 있을 경우 최적 우선 탐색의 일종인 A*알고리즘이 널리 사용된다[14]. A*알고리즘에서 경로 비용(f(n))은

$$f(n) = c(n) + h(n)$$

이 사용되며 여기서 c(n)은 출발점에서 점 n까지의 경로 비용이고 h(n)은 점 n에서 목적지까지의 직선거리를 나타낸다. A* 알고리즘에서는 다음에 진행할 수 있는 점들을 $OPEN$이라고 하는 리스트에 넣고 이들 점들의 경로 비용을 산출하여 가장 경로 비용이 적은 것을 다음에 진행할 점으로 선택한다. 단방향 A* 알고리즘은 최적 경로를 보장하고 다른 단방향 탐색 방법에 비하여 매우 우수한 것으로 알려져 있으며[8], 단방향 A* 알고리즘에서 메모리 사용을 개선한 IDA*(iterative-deepening A*) 방법이 Korf에 의해 제시되었다[9].

단방향 A*알고리즘이 매우 우수한 방법이므로 이를 개선할 방법을 양방향에서 찾으려고 하는 시도가 있었으며 Pohl에 의해 처음으로 BHPA라고 불리는 양방향 A*알고리즘이 제안되었다[10]. BHPA 알고리즘에서는 전방 탐색의 선두와 후방 탐색의 선두가 만났을 때 탐색을 종료하는 것으로 하였으며, 만일 점 n에서 두 탐색의 선두가 만났다고 하면 전체 경로 비용은

$$L = g_1(n) + g_2(n)$$

으로 표현되고 여기서 g_1(n)은 출발점에서 점 n까지의 경로 비용을 나타내고, g_2(n)는 점 n에서 도착점까지의 경로 비용을 나타낸다. 문제는 g_1(n)과 g_2(n)를 합친 L이 출발점에서 도착점까지의 최적 경로 비용이 아닐 수도 있다는 것이다(증명은 3장 참조).

BHPA에서 최적 경로를 보장받기 위해서는

$$L_{\min} \le \max\left(\min_{x \in OPEN_1} f_1(x), \min_{x \in OPEN_2} f_2(x) \right) \quad (1)$$

의 조건이 만족되어야 한다. 여기서 $OPEN_1$은 전방 탐색 시 다음에 진행할 수 있는 점들을 위한 리스트이며 $OPEN_2$는 후방 탐색을 위한 리스트이다. $L_{\min}$은 지금까지 탐색된(전방 탐색과 후방 탐색의 선두가 만난 경로 중) 최저 경로 비용을 의미하며, f_1(x)은 전방 탐색 시 출발점에서 점 x까지의 경로 비용과 점 x에서 목적지까지의 직선거리의 합을 나타내며, f_2(x)은 후방 탐색의 경우를 나타낸다.

식 (1)은 전방 탐색과 후방 탐색의 선두가 만났을

때 $OPEN_1$과 $OPEN_2$ 내에 있는 모든 점들의 $f(x)$값들이 지금까지의 최저 경로 비용보다 커야만 한다는 것으로, 이를 확인하는데 많은 시간이 소요되는 관계로 양방향 탐색의 효율성을 저하시키고 경우에 따라 단방향에서의 탐색 시간보다 최악의 경우 2배 가까이 소요될 수도 있다[11].

이러한 문제점을 개선하기 위하여 전방 탐색의 선두 노드에서 후방 탐색의 $OPEN_2$의 각 점들 간의 거리를 계산하고, $OPEN_2$의 점에서 도착점까지의 경로 비용을 합친 것을 경로 비용을 사용하는 front-to-front라는 방법이 제안되었다[12,13]. 그러나 이 방법은 탐색 노드의 수는 줄어드나 계산 비용의 증가로 인하여 전체적인 탐색 비용의 절감에는 그다지 효과적이지 못한 것으로 알려졌고, perimeter search라 불리는 방법도 제안되었으나 이 방법 또한 탐색 노드의 수는 줄어드나 전통적인 양방향 A* 알고리즘에 비하여 계산량이 많이 증가하게 된다[14,15].

3. 제안 알고리즘

일반적으로 양방향 A* 알고리즘은 출발지에서 목적지로 탐색을 진행하는 전방 탐색과 목적지에서 출발지로 탐색을 진행하는 후방 탐색을 동시에 진행하며 전방 탐색과 후방 탐색이 중간에서 만났을 경우 탐색을 종료하게 된다. 이 때 전방 탐색과 후방 탐색이 만난다는 것을 그림 3.1(a)에서와 같이 전방의 선두와 후방의 선두가 만났을 때를 만나는 것으로 정의할 수 있고 그림 3.1(b)에서와 같이 전방이나 후방의 선두가 이미 지나온 점을 만났을 때도 만난 것으로 정의할 수 있다. 본 논문에서 전방과 후방 탐색이 만난다는 것은 이 두 경우를 모두 포함하며, 이 두 경우 탐색을 종료한다면 탐색된 경로는 최적 경로를 보장할 수 없다는 점을 정리 1에 증명하였다.

정리 1) 양방향 A* 알고리즘에서 전방 탐색과 후방 탐색이 만났을 때 탐색이 완료한 것으로 한다면, 양방향 A* 알고리즘은 최적 경로를 보장하지 않는다.

증명 : 증명은 매우 간단하며 그림 3.2와 같이 후방 탐색이 점 p까지 탐색되었고 전방 탐색의 경우 $OPEN_1$리스트에 n_1과 n_3가 있다고 하자. 이 때 n_1에 연결된 점이 p이고 n_3에 연결된 점은 n_4이므로 $f_1(p)=c_1(p)+h_1(p)$와 $f_1(n_4)=c_1(n_4)+h_1(n_4)$의 값을 비교하여 적은 비용을 갖는 점을 선택하여 탐색을 진행하게 된다. 만일 $f_1(p)$의 값이 $f_1(n_4)$의 값보다 적다고 하면 점 p로 진행하여 전방 탐색과 후방 탐색이 만나게 된다. 이 때 $f_1(p)$와 $f_1(n_4)$와의 사이에 다음의 관계가 성립될 수 있다.

$$f_1(p)<f_1(n_4),\ c_1(p)>c_1(n_4),\ h_1(p)<h_1(n_4)$$

즉 전방 탐색 시 출발점에서 점 p까지의 경로 비용이 점 n_4까지의 경로 비용보다 큼에도 불구하고 목적지까지의 직선거리가 가깝기 때문에 선택이 된 경우가 발생한 것이다. 이 경우는 후방 탐색의 경우에도 마찬가지로 발생할 수 있는데 그림 3.2에서 n_2에서 p가 선택되기 직전 $OPEN_2$에 점 n_2와 n_5가 있었고, $f_2(p)$과 $f_2(n_4)$를 비교하여 $f_2(p)$가 적어서 p로 탐색이 진행되었으나 위의 경우에서와 같이 $c_2(p)>c_2(n_4)$, $h_2(p)<h_2(n_4)$가 성립될 수 있다. 이와 같은 경우는 비록 양방향 탐색이 만난 경로는 n1-p-n2의 경로이나 n3-n4-n5를 잇는 경로의 경로 비용이 더 적다는 것을 알 수 있다.

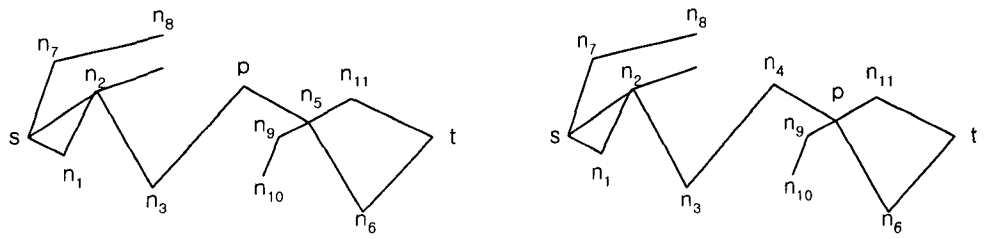

그림 3. 1 양방향 탐색이 만난 경우 (a) s-n_1-n_2-n_3-p 탐색이 되었고, t-n_6-n_5에서 p로 진행한 경우 (b) t-n_{11}-p-n_9 탐색이 되었고 s-n_1-n_2-n_3-n_4에서 p로 진행한 경우

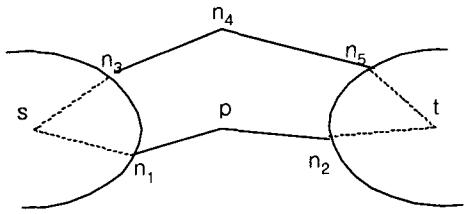

그림 3.2 양방향 탐색이 점 p에서 만난 경우

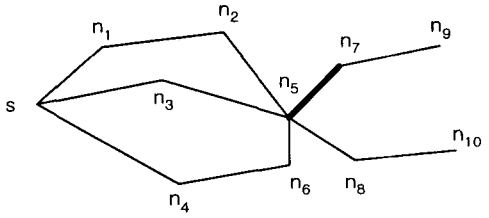

그림 3.3 링크 n_5n_7을 지나는 경로가 여러 개 있는 경우

양방향 탐색이 선두가 만난 경우가 아니라 그림 3.1 (b)와 같이 이미 지나온 점을 만났을 때도 같은 방법으로 증명될 수 있다. ■

이러한 이유로 인하여 최적 경로를 보장하기 위해서는 양방향 A* 알고리즘에서 두 탐색이 만났을 때 식 (1)의 조건이 만족할 때까지 탐색을 계속 수행하여야 한다. 이것이 양방향탐색의 탐색 시간을 증가시키는 요인이 된다. 그러나 양방향 탐색의 교차로 이루어진 경로는 최적 경로는 보장하지 못하지만 실험에 의하면 최적 경로와 많은 차이가 없으므로 이 경로를 준최적 경로(quasi-optimal path)로서 사용한다면 탐색 비용은 매우 절감될 수도 있다.

본 논문에서 제시하는 알고리즘은 기존의 양방향 A*알고리즘을 크게 2가지 점에서 개선하였다. 첫째는 비교적 간단한 개선으로 반복 탐색을 방지하기 위한 방법으로 정리 2에 방법과 그에 대한 증명을 제시하였다. 둘째는 기존의 방법보다 탐색 시간을 대폭 줄일 수 있는 중요한 개선으로 전방과 후방 탐색이 만났을 때 최적 경로를 보장하기 위해서 수행하는 식 (1)에 해당하는 부분으로 새로운 알고리즘을 정리 3에 그 증명과 함께 제시하였다.

정리 2) 각 점들을 연결하는 선을 링크(link)라고 할 때 A*알고리즘에서 처음 링크를 사용하는 경로가 다음에 같은 링크를 사용하는 경로보다 경로 비용이 적다. 따라서 최적 경로 탐색에서 한번 지나온 링크를 다시 사용하는 경로는 탐색할 필요가 없다.

증명 : 전방 탐색의 경우만을 고려하여 보면 그림 3.3에서 $OPEN_1$리스트에 n_2, n_9, n_6가 있다고 하고 n_7의 경로는 s-n_3-n_5-n_7의 경로를 거쳤다고 하자. 이는

$$f(n_7) \le f(n_2),\ f(n_7) \le f(n_6)$$

의 관계가 성립한다는 것을 의미한다. 따라서 링크 n5-n7을 사용하는 다른 경로인 s-n_1-n_2-n_5-n_7과 s-n_4-n_6-n_5-n_7의 경로는 s-n_3-n_5-n_7의 경로보다 경로 비용이 크다는 것을 의미하므로, 이러한 경로는 최적 경로로 선택될 수 없으므로 무시할 수 있다. 이 정리는 도로에서 P-turn을 해결하는데 매우 효과적으로 사용될 수 있다. ■

양방향 A* 알고리즘에서 양방향 탐색이 만났을 때 그림 3.4에서 일점쇄선으로 표시된 D선과 같이 출발지와 목적지를 잇는 선과 수직이며 교차점을 지나는 직선을 분할선이라고 정의 한다. 출발지와 목적지가 이 분할선을 중심으로 서로 반대편에 있을 경우, 전방 탐색에서 $OPEN_1$리스트에 있는 점들은 이 분할선을 중심으로 출발지 쪽에 있거나 혹은 목적지 쪽에 있고, $OPEN_2$리스트에 있는 점들도 출발지 쪽에 또는 목적지 쪽에 있다. $OPEN_1$, $OPEN_2$리스트에 있는 점들의 분할선까지의 수직 거리, $l_i(x)$는 다음과 같이 표현한다.

$$l_1(x) = \frac{(s-p)\cdot(x-p)}{|(s-p)|},\quad x \in OPEN_1$$

$$l_2(x) = \frac{(t-p)\cdot(x-p)}{|(t-p)|},\quad x \in OPEN_2$$

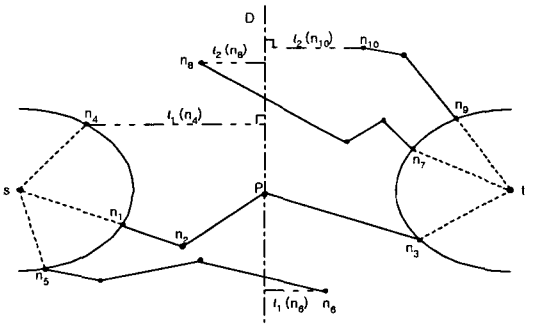

그림 3.4 양방향 탐색 만났을 때 노드에서 분할선까지의 거리

정리 3) 양방향 A*알고리즘에서 최적 경로를 보장되기 위해서는

$$L^{1}_{min} \le \min(c_1(x)+l_1(x)), \quad x \in OPEN_1$$
$$L^{2}_{min} \le \min(c_2(x)+l_2(x)), \quad x \in OPEN_2 \qquad (2)$$

을 만족하여야 한다. 여기서 L^{1}_{min}과 L^{2}_{min}은 지금까지 탐색된 최저 경로 비용를 갖는 경로에서, L^{1}_{min}은 출발점에서 두 탐색이 만난 교차점까지의 경로 비용을 나타내고 L^{2}_{min}은 교차점에서 목적지까지의 경로 비용을 나타내며 전체 경로 비용은 두 경로 비용의 합인

$$L_{min} = L^{1}_{min} + L^{2}_{min}$$

로 표현된다.

증명 : 양방향 탐색에서 두 탐색이 서로 만났고 이로서 만들어진 경로가 지금까지 탐색된 출발지와 목적지간의 경로 비용 중 가장 적다고 가정하자. 그리고 이 경로보다 경로 비용이 적은 또 다른 최적 경로가 어딘가에 존재한다고 가정하고, 이해를 돕기 위해 그림 3.5에서 점 p는 양방향 탐색이 만난 교차점이고, s에서 n_1까지 그리고 점선을 거쳐 n_2에서 t까지의 경로가 최적 경로라고 하자.

n_1, n_3는 $OPEN_1$리스트에 있는 점이고 n_2, n_4는 $OPEN_2$리스트에 있는 점이며 이중 n_1 과 n_2가 확장하여 최적 경로를 만들게 된다. 이러한 최적 경로가 존재할 때 n_1과 n_2는 다음의 4가지 경우 중 하나의 경우에 해당된다.

① n_1은 분할선을 중심으로 출발점 쪽에 있고 n_2는 목적지 쪽에 있는 경우. 이 때 수직 거리는 $l_1(n_1)>0$, $l_2(n_2)>0$가 된다.

② n_1은 분할선을 중심으로 목적지 쪽에 있고 n_2도 목적지 쪽에 있는 경우. 이 때 수직 거리는 $l_1(n_1)<0$, $l_2(n_2)>0$가 된다.

③ ②와 반대의 경우로, 이 때는 $l_1(n_1)>0$, $l_2(n_2)<0$가 되며

④ n_1은 목적지 쪽에 있고 n_2는 출발지 쪽에 있는 경우, 이 때는 $l_1(n_1)<0$, $l_2(n_2)<0$가 된다.

①의 경우는 그림 3.5에서 n_1, n_2경우로서 출발지에서 n_1까지 경로 비용과 분할선까지의 수직 거리의 합이 L^{1}_{min}보다 크고, n_2에서 도착점까지의 경로비용과 분할선까지의 수직 거리의 합이 L^{2}_{min}보다 크다고 하자. 그러면 n_1과 n_2가 만나서 이루는 경로는 L_{min}보다 크게 되므로 최적 경로가 될 수 없으며 따라서 L_{min}이 최적 경로값이 된다.

④의 경우는 n_1, n_2의 경로 비용에서 수직 거리를 뺀 값이 L^{1}_{min}과 L^{2}_{min}보다 크므로 n_1과 n_2가 만나서 이루는 경로는 최적 경로가 될 수 없다.

②의 경우는 그림 3.6과 같은 경우로서 n_1에서 n_2까지 최단 거리는 $|l_2(n_2)+l_1(n_1)|$ ($l_1(n_1)<0$) 보다는 크거나 같다. 만일 n_1과 n_2를 지난 경로가 최단 경로라고 하면 경로 비용(L)은

$$L \ge c_1(n)+c_2(n_2)+|l_2(n_2)+l_1(n_1)| \ge$$
$$c_1(n)+c_2(n_2)+l_2(n_2)+l_1(n_1)$$

의 관계가 성립한다. 그러므로

$$c_1(n_1)+l_1(n_1) \ge L^{1}_{min}, \quad c_2(n_2)+l_2(n_2) \ge L^{2}_{min}$$

의 관계가 있다고 하면

$$L \ge L^{1}_{min} + L^{2}_{min} = L_{min}$$

가 되므로 n_1과 n_2를 지나는 경로가 최적 경로가 될 수 없으며 L_{min}가 최적 경로값이 된다.

③은 ②와 같은 방법으로 증명된다. ■

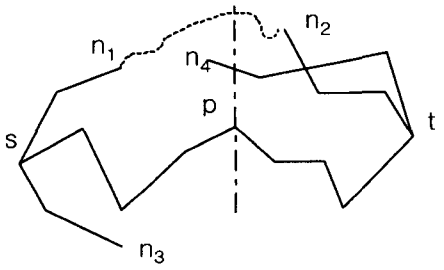

그림 3.5 양방향 탐색이 만나고 다른 최적 경로가 존재할 경우

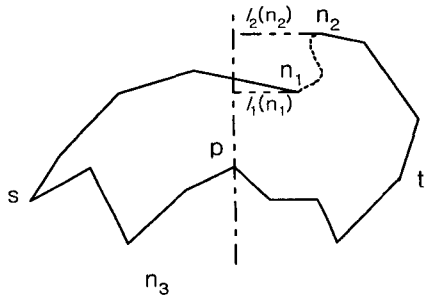

그림 3.6 $OPEN_1$과 $OPEN_2$의 점이 같은 쪽에 있을 경우

A*알고리즘에서 사용하는 경로 비용은 지금까지 진행되어온 경로의 비용과 현재의 위치에서 목적지까지의 직선거리를 사용한다. 직선거리의 사용은 탐색 공간을 한정하여 매우 효율적으로 최적 경로를 찾을 수 있도록 해주는 반면 양방향 탐색에서 두 탐색이 만났을 때 최적 경로를 보장해 주지 못하는 이유가 되기도 한다.

최적 경로 보장을 위하여 BHPA에서 사용하는 식(1)은 현재까지 탐색된 출발지와 목적지를 잇는 경로 중 최저의 경로 비용보다 더 작은 경로 비용이 없다는 것을 확인하기 위하여 전방 탐색의 모든 가능성과 후방 탐색의 모든 가능성에 대하여 조사한다. 즉 전체의 경로 비용을 사용하여 조사를 하게 되므로 탐색 시간이 단방향 탐색에 비하여 향상되기가 어렵고 오히려 경우에 따라 더 오래 걸릴 수도 있다. 반면 식(2)는 전체 경로 비용을 사용하지 않고 전방 탐색 비용과 후방 탐색 비용을 나누어 고려하므로 탐색 공간이 매우 줄어들게 되어 효율적이 된다. 즉 교차점까지의 비용인 $L^1_{\min}$와 $L^2_{\min}$을 사용하므로 탐색 공간이 매우 줄어들게 된다.

이러한 이유로 인하여 정리 3에 제안한 알고리즘은 BHPA방법에 비하여 탐색 노드의 수가 현저히 줄어들게 되고 따라서 탐색 시간도 줄어들게 된다. 제안한 방법의 효율성과 타당성을 확인하기 위하여 4장에서 실험 결과를 제시하였다.

4. 실 험

본 논문에서 제안한 알고리즘의 타당성을 검토하기 위하여 여의도 근처의 실제 지도를 사용하여 최적 경로를 탐색하는 실험을 수행하였다. 그림 4.1은 실험에 사용된 지도와 출발지와 목적지를 입력하였을 때 탐색된 최적 경로를 굵은 선으로 표시된 것을 보여주고 있다.

실험에 사용된 지도는 노드 수가 10590개이며, 이 노드들을 연결하는 링크의 수는 22874개이다.

실험은 표 4.1에서와 같이 30개의 서로 다른 출발지와 목적지를 입력하여 탐색된 최적 경로를 단방향 A*알고리즘과 BHPA방법 그리고 본 논문에서 제시한 방법(New로 표시)과 비교하였다. 비교는 탐색 비용, 탐색 노드 수, 그리고 탐색 시간에 대하여 비교하였다. 탐색 경로의 경우는 3 방법 모두 최적 경로를 탐색하는 일치된 결과를 보여주었다. 탐색 노드 수의 비교는 그림 4.2에서와 같이 본 논문에서 제시한 방법(New)이 단방향과 비교하여 약 1/2만큼, BHPA에 비하여는 1/3만큼 줄어드는 우수함을 보여주고 있다. 탐색 시간의 경우도 그림 4.3에서와 같이 단방향의 절반정도, BHPA의 약 1/3정도만큼 매우 빠른 것을 알 수 있다. 탐색 노드 수의 줄어듦에 비하여 탐색 시간이 같은 정도로 줄어들지 않는 것은 제안한 방법이 다른 방법보다 계산량이 약간 증가하였기 때문으

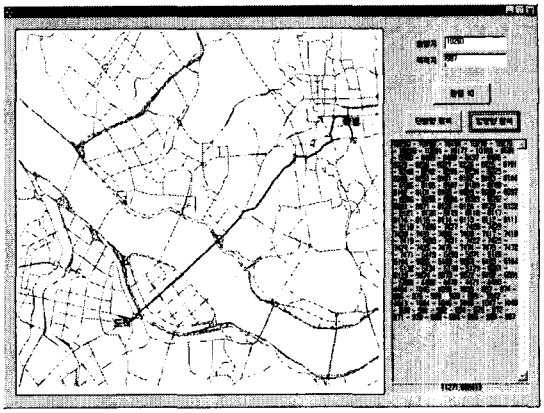

그림 4.1 실험에 사용된 지도와 최적 경로 탐색 결과 예

표 4.1 실험에 사용된 데이터 (New는 본 논문에서 제안한 방법)

출발점	도착점	경로 비용	탐색 노드 수			탐색 시간		
			단방향	BHPA	New	단방향	BHPA	New
5696	7706	6475.4	758	1544	537	6	14	5
3706	6277	1066.6	64	84	43	0	1	1
1293	818	5239.2	797	857	220	7	7	2
7885	2555	11076.3	1363	2068	180	12	17	2
2212	171	9570.0	1865	2602	1454	15	21	12
1684	7883	12690.4	3905	3783	1216	33	32	10
1928	8335	8693.0	2041	2929	1738	17	25	15
1822	5083	3720.3	421	591	216	4	5	2
3959	5148	4163.3	484	669	198	4	6	2
7085	9872	7271.5	1273	1994	783	11	17	6
3085	6967	3855.9	746	713	216	6	6	2
10094	4853	6525.8	1221	1874	1007	11	16	9
8836	6476	1640.9	50	55	26	0	1	1
7917	7502	8713.1	1493	2124	748	13	18	6
491	1709	6503.6	1413	1853	816	12	16	7
441	3222	1634.3	107	222	83	1	2	1
10507	4196	11650.0	2037	3949	2026	18	34	18
7794	4256	8822.2	1564	3059	781	14	27	7
5248	3875	6138.6	1209	1757	672	10	15	6
2664	6090	10880.6	2579	3094	1388	21	26	12
1884	5303	4437.1	590	730	264	5	6	2
2975	6608	13407.5	3281	4070	1391	27	34	12
2217	8300	5731.8	689	1407	323	6	12	3
4983	929	6787.8	1309	1836	616	11	15	5
3292	586	2396.3	328	399	220	3	3	2
6852	2886	12176.0	3631	4082	1196	31	33	10
2204	7207	9945.7	2510	3736	1563	21	32	13
8355	3646	12827.1	3820	4203	2550	33	35	22
9610	1330	9177.0	2280	3738	1629	19	32	14
8012	8594	2294.2	75	99	12	1	1	0
합계		215511.7	43903	60121	24112	372	509	209

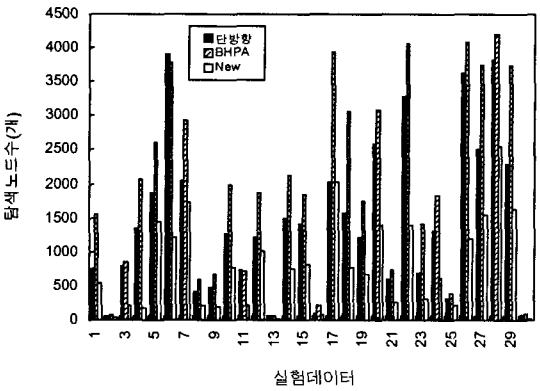

그림 4.2 탐색 노드 수 비교

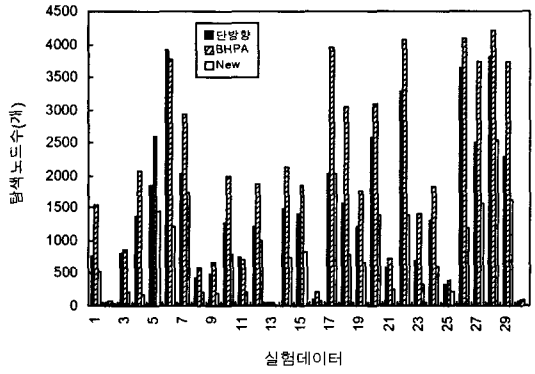

그림 4.3 탐색 시간 비교

로 판단된다.

5. 결 론

최적 경로 탐색 문제는 여러 분야에서 많은 연구가 되어 온 분야이나 최근에 지능형 교통 시스템이나 차량 항법 장치와 같은 분야에서 필요성이 대두되면서 효율적인 알고리즘을 찾고자 하는 연구가 진행되고 있다.

본 논문에서는 최적 경로 탐색에 많이 사용되는 A*알고리즘의 효율성을 매우 개선한 새로운 양방향 A*알고리즘을 제안하였다. 제안된 알고리즘은 실험을 통하여 그 성능을 검증하였으며 실험 결과 탐색 노드 수와 탐색 시간에서 매우 우수한 성능을 보여주는 것을 알 수 있었다.

본 논문에서 제안한 알고리즘은 일반적인 그래프 탐색에 적용되는 알고리즘으로 이것을 도로에서의 최적 경로 탐색에 적용할 경우 도로의 특성이나 교통량 등이 반영되면 더욱 효과적인 결과를 얻을 수 있을 것으로 판단된다.

참 고 문 헌

[1] S. Russel, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995.

[2] S. Even, Graph Algorithm, Computer Science Press, 1979.

[3] 송성훈, 김기섭. "실시간 차량경로결정을 위한 동적 최단경로 알고리즘", 과학기술연구논문집, Vol.10 No.2, 1999.

[4] 이승환, 최기주, 김원길. "도시부 ATIS 효율적 적용을 위한 탐색영역기법 및 양방향 링크탐색 알고리즘의 구현", 대학교통학회지, Vol 14. No. 3, 1996.

[5] P.E.Hart, et al. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Trans. Sys. Sci. and Cyb. SSC-4, pp100-107, 1968.

[6] T.Ikeda, M.Y.Hsu, et al."A Fast Algorithm For Finding Better Routes By AI Search Techniques", IEEE VNIS'94, 1994.

[7] 이재무, 김종훈, 정홍.석. "차량 항법 시스템의 경로 탐색을 위한 탐색 알고리즘들의 성능 비교", 한국정보교육학논문집, Vol 2, No 2, 1998

[8] R. Dechter, J. Pearl, "Generalized best-first startegies and the optimality of A*", J. ACM, Vol. 32, No. 3, pp 505-536, 1985.

[9] R. Korf, "Depth-first iterative deepening: An optimal admissible tree search", Artificial Intelligence, Vol. 27, No. 1, pp 97-109, 1985

[10] I. Pohl, Machine Intelligence, pp. 127-140, Edinburgh Univeristy Press, 1979.

[11] H. Kaindl, G. Kainz, "Bidirectional Heuristic Search Reconsidered", J. of Artificial Intelligence Research, Vol 7, pp. 283-317, 1997.

[12] G. Politowski, I. Pohl, "D-node retargeting in bidirectional heuristic search", National Conf. on A.I.(AAI-91), pp. 434-440, The MIT Press, 1984.

[13] H. Davis, R. Pollack, T. Sudkamp, "Towards

a better understanding of bidirectional search", Proc. Fourth National Conf. on A.I. (AAAI-84), pp. 68-72, The MIT Press, 1984

[14] J. Dillenburg, P. Nelson, "Perimeter search", Artificial Intelligence, Vol. 65, No. 1, pp. 165-178, 1994.

[15] G. Manzini, "BIDA*: an important perimeter search algorithm", Artificial Intelligence, Vol. 75, No. 2, pp. 347-360, 1995.

황보 택근

1983년 고려대학교 공과대학 공학사
1987년 CUNY Computer Science 석사
1995년 S.I.T. Computer Science 박사
1988년~1993년 Q-Systems, Senior Technical Staff
1995년~1997년 삼성종합기술원 선임연구원
1997년~현재 경원대학교 소프트웨어대학 소프트웨어학부 조교수
관심분야 : 컴퓨터그래픽스, 영상처리, GIS 등