차량 항법 시스템의 경로 탐색을 위한 탐색 알고리즘들의 성능 비교

이재무* · 김종훈** · 전흥석***
부산교육대학교 컴퓨터교육과*, 전자통신연구원 OS연구팀**, Thinkware System***

요 약

차량 항법 시스템에서의 경로 탐색은 두 지점 간 최단 경로 탐색을 위한 알고리즘을 이용해서 이루어질 수 있다. 다양한 최단 경로 탐색 알고리즘들에 대한 성능 평가는 알고리즘의 적용 분야에 따라 제각기 달라질 수 있는데 차량 항법 시스템에서는 선정된 경로의 정확성과 경로 탐색에 소요되는 시간이 통합적으로 평가되어야 한다. 본 논문에서는 지금까지 알려진 다양한 경로 탐색 알고리즘들 중에서 차량 항법 시스템의 경로 안내에 최적의 알고리즘을 판단하기 위하여 시뮬레이션을 통해 각 알고리즘들의 성능을 비교 측정하였다. 시뮬레이션은 실용성을 위하여 실제의 디지털 도로 지도 데이터 베이스를 이용하여 실시하였다. 실험 결과에 의하면 양방향 탐색을 이용한 Bi-directional Dijkstra 알고리즘과 양방향 탐색과 경험적 탐색 방법을 함께 이용한 Modified Bi-directional A* 알고리즘이 다른 알고리즘들에 비해 가장 뛰어난 성능을 보여주었다.

Performance Evaluation of Different Route Planning Algorithms in the Vehicle Navigation System

Jaemu Lee* · Jonghoon Kim** · H.seok Jeon***
Pusan National University of Education*, ETRI**, Hong-Ik University***

ABSTRACT

Vehicle navigation systems employ a certain route planning algorithm that provides the shortest path between the starting point and the destination point. The performance of a given route planning algorithm is measured through the degree of optimal route selection and the time cost to complete searching an optimal path. In this paper, various route planning algorithms are evaluated through computer simulation based on a real digital map database. Among those algorithms evaluated in this paper, the Modified Bi-directional A* algorithm is found to be the best algorithm for use in vehicle navigation systems.

1. 서 론

차량 항법 시스템(Vehicle Navigation System)에서 경로 탐색(Route Planning)은 운전자에게 현 위치에서 목적지까지 가장 빠른 시간에 도달할 수 있

는 경로를 알려주는 것이다. 그러므로 경로 안내의 문제는 그래프 이론에서 두 지점간의 최단 경로 탐색을 위한 알고리즘을 이용하여 해결할 수 있다.

최단 경로 탐색을 위한 가장 대표적인 방법은 Dijkstra의 최단 경로 알고리즘(Shortest Path Algorithm)을 이용하는 것이다. Dijkstra 알고리즘은 출발지에서 인접한 노드들까지의 비용을 계산하여 가장 적은 비용의 노드로 이동하며, 이러한 과정을 목적지에 도착할 때까지 반복 수행한다. Dijkstra 알고리즘은 확실하게 최단의 경로를 선정할 수 있다는 장점이 있으나 목적지의 방향에 관계없이 모든 방향의 노드를 검색하게 된다는 단점이 있다. 모든 방향의 노드를 검색하게 되면 출발지와 목적지간의 거리가 매우 멀어서 탐색해야 될 경로가 매우 많을 경우 최단 경로를 선정하는데 소요되는 시간이 무시할 수 없을 정도로 커지게 된다. 이러한 경우에는 최단 경로를 선정하는데 소요되는 시간 또한 선정된 경로의 정확성 못지 않게 차량 항법 시스템의 성능을 평가하는 중요한 요소가 된다. 따라서 차량 항법 시스템에서 사용될 경로 탐색 알고리즘은 선정 경로의 정확성과 함께 빠른 시간 내에 최단 경로를 선정할 수 있어야 한다.

최단 경로의 탐색 시간을 줄이기 위해서 지금까지 여러 가지 해결책이 제시되었는데 가장 대표적인 방법으로 경험적 탐색(Heuristic Search) 방법[3]과 양방향 탐색(Bi-directional Search) 방법[4, 5], 그리고 이들을 혼합한 방법[1, 2, 6, 7] 등이 있다. 경험적 탐색 방법을 사용하는 A* 알고리즘[3]은 출발지 노드에서 현재 노드의 인접한 노드들에 이르는 거리뿐만 아니라 현재 노드의 인접한 노드들에서 목적지까지의 거리도 고려하여 이동 노드를 선택한다. 이를 통해 불필요한 노드의 계산을 제거하여 전체적으로 탐색 시간을 줄이기 위한 방법이다. 그러나 A* 알고리즘은 Dijkstra 알고리즘에 비해 인접한 노드들에서 목적지까지의 거리 계산을 추가로 해야 하는 부담을 안고 있다. 또한 A* 알고리즘은 선정된 경로가 출발지에서 목적지까지의 최단 경로라는 보장은 할 수 없다. 최단 경로 탐색 시간을 줄이기 위한 또 다른 방법으로 양방향 검색 방법[4, 5]이 있다. 이는 Dijkstra 알고리즘의 경로 탐색이 출발지에서 목적지로의 한 방향으로 이루어지고 있는데 반해 양쪽 방향에서 모두 탐색을 진행하는 방법이다. 즉, 출발지에서 목적지 방향으로 탐색하는 전방 탐색(forward search)과 목적지에서 출발지 방향으로 탐색하는 후방 탐색(backward search)을 병행하여 서로의 탐색이 중간에 교차될 때 탐색을 완료하게 된다. 이론적으로 양방향 탐색은 단 방향 탐색에 비해 전체 탐색 노드 수의 절반을 줄일 수 있는 장점이 있다. 그러나 만일 양방향 탐색은 양단에서의 탐색이 교차하지 않을 경우 오히려 단 방향 탐색에 비해 최고 두 배의 노드들을 탐색해야 하는 단점이 있다. 경험적 탐색 방법과 양방향 탐색 방법을 함께 사용하는 방법[1, 2, 6, 7]도 있다. 이는 말 그대로 경험적 탐색 방법을 출발지에서 목적지로, 목적지에서 출발지로의 양방향으로 진행하는 방법이다. 이는 두 가지 접근 방법의 장점을 모두 살리기 위해 시도된 방법이다. 그러나 이로 인해 두 가지 방법의 단점 역시 모두 가지게 된다.

그렇다면 과연 지금까지 제안된 여러 가지 최단 경로 탐색 알고리즘들 중에서 차량 항법 시스템의 경로 안내에 최적의 성능을 보여줄 수 있는 알고리즘은 무엇인가? 본 논문에서는 이러한 문제에 대한 해답을 얻기 위하여 차량 항법 시스템의 경로 탐색을 위한 다양한 알고리즘들을 시뮬레이션을 통해 성능 비교를 하였다. 시뮬레이션은 실험 결과의 실용성을 위하여 실제의 디지털 도로 지도 데이터 베이스를 이용하여 진행하였다.

실험 결과에 의하면 Dijkstra 알고리즘은 가장 최단의 경로를 선정하지만 경로 탐색 속도가 느려 효율성이 떨어진다. 선정된 경로의 정확성과 경로 탐색에 소요되는 시간을 통합적으로 고려해 볼 때 양방향 탐색을 이용한 Bi-directional Dijkstra 알고리즘과 양방향 탐색과 경험적 탐색 방법을 함께 이용한 Modified Bi-directional A* 알고리즘이 높은 정확성을 바탕으로 빠른 시간 내에 최단 경로를 선정하여 다른 알고리즘들에 비해 가장 뛰어난 성능을 보여주었다.

본 논문의 나머지 부분은 다음과 같이 구성되어 있다. 2장에서는 최단 경로 탐색을 위한 여러 가지 알고리즘들의 내용 및 특성들을 상세히 살펴보고, 3장

에서는 각 알고리즘들을 시뮬레이션한 결과와 실험 결과를 바탕으로 각 알고리즘들의 성능을 비교 분석하며, 4장에서 결론 및 향후 연구를 제시한다.

2. 경로 탐색 알고리즘들

이 장에서는 지금까지 제안되었던 다양한 경로 탐색 알고리즘들의 내용 및 특성들을 간략하게 정리한다.

2.1 Dijkatra 알고리즘

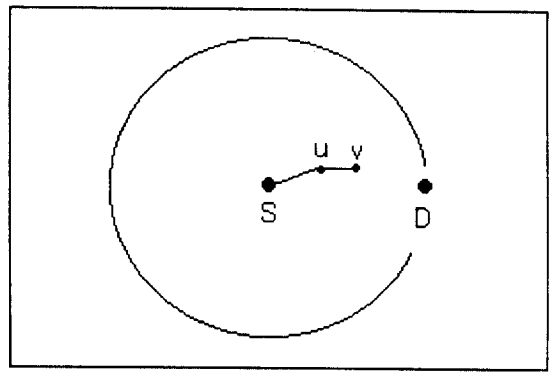
Dijkstra 알고리즘은 최단 경로 탐색을 위한 가장 기본적인 알고리즘이다. Dijkstra 알고리즘은 식 (1)과 같이 출발지 u 노드에서 인접한 모든 v 노드까지의 비용 g를 두 노드간의 거리 L(u,v)로 계산하여 노드 평가 함수 f에 할당한 후 f의 값이 가장 작은 노드로 이동하며 이를 목적지에 도착할 때까지 반복 수행하는 알고리즘이다.

$$f = g = L(u,v) \qquad (1)$$

Dijkstra 알고리즘을 사용하면 노드의 검색 반경이 <그림 1>과 같이 출발지를 중심으로 동심원을 그리며 넓어지게 된다. Dijkstra 알고리즘은 확실하게 최단 경로를 선정할 수 있는 장점이 있으나 목적지의 방향에 관계없이 모든 방향의 노드를 검색하게 되어 검색 시간이 늦어지는 단점이 있다.

2.2 A* 알고리즘

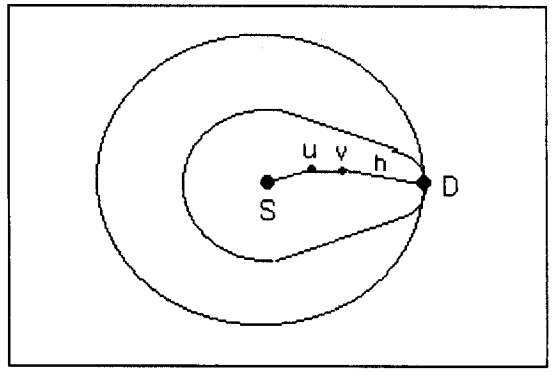
A* 알고리즘은 목적지의 방향에 관계없이 모든 방향의 노드를 모두 검색하는 Dijkstra 알고리즘의 비효율성을 해결하기 위해 제시된 알고리즘들 중의 하나이다. A* 알고리즘에서는 탐색 노드의 선정에 있어서 Dijkstra 알고리즘에서 사용하는 출발지와의 거리 g 이외에 식 (2)와 같이 목적지까지 직선 거리 h를 추가하여 선정한다.



<그림 1> Dijkstra 알고리즘의 검색 영역
(S: starting point, D: destination point,
u: current node, v: adjacent node of u)

$$f = g + h \qquad (2)$$

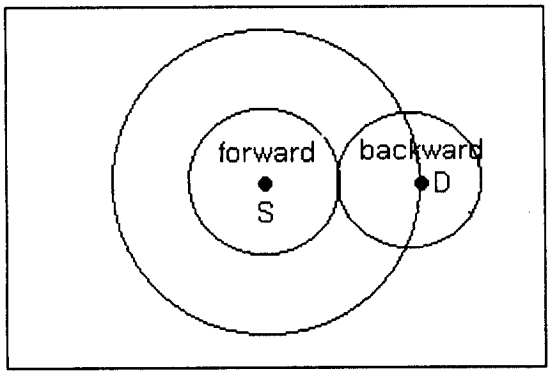
이로 인해 <그림 2>와 같이 목적지에서 멀어지는 노드에 대한 불필요한 계산을 생략하게 되어 전체적인 검색 속도를 증가시키고자 하는 것이다. 물론 A* 알고리즘은 Dijkstra 알고리즘에 비해 h를 계산하기 위한 추가적인 계산의 부담을 가지고 있다. 또한 A* 알고리즘은 h의 사용으로 인해 선정된 경로가 최적의 경로임을 보장할 수 없게 된다.



<그림 2> A* 알고리즘의 검색 영역
(S: starting point, D: destination point,
u: current node, v: adjacent node of u)

2.3 Bi-directional Dijkstra 알고리즘

Bi-directional Dijkstra 알고리즘은 Dijkstra 알고리즘의 전체 검색 반경을 줄이기 위해 출발지에서 목적지로의 검색뿐만 아니라 동시에 목적지에서 출발지로의 검색을 교대로 수행한다. 결국 최단 경로 탐색은 양단에서의 검색이 교차되는 시점에서 종료되어지며 양방향에서 선정한 최단 경로의 합이 출발지에서 목적지까지의 최단 경로가 된다. Dijkstra 알고리즘의 검색 반경이 출발지를 중심으로 동심원을 그리며 퍼지기 때문에 Bi-directional Dijkstra 알고리즘은 이론적으로 <그림 3>과 같이 검색 반경이 Dijkstra 알고리즘에 비해 절반으로 줄어들어 경로 탐색 시간을 줄이게 된다. 그러나 Bi-directional Dijkstra 알고리즘 역시 Dijkstra 알고리즘과 같이 목적지의 방향과 관계없이 모든 방향의 노드를 탐색하는 단점을 여전히 지니고 있다.



<그림 3> Bi-directional Dijkstra 알고리즘의 검색 영역
(S: starting point, D: destination point)

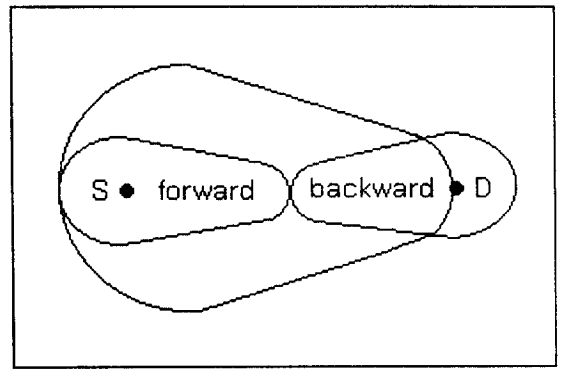
2.4 Bi-directional A* 알고리즘

Bi-directional A* 알고리즘은 Dijkstra 알고리즘의 단점을 해결하기 위한 A* 알고리즘과 Bi-directional Dijkstra 알고리즘을 통합한 알고리즘이다. 즉, 출발지와 목적지에서 교대로 탐색을 수행하며 이때 탐색 노드의 선정을 위해 탐색 시작 노드부터의 거리 이외에 탐색 종료 노드까지 직선 거리 h를 함께 고려한다.

이를 통해서 전체 검색 반경을 <그림 4>와 같이 줄이고 불필요한 계산을 생략하여 최단 경로 탐색 시간을 줄이게 된다. 그러나 Bi-directional A* 알고리즘은 A* 알고리즘과 마찬가지로 h의 사용으로 인해 선정된 경로가 최단의 경로임은 보장할 수 없다.

2.5 Modified bi-directional A* 알고리즘

Bi-directional A* 알고리즘의 가장 큰 단점은 출발지와 목적지간의 경로가 여러 가지일 경우에 양단에서의 검색이 반드시 교차하지는 않는다는 것이다. 양



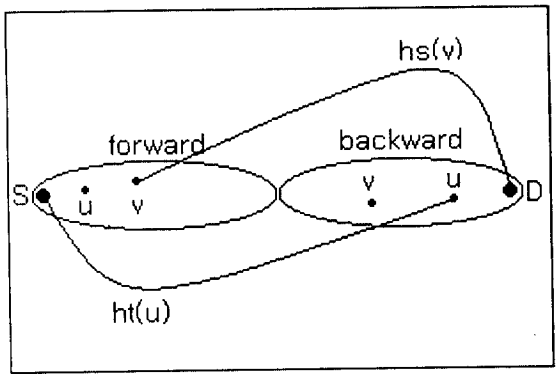
<그림 4> Bi-directional A* 알고리즘의 검색 영역
(S: starting point, D: destination point)

단에서의 검색이 교차하지 않을 경우에는 오히려 A* 알고리즘에 비해 Bi-directional A* 알고리즘이 두 배의 탐색 비용을 요구하게 된다. 양단에서의 검색이 교차하지 않는 이유는 내부의 두 개의 A* 알고리즘들이 각기 독립적인 heuristic function을 사용하기 때문[1]이다. 이러한 문제점을 해결하기 위하여 Ikeda et al. [1]은 A* 알고리즘을 Dijkstra 알고리즘으로 변환하는 기법을 이용한 Modified Bi-directional A* 알고리즘을 제안하였다. Modified Bi-directional A* 알고리즘에서는 Dijkstra 알고리즘에서 사용하는 u와 v 두 노드간의 길이 L 대신에 식 (3)과 같은 새로운 노드간의 길이 L'를 사용한다. 식

(3)에서 h_s는 현재 노드에서 목적지까지의 직선 거리를 의미하고 h_t는 현재 노드에서 출발지까지의 직선 거리를 의미한다. k_1과 k_2는 L'의 결정에 있어서 전방 탐색과 후방 탐색의 비중을 결정하는 상수이다.

$$L'(u,v) = L(u,v) + k_1(h_s(v) - h_s(u)) + k_2(h_t(u) - h_t(v)) \quad (3)$$

식 (3)에서 최적의 성능을 보이는 k_1과 k_2의 값은 도로의 특성에 따라 가변적이며 본 논문에서는 Ikeda et al. [1]의 논문에서 k_1과 k_2의 값이 각각 1/2일 경우에 도로의 특성에 관계없이 가장 안정적으로 경로 탐색을 진행한다고 밝힌 바에 따라 k_1과 k_2의 값으로 1/2을 사용하였다. Modified Bi-directional A* 알고리즘에서는 전방 탐색과 후방 탐색 모두 탐색 시작 노드와 탐색 종료 노드를 고려하므로 양단에서의 검색이 교차하지 않게 되는 Bi-directional A* 알고리즘의 문제를 해결하고자 하였다. <그림 5>는 Modified Bi-directional A* 알고리즘의 줄어든 검색 반경과 양단의 검색이 교차하는 모습을 보여준다.



<그림 5> Modified bi-directional A* 알고리즘의 검색 영역
(S: starting point, D: destination point, u: current node, v: adjacent node of u)

3. 성능 평가

본 장에서는 다양한 최단 경로 탐색 알고리즘들의 성능을 비교하기 위하여 실제의 디지털 도로 지도 데이터 베이스를 가지고 시뮬레이션한 결과와 그 결과를 바탕으로 분석한 내용을 정리한다.

3.1 시뮬레이션 환경

3.1.1 시스템 환경

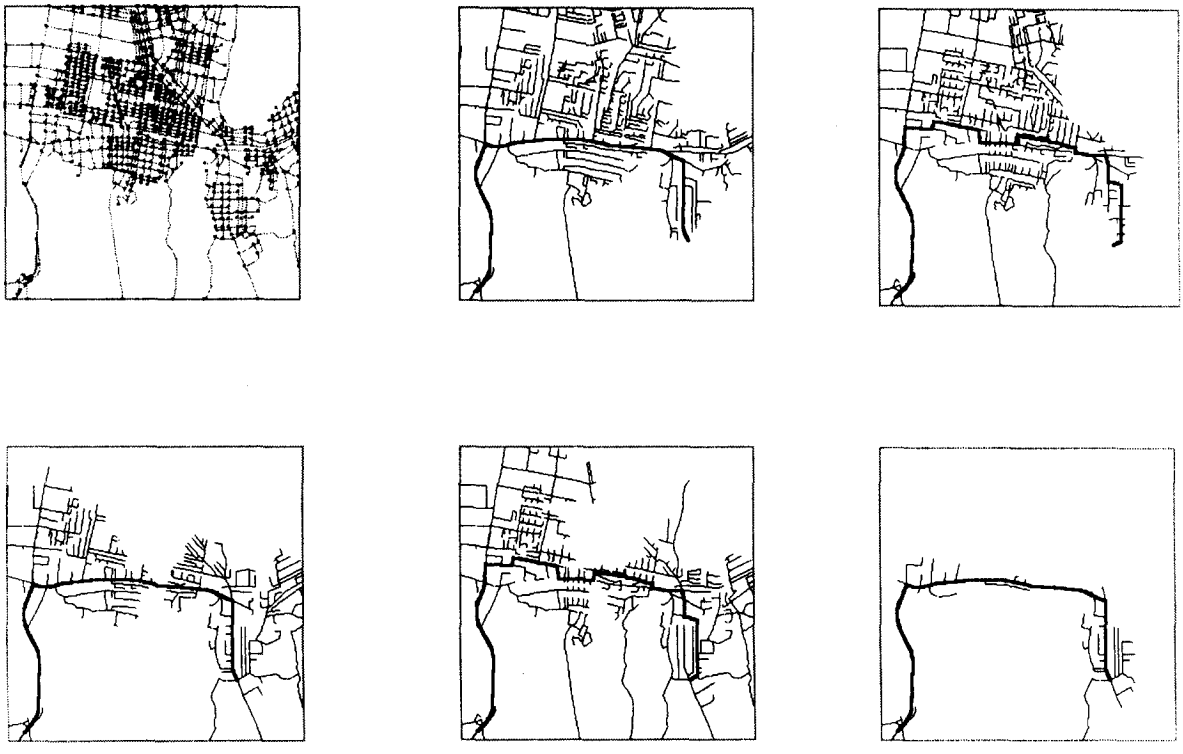
시뮬레이션은 windows95를 운영체제로 탑재한 Pentium 166MHz의 프로세서와 32MB의 주기억장치를 가진 컴퓨터 시스템에서 이루어졌다.

3.1.2 디지털 도로 지도 데이터 베이스

실험에 사용한 지도 정보는 NOVA Laboratory inc.에서 제공한 디지털 도로 지도 데이터 베이스를 사용하였다. 디지털 도로 지도 데이터 베이스는 이차원 Mesh 단위의 각 파일에 관리 자료, 도로망 자료 및 배경 자료를 수용하여 작성된 것이다. 본 논문에서는 디지털 도로 지도 데이터 베이스에서 서울과 경기도 지역의 지도 축적 1/50,000과 1/12,500의 기본 도로망에 관한 자료를 추출하여 실험에 사용하였다.

3.2 시뮬레이션 내용

차량 항법 시스템에서 경로 탐색 알고리즘의 성능을 평가하는 기준은 탐색된 최단 경로의 정확성과 탐색에 소요된 시간이다. 본 논문에서는 탐색 알고리즘들이 선정한 경로의 정확성을 측정하기 위하여 각 알고리즘을 이용하여 탐색된 최단 경로의 길이를 측정하였으며 탐색 속도를 측정하기 위해서 탐색 과정에서 방문했던 전체 노드 수와 각 경로 탐색 알고리즘 모듈의 호출에 소요된 시간을 측정하였다. 또한 최단 경로의 특성에 따른 각 알고리즘의 성능 변화를 알아보기 위하여 경로를 직선 길과 굽은 길로 분류하여 실험하였다. 직선 길은 출발지와 목적지간의 최단 경로가 직선의 형태를 나타내는 길을 의미하고 굽은 길은 직선의 형태를 나타내지 않는 길을 의미한다. 실험은 직선 길과 굽은 길 별로 각기 서로 다른 10여 가지씩, 총 20여 가지의 경로 탐색 실험을 진행하였다. 탐색에 소요된 시간의 정확성을 위하여 동일한 경로의 실험을 5회 반복 실시하여 소요 시간의 평균



<그림 6> 최단 경로 탐색 실험의 결과 예

을 해당 경로의 탐색에 소요된 시간으로 설정하였다.

3.3 시뮬레이션 결과 및 분석

<그림 6>은 실험에 사용된 맵(map)의 한 예와 각 알고리즘에 대한 최단 경로 탐색 결과이다. 맵은 차량 항법 시스템에서 지도 축척 1/12,500의 부천시 지역을 표현한 것인데 화면 축척이 약 5.5×4.5Km으로서 노드 수가 1,003개이며 링크 수는 1,636개를 포함하고 있다. 그리고 각 알고리즘에 대한 결과에서 굵은 선은 알고리즘 별로 선정된 최단 경로를 의미하고 가는 선은 경로 탐색 과정에서 방문했던 전체 노드들의 모습을 나타낸다.

3.3.1 정확성

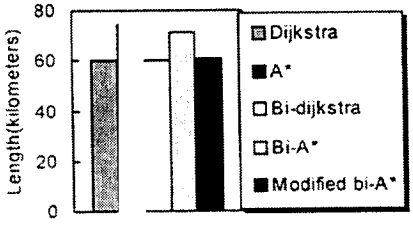
이 절에서는 각 경로 탐색 알고리즘들을 이용하여 선정된 탐색 경로의 정확성을 비교 분석한다. 이론적으로 Dijkstra 알고리즘이 최단의 경로를 선정하기 때문에 알고리즘들 간의 성능 차이를 알아보기 위하여 Dijkstra 알고리즘을 기준으로 다른 알고리즘들이 선정한 경로의 길이를 관찰하였다.

<그림 7>과 <그림 8>은 10회에 걸친 실험을 통해 각 알고리즘 별로 선정된 최단 경로의 길이의 합을 나타낸 그림이다. 실험 결과에 의하면 Bi-directional Dijkstra 알고리즘 그리고 Modified Bi-directional A* 알고리즘 등에 의해 선정된 최단 경로의 길이는 Dijkstra 알고리즘에 의해 선정된 최단 경로의 길이와 거의 비슷한데 비해 A* 알고리즘과 Bi-directional A* 알고리즘 등에 의해 선정된 최단 경로의 길이는 상대적으로 매우 길다는 것을 알 수 있다.

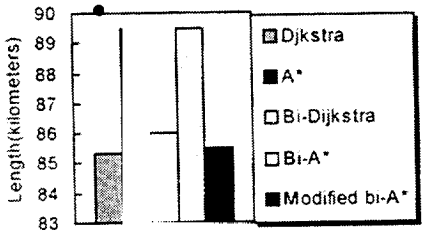
3.3.2 탐색 노드 수

일반적으로 경로 탐색 알고리즘의 특성을 파악하기

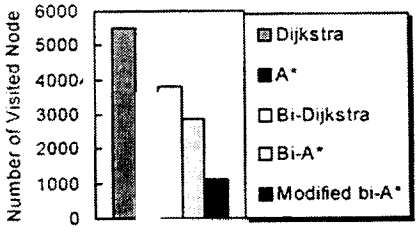
위해 가장 많이 사용하는 방법이 알고리즘의 수행 과정에서 최단 경로를 선정하기 전까지 방문하게 되는 전체 노드의 수를 관찰하는 것이다. 본 논문에서도 각 알고리즘 별 전체 방문 노드 수를 관찰해 보았는데 그 결과를 <그림 9>와 <그림 10>에 나타냈다. <그림 9>와 <그림 10>은 각 알고리즘들이 전체 실험을 통해 방문한 노드 수의 총 합을 도표로 나타낸 것이다.



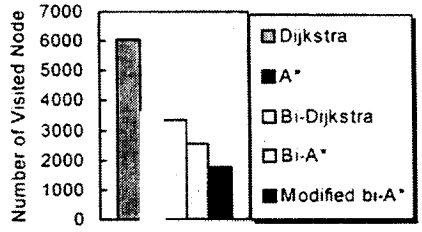
<그림 7> 직선 길 최단 경로 길이



<그림 8> 굽은 길 최단 경로 길이



<그림 9> 직선 길 탐색 노드 수

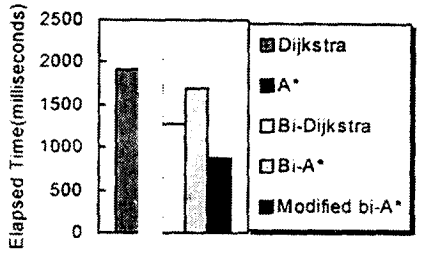


<그림 10> 굽은 길 탐색 노드 수

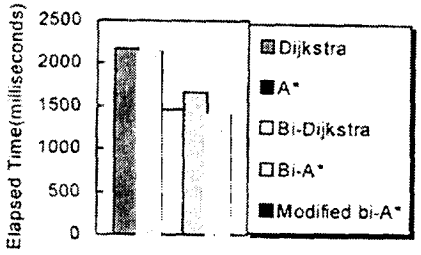
3.3.3 소요 시간

탐색 노드 수만으로 경로 탐색 알고리즘의 경로 탐색 속도를 평가하기에는 부족함이 있다. 왜냐하면 각 경로 탐색 알고리즘들이 탐색할 노드의 결정에 소요되는 시간이 각기 다르기 때문이다. 경우에 따라서는 탐색 노드 수는 적더라도 탐색 노드의 결정에 소요되는 시간이 많이 걸려 전체적으로 성능이 떨어지는 경우가 발생하게 된다. 따라서 본 논문에서는 각 경로 탐색 알고리즘별 최단 경로 선정에 소요되는 시간을 측정하였다. 전체 실험을 통하여 경로 탐색에 소요된 시간의 총합을 알고리즘 별로 나타낸 것이 <그림 11>과 <그림 12>이다.

탐색 시간을 줄이기 위하여 경험적 탐색 방법을 사용한 A* 알고리즘은 h 계산의 부담으로 인해 굽은 길 실험에서는 약간의 성능 향상을 보이고 직선 길에서는 오히려 속도가 저하되는 결과를 나타낸다. 따라서 h 계산의 부담을 줄이기 이전에는 A* 알고리즘은 별로 효과적이지 못한 방법으로 평가된다. 반면 양방향 탐색을 사용한 Bi-directional Dijkstra 알고리즘을 보면 직선 길과 굽은 길 모두에서 뛰어나 속도의 향상을 보이고 있다. 두 가지 방법을 모두 사용한 Bi-directional A* 알고리즘과 Modified Bi-directional A* 알고리즘의 경우를 보면 전체적으로 Dijkstra 알고리즘에 비해 성능 향상을 보였으며, 특히 직선 길 탐색에서는 Modified Bi-directional A* 알고리즘을 이용한 탐색이 가장 좋은 성능을 나타냈다.



<그림 11> 직선 길 탐색 소요 시간



<그림 12> 굽은 길 탐색 소요 시간

4. 결론 및 향후 연구

본 논문에서는 차량 항법 시스템에서 경로 탐색을 위한 다양한 경로 탐색 알고리즘들의 성능을 시뮬레이션을 통해 비교 측정하였다. 정확성과 빠른 탐색 시간 모두를 요구하는 차량 항법 시스템에서의 경로 탐색에서는 Bi-directional Dijkstra 알고리즘과 Modified Bi-directional A* 알고리즘을 사용하는 것이 적합한 것으로 평가되었다. 아울러 이러한 알고리즘들을 이용하여 좀 더 효율적으로 경로를 탐색하기 위해서는 Bi-directional Dijkstra 알고리즘은 탐색 반경을 줄이고, Modified Bi-directional A* 알고리즘에서는 h 계산의 부담을 줄일 수 있도록 최적화 되어야 한다.

지금까지 진행된 최단 경로 안내는 동적인 교통 상황에 대한 정보 없이 단지 도로의 거리 정보만을 이용하여 이루어졌다. 그러나 차량 항법 시스템이 좀 더 현실적인 최단 경로를 안내하기 위해서는 동적인 교통 상황 정보를 고려하여 목적지까지의 최단 경로를 선정하여야 하며 앞으로 이를 위한 연구가 체계적으로 진행될 것이다.

참고문헌

[1] Takahiro Ikeda, Min-Yao Hsu, Hiroshi Imai, Shigeki Nishimura, Takeo Hashimoto, Kenji Tenmoku, Kunihiko Mitoh, A Fast Algorithm For Finding Better Routes By AI Search Techniques. *Proc. Vehicle Navigation & Information Systems,* 1994.

[2] D. Champeaus, Bidirectional Search Again, J. *ACM 30.* 1983, pp.22-32.

[3] P. E. Hart, N. J. Nilson, and B. Rafael, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Trans. Sys. Sci. and Cyb. SSC-4,* 1968, pp.100-107

[4] T. Hiraga, Y. Koseki, Y. Kajitani, and A. Takahashi. An Improved Bidirectional Search Algorithm for the 2 Terminal Shortest Path. *The 6th Karuizawa Workshop on Circuits and Systems,* 1993, pp.249-254(in Japanese).

[5] M. Luby, and P. Ragde, A Bidirectional Shortest-Path Algorithm With Good Average-Case Behavior, *Proc. 12th International Colloquium on Automata Languages and Programming. LNCS 194,* 1985, pp.394-403.

[6] I .Pohl, Bi-Directional Search. *Machine Intelligence vol. 6,* pp.127-140, 1971.

[7] Y. Shirai and J. Tsuji, *Artificial Intelligence,* Iwanami Course: Information Science vol. 22, Iwanami, Tokyo. 1982(in Japanese).