최저 속력 동적 휴리스틱을 이용한 경로탐색†

Path Finding with Minimum Speed Dynamic Heuristic

문대진* / Dae-Jin Moon, **조대수**** / Dae-Soo Cho

요 약

이 논문에서는 단말기 기반 시스템에서 경로를 탐색할 때 탐색비용을 줄이고, 경로탐색의 질을 높이기 위해 동적 휴리스틱을 제안한다. 동적 휴리스틱이란 고정된 정보가 아닌 서버로부터 지속적으로 전송받는 휴리스틱으로, 교통정보를 계산한 데이터이다. 서버 기반의 경로탐색 서비스는 서버에서 경로를 탐색하여 결과를 클라이언트에 제공하지만, 제안하는 방법은 휴리스틱 정보만을 클라이언트에 전송하고 이를 활용하여 경로탐색을 하게 된다. 이 논문에서는 제안하는 동적 휴리스틱을 적용하기 위해 새로운 알고리즘을 제안한다. 제안하는 동적 휴리스틱은 지도를 그리드로 나누고, 각 그리드는 구역 내 간선들의 최저속도 정보를 가진다. 그리드의 최저속도가 기준치에 미달되면 해당 그리드를 제거하고 경로를 탐색한다.

Abstract

In this paper, we propose a Dynamic Heuristic to reduce the number of node accesses and improve quality of path in the client-based navigation service. The Dynamic Heuristic is to use heuristic data from server that is calculated with traffic data. The server-based navigation service provides a path searched on server and transmits it to client, but we propose that server only provide heuristic data to client. The proposed client searches a path with heuristic transmitted data from server. We present a new algorithm for using Dynamic Heuristic in the path-finding. The algorithm bases Grid Based Path-Finding, and has minimum speed data of edges in grid. It removes several grids whose minimum speed is less than limited speed.

주요어 : 동적 휴리스틱, 그리드 제거법, A* 알고리즘
Keyword : Dynamic Heuristic, Grid Removing, A* Algorithm

1. 서 론

복잡한 도로 네트워크상에서 임의의 (출발지, 목적지)에 대해서 최적의 경로를 탐색하기 위한 다양한 연구가 진행되어 왔으며, 현재 다양한 상용 서비스가 제공되고 있다. 일반적으로 경로 탐색의 주체에 따라 서버기반 경로탐색과 단말기기반 경로탐색으로 구분될 수 있으며, 검색결과에 따라 최적 경로탐색과 준최적 경로탐색으로 나뉠 수 있다.

서버기반 경로탐색방식은 단말기에서 (출발지,

† 이 논문은 2007년도 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2007-331-D00376)
■ 논문접수 : 2008.2.21 ■ 심사완료 : 2008.4.21
* 동서대학교 컴퓨터정보공학부 학사과정(wizardyk@nate.com)
** 교신저자 동서대학교 컴퓨터정보공학부 조교수(dscho@dongseo.ac.kr)

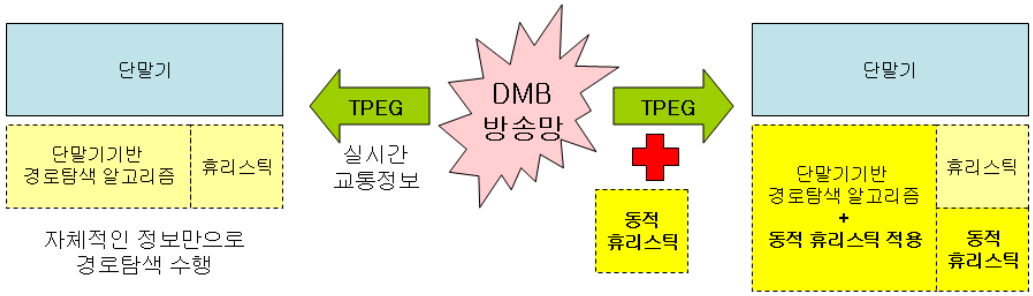
목적지)쌍을 서버로 전송하면 서버에서 그 경로를 탐색하여, 탐색된 경로를 다시 단말기로 전송하는 방식이다. 서버에서는 수집된 실시간 교통정보를 반영할 수 있으며, 고성능의 H/W를 이용하기 때문에, 최적 경로탐색 알고리즘이 사용된다. 단말기기반 경로탐색방식은 서버와의 통신 없이 단말기 자체적으로 경로를 탐색하는 방식이다. 상대적으로 저사양의 PDA 등의 단말기에서 자체적으로 경로탐색을 수행해야 하므로, 일반적으로 최적경로 탐색이 아닌, 준최적 경로탐색 알고리즘이 사용된다. 이 논문은 단말기 기반 경로탐색에서 준최적 경로를 탐색하는 알고리즘을 다룬다.

일반적으로 단말기 기반 경로탐색방식은 서버로부터 별도의 정보를 제공받지 못하였으나, 지상파 DMB용 TPEG(Transport Protocol Expert Group) 서비스를 통해 단말기가 경로탐색에 필요한 부가적인 정보를 받을 수 있는 채널이 생기게 되었다. 즉, 단말기에서도 TPEG 서비스를 통해 실시간 교통정보가 반영된 경로탐색이 가능하게 되었으며, 현재 일부 서비스에서 실시간교통정보가 반영된 단말기기반 경로탐색 서비스가 제공되고 있다(그림 1-(a)).

이 논문에서 제안하는 아이디어는 다음과 같다. TPEG 서비스 채널을 통해서 실시간 교통정보 기반의 동적 휴리스틱(dynamic heuristic)을 생성하여 단말기의 경로탐색에 활용한다면 경로탐색 비용도 줄이고 탐색결과의 정확도로 개선할 수 있다(그림 1-(b)). 즉, 일반적으로 단말기기반 경로탐색에서 사용하는 준최적 경로탐색 알고리즘에서는 고정된 휴리스틱을 사용하고 있으나, 이 논문에서는 실시간으로 변경되는 휴리스틱을 사용하는 경로탐색 알고리즘을 제안하고자 한다.

이 논문에서는 현재의 단말기 기반의 시스템에서 사용되는 유동적인 도로 데이터를 이용하여 기존의 A* 알고리즘[1]을 변형시킨 새로운 알고리즘 제안하고자 한다. 제안하는 알고리즘은 기존의 A* 알고리즘의 휴리스틱인 맨해튼 거리를 쓰지만, 기존의 방법과 달리 지도를 일정한 크기로 나눈 그리드를 가진다. 각 그리드는 자신의 범위내의 도로 가운데 속력이 최저인 수치를 정보로 가진다. 이 정보는 제안하는 알고리즘에서 경로를 탐색할 때 최저속력이 기준치에 미치지 못하는 그리드를 탐색 범위에서 제거하는 방법으로 쓰인다. 제안하는 알고리즘과 기존의 A* 알고리즘을 토대로 프로그램을 구현하고 실제와 비슷한 형태를 가지는 도로데이터를 이용하여 실험을 하였다. 실험결과 제안하는 알고리즘이 탐색한 경로의 이동시간이 기존의 A* 알고리즘 보다 성능이 향상되었다.

이 논문의 구성은 다음과 같다. 2장에서는 기존의 경로탐색 알고리즘에 대해 알아보고, 3장에서 제안하는 알고리즘에서 사용되는 동적휴리스틱에 대해 다룬다. 4장에서는 제안하는 알고리즘으로 실행한 실험 결과를 분석하고, 5장에서 결론 및 향후 과제에 대해 설명 하겠다.

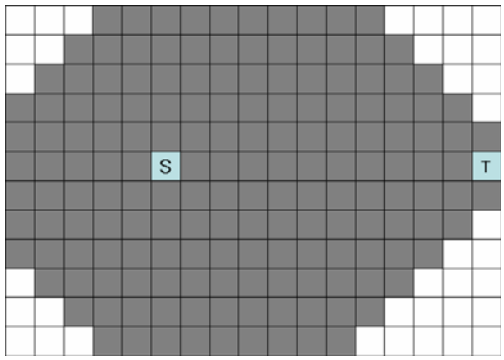


(a) 현재 단말기기반 경로탐색 모델 (b) 향후 단말기기반 경로탐색 모델

<그림 1> 동적휴리스틱을 적용한 단말기기반 경로탐색 모델

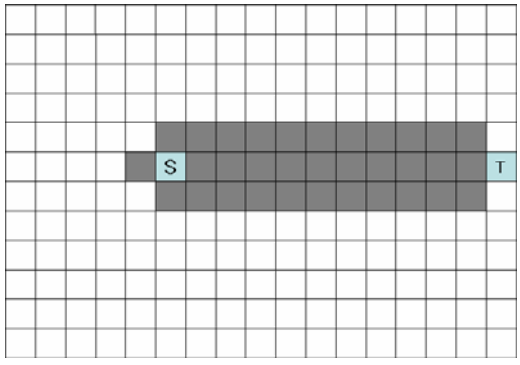
2. 관련연구

경로탐색 알고리즘은 검색결과에 따라 최적 경로탐색[2,3,4,5,6,7]과 준최적 경로탐색[8,9,10,11,12,13,14,15]으로 나뉠 수 있다. 최적해를 탐색하는 방식에서는 경로 뷰(Path View)[9,2,3,4]라고 불리는 미리 계산되어진 패스 정보를 활용함으로서 검색비용을 줄이려는 연구들이 많이 수행되었다. 즉, 다익스트라(Dijkstra) 알고리즘[7]과 같이 최적의 해를 찾는 알고리즘은 많이 있으나, 도로네트워크를 구성하는 노드의 수가 커짐에 따라 검색비용이 매우 커지기 때문에, 실제 응용에서는 사용할 수 없다. 따라서 검색비용(시간)을 줄이기 위해서 미리 계산되어진 정보가 필요하다. 플랫 패스 뷰(FPV: Flat Path View)[2,3]는 모든 노드 쌍에 대해서 미리 최적의 경로를 계산해 놓는 방식이다. 패스 뷰는 미리 계산되어진 패스라는 의미로 실체화된 패스 뷰(Materialized Path View)라는 용어에서 출발하였다. 패스 뷰를 사용하는 경우에 경로 탐색비용은 줄어들지만 FPV를 유지하는 비용이 매우 커지는 단점이 있다. 즉, 도로네트워크 정보에 변경이 발생하면, 패스 뷰를 재계산해야 하는 단점이 있다. 따라서 계층구조를 활용하여, 패스 뷰의 양을 줄이는 연구들[4,5,6]이 다양하게 수행되었다.



<그림 2> Dijkstra 알고리즘의 탐색 범위

준 최적해를 탐색하는 방법은 휴리스틱을 이용하고 있다. 즉, 경로 탐색에 힌트가 될 수 있는 지식 등을 휴리스틱으로 이용함으로써, 최적의 해를 보장하지는 않지만 빠른 시간에 경로를 탐색하는 방법으로서, A* 알고리즘이 가장 널리 활용되고 있다. A* 알고리즘은 탐색 비용을 줄이기 위해 휴리스틱 함수를 사용한다. 휴리스틱은 사전적으로 학습을 돕는다는 의미로 A* 알고리즘에서 목적지를 빨리 찾기 위해 사용된다. 휴리스틱은 일반적으로 맨해튼 거리를 사용하는데, 이는 목적지까지의 거리를 가로와 세로의 거리의 합으로 나타낸 것이다. 이 알고리즘으로 경로를 탐색할 때 주변의 이동 가능한 간선으로 탐색을 하는데 이때 현재까지 이동한 비용(G)과 휴리스틱 비용(H)의 합(F)이 낮은 곳을 우선 탐색하게 된다. G 값은 현재까지 이동하는데 드는 경로의 비용이지만 H 값은 목적지까지의 거리를 도로로 이동한 값이 아니라 가상의 거리로 목적지까지 이동할 때 드는 추정 비용이다. 따라서 H 값의 유무에 따라, 또는 비중에 따라 경로 탐색의 정확도와 탐색비용이 달라진다.



<그림 3> A* 알고리즘의 탐색 범위

H 값이 없거나 그 비중이 매우 낮을 경우 A* 알고리즘이라 하더라도 최적의 해는 찾을 수 있다. 그러나 탐색 범위가 넓고, 탐색 시간이 길어진다. H 값의 비용이 클 경우 해를 찾는 비용이 줄어든다. “overdo heuristic”을 사용하는 A* 알고리즘[6]은 탐색비용을 줄이기 위해 목적지까지의 추정비용인 H에 적절한 값을 곱하여 탐색비용은 감소 시켰다. 시작점과 목적지를 둘러싸는 영역의 노드 접근을 제한하는 방법[5]도 있었으나, 결과적으로 탐색

비용을 감소시키는 반면 경로의 질이 떨어졌다.

기존의 경로탐색 기법들은 지도상에서 미리 알 수 있는 정적인 데이터의 조합으로 경로를 탐색하는 문제점을 갖는다. 즉, 도로의 상황은 시시각각 변하기 때문에 기존의 알고리즘으로 탐색할 경우 최단경로를 찾을 수 있어도 이동시간도 최적인 경로를 찾을 수 없다. 따라서 동일한 경로를 탐색하더라도 현재 도로의 상황에 따라 다른 경로가 탐색될 수 있는 동적인 데이터를 기반으로 탐색하는 방법이 필요하다.

현재의 네비게이션 시스템은 TPEG을 통해 교통정보를 전송 받을 수 있다. 이를 통해 고정된 데이터가 아닌 동적인 데이터를 사용하여 경로를 탐색할 수 있다. 그러나 단말기 기반의 탐색의 경우 도로의 상황정보를 토대로 경로를 탐색 한다 해도 센터 기반 시스템 보다 하드웨어 자체 성능이 낮기 때문에 A* 알고리즘과 같은 휴리스틱을 사용하는 준 최적 경로 탐색을 한다. 또한, 단말기 자체에서 경로를 탐색하기 때문에 지도의 모든 도로의 상황정보를 전송받아 탐색할 경우 데이터를 전송받는 추가 연산시간이 생기기 때문에 경로탐색속도가 매우 느리다. 따라서 단말기 기반의 경로탐색 시스템에서 탐색비용을 낮추고, TPEG을 통한 도로 교통정보를 활용하여 경로의 질을 향상 시킬 수 있는 알고리즘이 필요하다.

3. 동적 휴리스틱을 적용한 최저 속력 제거 알고리즘

3.1 동적 휴리스틱

같은 시작점과 목적지를 가지는 경로라도 도로 교통 상황에 따라 최적경로가 변할 수 있다. 경로탐색으로 산출된 도로가 목적지까지의 가장 짧은 경로라도 그 도로가 정체 구간 이거나 신호를 받을 구간이 많으면 최적 경로가 되지 않을 수 있다. 이는 거리상으로 최단 경로가 이동시간까지 최적이 아님을 뜻한다. 최적 경로를 탐색하려면, 도로의 거리뿐만 아니라 도로의 상황, 즉 속력에 대한 정보를 고려하여 경로를 탐색하여야 한다. 이미 정해진 도로의 거리와 달리 속력에 관한 정보는 때에 따라 변한다. 따라서 최적 경로를 탐색하기 위해서 지속적으로 도로의 상황정보를 전송받아 이를 경로 탐색에 이용해야 한다.

과거와 달리 현재의 경로탐색 서비스는 도로상황을 고려한 탐색이 가능하다. 센터 기반의 경로탐색 서비스의 경우 여러 곳의 도로 데이터를 서버로 전송하고, 서버에서는 고성능의 서버 컴퓨터를 사용하여 최적 경로 탐색 기법을 통해 경로를 탐색한다. 반면 단말기 기반의 경로탐색 서비스는 서버로부터 도로 데이터를 전송 받고, 이를 이용하여 경로탐색에 이용하지만 센터 기반에 비해 상대적으로 하드웨어의 성능이 낮기 때문에 주로 준 최적 경로 탐색을 하게 된다.

일반적으로 준 최적 경로 탐색에서 휴리스틱은 목적지까지 이동하는데 드는 가상의 비용으로 거리에 관한 정보를 이용하여 비용을 추출한다. 거리를 이용하는 휴리스틱의 경우 동일한 거리에서 항상 같은 값으로 고정된 휴리스틱이 산출된다. 그러나 현재의 경로 탐색 시스템에서 휴리스틱을 고정 시킬 경우 최적 경로와 거리가 먼 경로를 탐색할 가능성이 높다. 단순히 거리에 관한 정보로는 현재 도로 상황에 따른 최적 경로를 탐색할 수 없기 때문이다. 따라서 도로 데이터를 이용한 준 최적 경로 탐색에서 때에 따라 같은 길이라도 휴리스틱이 변해야 한다.

이 논문에서는 고정되지 않고 경로 탐색 때마다 변하는 휴리스틱을 동적 휴리스틱이라 정의한다. A* 알고리즘에서 쓰이는 휴리스틱의 의미를 포함할 뿐만 아니라 경로 탐색에 영향을 주지만, 탐색 때마다 그 값이 변할 수 있는 모든 비용을 동적 휴리스틱이라 정의한다.

3.2 최저 속력 동적 휴리스틱

최근 단말기 기반의 경로탐색 시스템에서도 동적 휴리스틱을 이용한 탐색이 이루어지고 있다. 그러나 동적 휴리스틱 정보를 전송받기 위한 프로토콜

이 존재한다 하더라도 센터 기반의 경로탐색 기법과 달리 모든 도로의 속력정보를 갱신하여 탐색하기 힘들다. 단말기 기반 탐색은 상대적으로 성능이 떨어지는 하드웨어를 사용하기 때문에 모든 도로에 대한 동적 휴리스틱을 적용할 경우 경로탐색 비용 이외의 휴리스틱 데이터 전송 비용이 들기 때문이다. 따라서 단말기 기반의 경로탐색 기법에 새로운 알고리즘이 필요하다.

이 논문에서는 단말기 기반의 경로탐색 시스템에서 탐색 비용을 줄이고 경로의 질을 높이기 위해 새로운 알고리즘을 제안한다. 제안하는 알고리즘은 기본적으로 A* 알고리즘을 기반으로 하며, 휴리스틱은 맨해튼 거리를 사용한다. 단말기 기반의 특성상 모든 도로에 대한 속력 정보를 갱신하여 도로상황을 고려한 탐색을 할 경우 탐색시간외의 데이터 전송시간이 소비될 수 있다. 따라서 제안하는 알고리즘에서 동적 휴리스틱은 단말기의 성능을 고려하여 최소한의 정보만을 전송받아 경로탐색에 활용한다. 또한, 전송 받는 휴리스틱 데이터는 모든 도로의 속력 정보가 아닌 맵을 일정크기의 그리드로 나누고 그리드내의 도로들 중 최저 속력과 최고 속력에 관한 정보만을 전송 받는다. 이 논문에서는 최저 속력에 관한 정보만을 이용하여 경로탐색에 이용한다. 차후에 최고 속력 정보를 이용한 알고리즘을 개발하여 경로탐색시스템에 보완할 예정이지만 이 논문에서는 이에 대해 다루지 않는다.

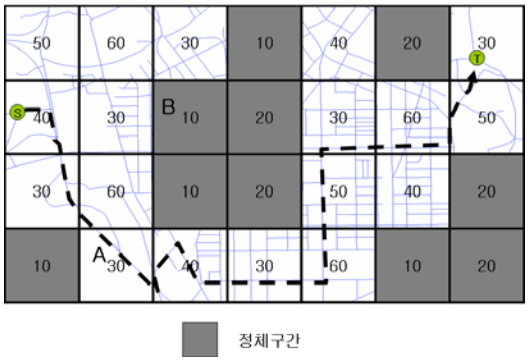
이 논문에서 제안하는 경로탐색 알고리즘은 <알고리즘 1>과 같으며, 구성은 다음과 같다.

<알고리즘 1> 최저 속력 동적 휴리스틱 A* 알고리즘

```
procedure DH_Min_Astar (G(V,E),s, d, grid, T)
{
   foreach u in V do
   {
      if (s,u) is edge then g(s,u) = edge(s,u)
      else g(s,u) = inf;
      g(s,s) = 0;
      path(s,u) = null
   }
   OpenListSet = [s];
   CloseListSet = emptySet;

   while not_empty(OpenListSet)
   {
      select w from OpenListSet where minimum(g(s,w)+h(w,d));
      OpenListSet = OpenListSet - [w];
      CloseListSet = CloseListSet + [w];

      if(w == d) then terminate;
      else
      {
         fetch(w.adjacencyList);
         foreach <u, g(w,u)> in w.adjacencyList
         if g(s,u) > g(s,w) + edge(w,u) and then
         {
            if( gird(u, w).speed < T ) then edge(w, u) = inf;

            g(s,u) = g(s,w) + edge(w, u);
            path(s,u) = path(s,w) + (w, u);
            OpenListSet = OpenListSet + [u];
         }
      }
   }
}
```

1) 목적지 입력
2) 서버로부터 동적 휴리스틱 정보 수신
3) A* 경로 탐색시작
   a) 탐색 중인 노드가 속하는 그리드의 최저 속력이 T 미만일 경우 검색에서 제외
   b) 최저 속력이 T 이상일 경우 맨해튼 거리를 휴리스틱으로 사용
4) 최종 경로 산출

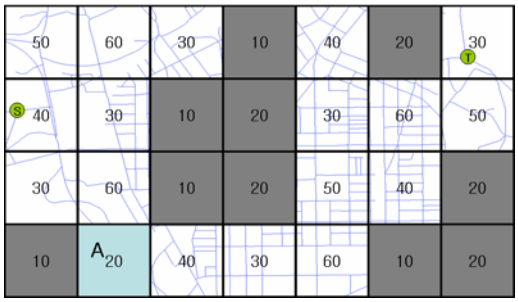
알고리즘의 구성에서 3)번의 A* 경로 탐색은 기존의 맨해튼 거리를 이용하는 탐색 방법과 동일하다. 그러나 a)에서와 같이 A* 알고리즘에서 노드를 방문했을 때 해당 노드가 속하는 그리드의 속력 정보를 확인하여 그리드의 최저 속력이 T 미만일 경우 해당 노드는 탐색에서 제외 된다. 그리드의 최저 속력이 낮은 구간은 도로의 정체구간을 뜻하며, 이는 실제 도로에서 정체되는 구간이 있을 경우 해당 도로로 길이 탐색되는 것을 애당초 방지하기 위해 사용한 것이다. 단, T는 이 논문에서 30km/h 로 정한다. 30km/h의 속력에 관한 기준은 평균 시내 도로의 최고 속도 제한이 60km/h 이상인 곳이 대부분이기 때문에 최고 속도의 1/2을 기준으로 한 임의 속력이다. 이 논문에서는 그리드 제거에 관한 내용만 다루고 그리드 제거 속력의 기준은 다루지 않는다.



<그림 4> 최저 속력 그리드 제거법을 이용한 경로탐색의 예

<그림 4>는 최저 속력 그리드 제거 기법을 이용한 경로탐색의 한 예이다. 그림에서 어두운 부분은 최저 속력이 T 이하인 그리드로 경로탐색에서 제외 된다. 제거되지 않은 그리드 상에서 목적지까지의 경로를 탐색하며 그림에서 점선으로 표시된 부분이 탐색되어 산출된 경로이다.

만약 그리드 제거를 하지 않은 상태에서 경로를 탐색한다면, 목적지 까지 최적 경로를 찾기 위해 우선적으로 시작점의 주변 노드부터 탐색을 하게 된다. 이 경우 속력이 낮은 구간이 시작점 주변에 있기 때문에 경로를 탐색을 할 때 A구역 보다 B구역을 우선 탐색할 경우가 높다. 그러나 B 구역의 속력이 낮기 때문에 B구역을 통해 목적지로 가는 것보다 A구역을 통해 목적지로 가는 최적경로를 늦게 탐색하게 된다. 논문에서 제안하는 알고리즘은 이와 같이 경로 탐색에 있어서 탐색범위를 줄이고 적은 정보로 더 효율이 좋은 경로를 찾는 것이 목적이다.



<그림 5> 그리드 제거법의 문제점

그러나 이 알고리즘에는 한 가지 단점이 있는데 이는 <그림 5>와 같이 그리드 제거로 인해 목적지를 찾을 수 없을 경우이다. 실제 도로상에서는 경로가 존재 하지만 그리드 제거로 인해 경로를 찾을 수 없는 경우가 생길 수 있다. 따라서 알고리즘 상에서 그리드의 제거는 물리적인 제거가 아닌 논리적인 제거로 이루어진다. 이 말은 경로 탐색에서 최저 속력이 낮은 그리드를 탐색에서 제외 하지만, 완전히 데이터를 없애는 것이 아니라 휴리스틱의 가중치를 매우 높게 주어 주변의 갈 수 있는 노드를 모두 탐색한 후에는 제거 되었던 구역의 노드도 탐색할 수 있게 하는 방법이다.

4. 구현 및 성능평가

4.1 알고리즘의 구현

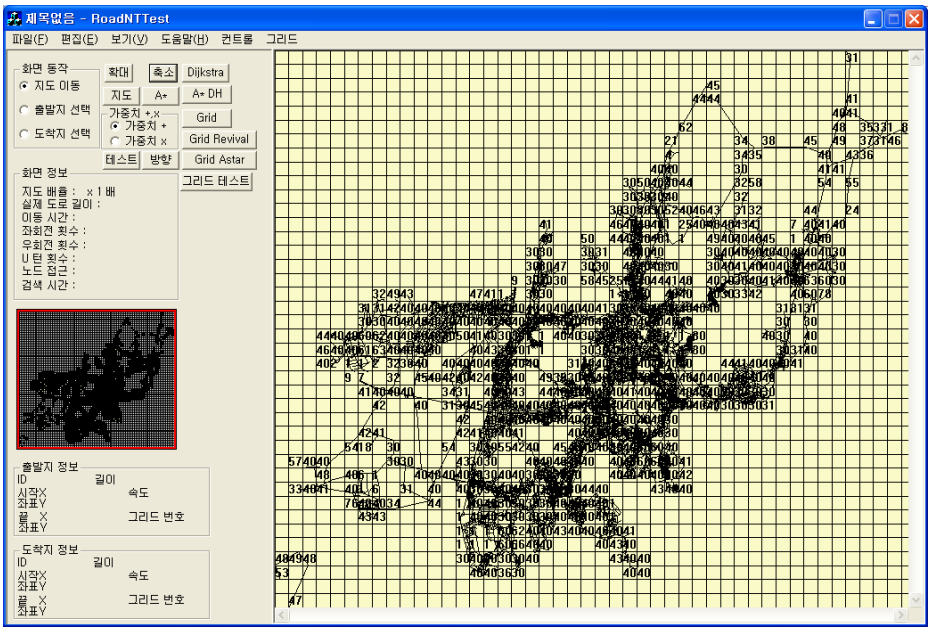
이 논문은 동적 휴리스틱을 적용한 경로탐색의 효율성에 대한 연구만을 목적으로 한다. 따라서 실제 상용화 되고 있는 TPEG을 이용한 경로탐색 시스템의 서버와 실제 도로데이터를 구현하지는 않았다. 알고리즘 구현의 범위는 클라이언트 프로그램으로 동적 휴리스틱은 서버로부터 전송 받는다는 가정 하에 매 탐색 때마다 다른 휴리스틱 값을 적용하였다.

<그림 6>은 제안하는 알고리즘의 성능을 평가하기 위해 프로그램을 구현한 모습이다. Windows XP 기반으로 VC++ 6.0에서 구현하였다. 그림에 보이는 지도는 부산시내 도로데이터를 기반으로 하는 데이터이며, 최적 속력 동적 휴리스틱을 적용하는 알고리즘을 위해 지도를 그리드로 나눈 것을 볼 수 있다. 각 그리드마다 구역 내의 간선들 중 최저 속력정보를 가지며 이를 지도에 표시하였다. 성능 비교의 모델로는 맨해튼 거리를 휴리스틱으로 사용하는 기본적인 A* 알고리즘과 도로의 속력을 고려하여 동적 휴리스틱을 적용한 A* 알고리즘, 그리고 논문에서 제안하는 최저 속력 동적 휴리스틱을 적용한 그리드 제거 알고리즘을 사용한다.

성능 비교를 위해 사용되는 데이터는 부산시내 도로데이터를 기반으로 하는 데이터로 두 점의 좌표와 길이를 정보로 가지는 간선이며, 하나의 간선이 하나의 도로를 나타낸다. 간선의 개수는 모두 106,254 개 이며 각 간선들은 주변의 간선들과 연결되어 있다. 간선의 최대 길이는 4,432m 이고, 최소 길이는 4m, 평균길이는 약 66m 이다.

4.2 성능비교대상 및 질의 셋

최저 속력 동적 휴리스틱을 사용하여 그리드를 제거하는 알고리즘의 성능을 알아보기 위해 성능실험을 하였다. 실험에 쓰인 알고리즘은 A* 알고리즘과 제안하는 알고리즘이다. 알고리즘을 구현 프로그램으로 성능을 분석하기 위해 <표 1>과 같은 질



<그림 6> 최저 속력 동적 휴리스틱 알고리즘의 실제 구현 모습

<표 1> 성능 비교 질의 셋

구 분	목적지까지의 직선거리	그리드 크기	그리드 제거 비율	질의 셋 개수
Q_1	0~5 km 5~10 km 10~15 km 15~20 km 20~30 km	0.5km	5%, 10%, 20%, 30%	4개
Q_2		0.1km 0.5km 1.0km	10%	3개

의 셋을 사용하였다.

제안하는 알고리즘은 그리드가 탐색에서 핵심적인 역할을 하기 때문에 그리드의 크기와 정체 구간으로 인한 그리드 제거율에 따른 질의 셋을 준비하였다. <표 1>과 같이 Q_1은 그리드의 크기를 0.5km로 고정한 채 그리드 제거 비율에 변화를 주었고, Q_2는 그리드 크기를 변화시키고 그리드 제거 비율은 10%로 고정하였다.

실험은 하나의 시작점과 목적지를 각각 A* 알고리즘과 제안하는 알고리즘으로 탐색을 하고, 목적지까지의 거리에 따라 각 질의 셋 별로 200회의 실험을 하였다. 탐색된 경로의 질을 평가하기 위해 경로의 길이를 사용하지 않고 각 도로의 속력정보를 주고, 산출된 경로의 [각 간선의 길이] / [각 간선의 속력]의 공식을 이용하여 이동시간을 평가하였다. 실제 도로에서 정체되는 큰 도로가 있으면, 이 도로로 진입하는 작은 도로들도 대부분 정체될 가능성이 높다. 이와 같은 상황을 고려하여 도로의 속력은 같은 그리드 안에서는 비슷한 범위 내에서 부여했다.

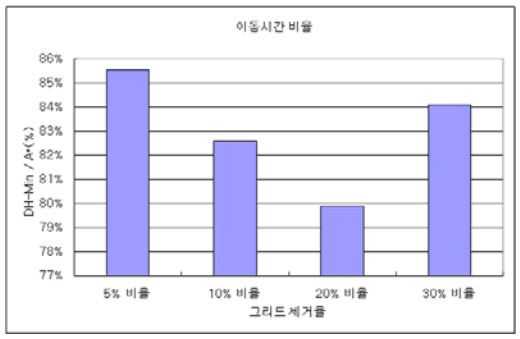
최대한 실제 도로와 비슷한 형태를 가지기 위해 <표 2>와 같이 도로의 속력을 부여하였다. 도로의 평균 속력 구간인 30~60km/h의 속력을 가지는 도로를 가장 많이 만들었다. 최저 속력 동적 휴리스틱을 적용한 그리드 제거 알고리즘에서 30km/h이하인 구역을 탐색에서 제거하기 때문에 전체 지도상에서 그리드 제거율에 따른 Q1의 실험에서는 1~30km/h구간과 30~60km/h구간의 비율을 조정하였다.

4.3 그리드 제거 비율에 따른 성능 비교

그리드 제거 비율은 실제 도로 상황 관점에서 보면 정체구간의 비율을 뜻한다. 그리드 제거율이 높으면 정체구간이 많고, 제거율이 낮으면 정체구간이 적다. 4.2절의 Q1을 이용하여 A* 알고리즘과 제안하는 알고리즘의 성능을 비교해 보았다.

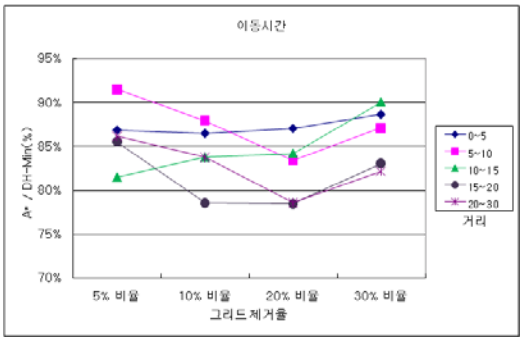
<표 2> Q_2에서의 도로 속력 분포

속 력	비 율	세부 속력	비 율
1~30km/h	10%	1~10km/h	10%
		10~20km/h	20%
		20~30km/h	70%
30~60km/h	80%	30~40km/h	25%
		40~50km/h	37%
		50~60km/h	37%
60~100km/h	10%	-	-



<그림 7> 그리드 제거율에 따른 경로의 이동 시간

<그림 7>은 그리드 제거율에 따른 이동시간을 [최저속력동적휴리스틱(DH-Min)] / [A* 알고리즘(A*)] 으로 표현한 그래프이다. 이동 시간은 각 알고리즘으로 탐색된 경로의 각 도로의 길이/속력을 합산한 것이다. 이동 시간이 짧을수록 최적 경로에 근접한다고 할 수 있겠다. <그림 7>에 보이는 바와 같이 최저 속력 동적 휴리스틱을 사용하는 알고리즘이 A* 알고리즘보다 이동시간의 비율이 낮음을 보인다. 특히, 그리드 제거율이 낮을 때보다 제거율이 높을 때 경로의 질이 향상되었다. 실제 도로와 접목시키면 제안하는 알고리즘은 소통이 원활한 구간이 많은 도로보다 약간의 교통 체증이 발생할 때 효율성이 높을 가능성이 있다.



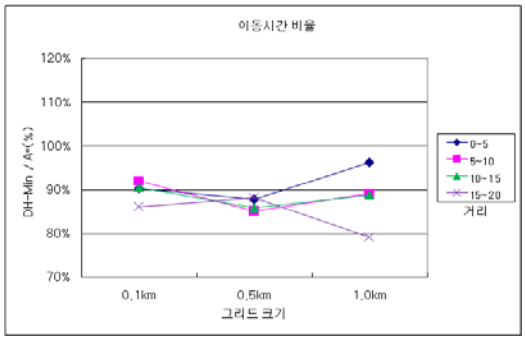
(단, 그래프에서 거리는 산출된 경로의 거리가 아닌 시작점과 목적지의 직선거리이다.)

<그림 8> 경로길이 세분화 이동시간 그래프

<그림 8>은 그리드 제거율에 따른 이동시간의 그래프를 각 경로의 거리에 따라 세분화 시킨 모습이다. 0~5km 구간의 짧은 경로를 탐색할 경우 그리드 제거율은 큰 영향을 미치지 못하는 것을 볼 수 있다. 15~20km 구간과 20~30km 구간의 그래프를 유추해보면 대체로 탐색하고자 하는 경로의 길이가 길 경우 제안하는 알고리즘이 더욱 최적에 가까운 경로를 찾을 수 있음을 알 수 있다.

4.4 그리드 크기에 따른 성능 비교

그리드의 크기에 따라 알고리즘의 성능이 변할 수 있다. 그리드의 크기가 너무 작으면 전송 받아야 할 동적 휴리스틱 데이터가 많아져 전송비용이 커지기 때문에 전체 탐색비용이 증가할 수 있다. 반대로, 그리드의 크기가 너무 크면 경로탐색에서 한 번에 제거되는 도로가 많아지기 때문에 최적경로에 크게 미치지 못하는 경로를 찾을 가능성이 높다. 이에 대한 성능을 실험하기 위해 4.2절의 Q_2를 이용하여 A* 알고리즘과 제안하는 알고리즘의 성능을 비교해 보았다.



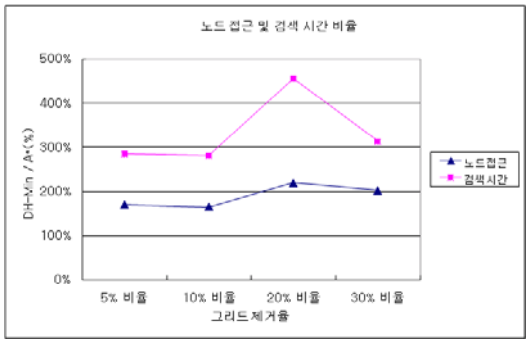
<그림 9> 그리드 크기에 따른 경로의 이동 시간

<그림 9>는 그리드 크기에 따른 이동시간을 경로의 거리에 따라 [DH-Min]/[A*]로 나타낸 그래프 이다. 그래프에서 나타는 것처럼 0.5km 크기의 그리드에서 가장 좋은 성능을 보이고 있다. 가장 큰 그리드 크기인 1.0km의 경우 탐색 거리에 따라 큰 편차를 보이고 있다. 반면, 그리드 크기가 작은 0.1km와 0.5km의 경우 대체로 탐색 거리와 상관

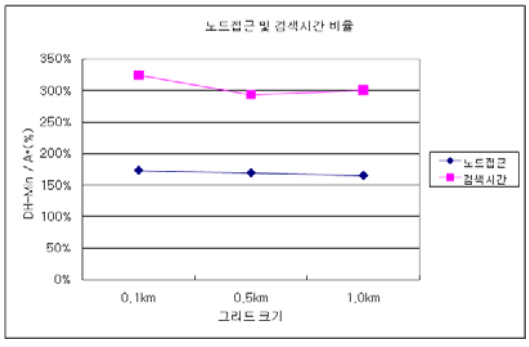
없이 일정한 성능을 보이고 있다.

4.5 노드접근 및 검색시간

좋은 경로를 찾을 수 있더라도 그 경로를 찾는데 비용이 많이 든다면 실시간 네비게이션 시스템에서 사용하기 어렵다. 4.2절의 질의 셋으로 경로를 탐색할 때마다 탐색하는데 드는 비용을 측정하기위해 노드접근 및 검색시간을 비교해 보았다.



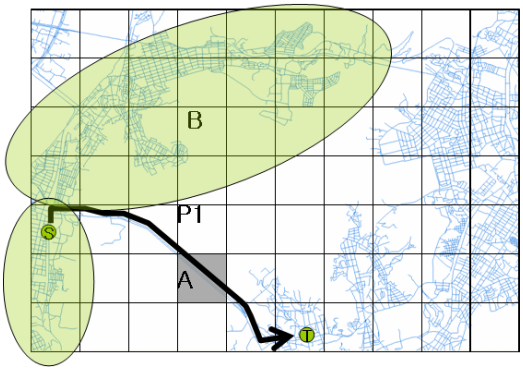
(a) 그리드 제거율에 따른 탐색비용



(b) 그리드 크기에 따른 탐색 비용

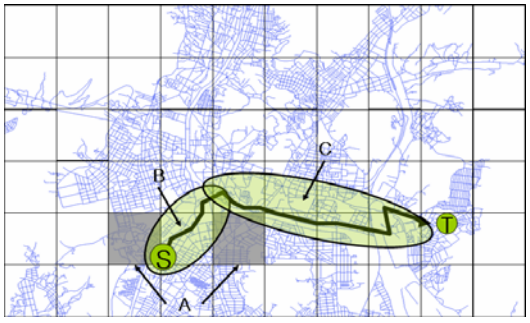
<그림 10> 경로 탐색에 따른 노드접근 및 검색시간

<그림 10>은 앞선 실험에서 측정된 경로탐색에서의 노드접근 및 검색시간을 A* 알고리즘에 대한 비율로 나타낸 그래프이다. 탐색비용이 줄어들 것이라는 예상과 달리 오히려 크게 증가한 것을 볼 수 있다. 이는 특정 경로에서 홀로 떨어진 유일의 도로의 그리드를 제거함으로 발생한다.



<그림 11> 노드접근 및 검색시간의 증가 원인

<그림 11>과 같이 목적지까지 경로를 탐색할 때 P1의 경로가 최적이라 가정할 때 A구역의 그리드가 최저 속력 동적 휴리스틱에 의해 제거될 경우 B구역을 탐색하게 된다. B구역을 탐색하면 목적지에 접근하기 위해 더욱 넓은 범위를 탐색해야 하기 때문에 결국 A구역을 경유하는 경로를 탐색하게 되지만, 기존의 경로탐색 방법으로 목적지를 찾으면 B구역을 탐색하지 않고 보다 빨리 P1의 경로를 찾을 수 있다.



<그림 12> 탐색 비용 최적의 예

만약 경로를 탐색할 때 <그림 12>와 같은 경우 탐색비용이 늘어나지 않는다. 왜냐하면 목적지까지 경로를 탐색하기 위해선 우선적으로 B구역을 탐색해야 하는데 이때 A구역의 그리드가 최저 속력 동적 휴리스틱에 의해 탐색에서 제외될 경우 탐색해야 하는 구역이 줄어들기 때문이다. 즉, 최적경로상에서 그리드제거가 없으면 탐색비용을 증가하지

<표 3> 동적 휴리스틱을 사용하는 알고리즘과의 성능비교 데이터 셋
(단, M(a, b)는 a노드와 b노드간의 맨해튼 거리이다.)

비교대상	A* 알고리즘	최저 속력 그리드 제거	A* 고정 휴리스틱 Weight 30	A* 고정 휴리스틱 Weight 60
휴리스틱	맨해튼 거리	맨해튼 거리 및 그리드 제거	맨해튼 거리 / 30 (이동시간)	맨해튼 거리 / 60 (이동시간)
우선탐색 기준평가 공식 (F)	E[i].Length + M(E[i], D)	if (E[i].GridMinSpeed >= 30) than E[i].Length + M(E[i],D)	(E[i].Length / E[i].Speed) + M(E[i], D) / 30	(E[i].Length / E[i].Speed) + M(E[i], D) / 60
그리드 사용	X	O	X	X
실시간 도로 데이터 사용 여부	X	O	O	O
질의	그리드 크기 : 0.5km 그리드 제거율 : 5%			

않을 가능성이 높다. C 구역에서 제거되는 그리드만 없으면 제안하는 알고리즘은 경로의 질뿐만 아니라 탐색비용도 줄일 가능성이 높다.

4.6 도로데이터 사용여부에 따른 성능 비교

앞선 실험에서 논문에서 제안한 알고리즘의 성능을 분석하기 위해 A* 알고리즘을 사용하여 비교하였다. 그러나 현재의 경로탐색 시스템은 실시간 도로 데이터를 사용하는 경우가 많다. 기존의 A* 알고리즘의 경우 도로 교통 상황정보를 사용하지 않기 때문에 실제 경로탐색에서 데이터 전송비용이 소모되지 않는다. 단순히 경로를 찾는 시간이 길어지더라도 전체 탐색비용이 더 낮다고 단정할 수 없다. 따라서 실제 도로 데이터를 전송받아 사용하는 알고리즘과의 성능을 비교해야 한다.

<표 3>은 실시간 도로 데이터를 사용하는 알고리즘과의 성능비교를 위한 데이터 셋이다. A* 알고리즘은 앞선 실험과 동일한 알고리즘이고, 최저 속력 그리드 제거 알고리즘은 이 논문에서 제안하는 알고리즘이다. 그리드 크기 및 제거율은 각각 0.5km와 5%로 고정하였다. A* 고정휴리스틱 Weight 30 과 Weight 60 알고리즘은 실시간 도로데이터인 각 도로의 속력정보를 이용한 A* 알고리즘으로 출발지(S)에서 목적지(D)까지의 경로탐색을 위한 노드 방문시 다음과 같은 수식을 이용한다.

$$F : G + H \tag{1}$$

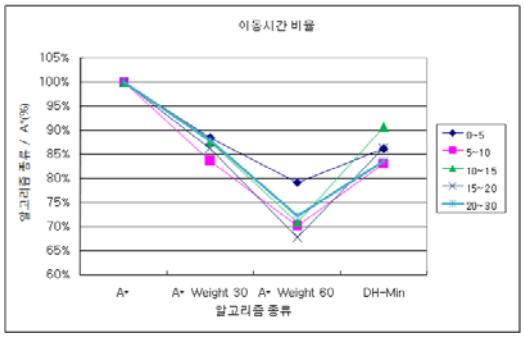
$$G : \sum_{i=s}^{c} (E[i].Length / E[i].Speed) \tag{2}$$

(단, $\sum_{i=s}^{c}$는 현재 방문 중인 노드까지의 탐색된 노드들의 최소 비용을 가지는 간선들의 집합이다.)

$$H : M(c, D) / (30 \text{ or } 60) \tag{3}$$

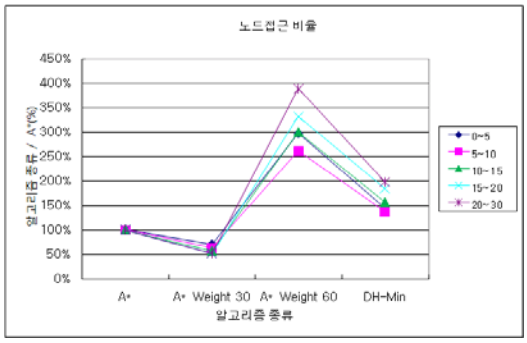
수식(3)에서 30과 60은 각각 Weight 30과 Weight 60 알고리즘에서 적용되는 속력을 뜻하며, H 값을 산출할 때 목적지까지의 거리를 30km/h 또는 60km/h의 속력으로 이동했을 경우 소비되는 시간이다. 30의 기준은 그리드 제거 속력 기준이 30km/h를 바탕으로 유추해낸 것이며, 60은 그 두 배이다. 두 가지의 알고리즘으로 성능비교를 하는 이유는 A* 알고리즘에서 휴리스틱의 비중이 경로의 질과 탐색 비용에 영향을 주기 때문이다. 목적지까지의 거리가 아닌 이동시간을 휴리스틱으로 계산했을 때 속력이 낮으면 휴리스틱의 비중이 커지므로 노드 접근수가 낮아 빠른 검색이 가능하지만 경로의 질이 떨어진다. 반대의 경우 노드접근수가 높

고 경로의 질이 향상된다.



<그림 13> 각 알고리즘에 따른 경로의 이동시간

<그림 13>은 알고리즘의 종류에 따른 경로의 이동시간을 그래프로 나타낸 것이다. 도로 상황 정보를 이용하면서 휴리스틱의 비중이 가장 낮은 A* Weight 60의 경우가 경로의 질이 가장 우수한 것을 볼 수 있다. 논문에서 제안하는 DH-Min을 사용하는 알고리즘은 A* Weight 30과 유사한 수준의 경로를 탐색해내고 있다.



<그림 14> 각 알고리즘에 따른 노드접근 비율

<그림 14>는 알고리즘에 따른 노드 접근 비율을 나타낸 그래프이다. A* Weight 30의 경우가 가장 낮은 노드접근 비율을 보이고 있다. 반면 가장 최적의 경로를 탐색하는 A* Weight 60의 경우 매우 높은 노드 접근 비율을 보인다. DH-Min의 경우 두 알고리즘의 중간 수준의 노드접근을 보인다.

위의 그래프를 유추해 볼 때 도로 상황 정보를 이용하더라도 경로의 질을 높이기 위해서는 탐색비용이 커짐을 알 수 있다. 이 논문에서 제안하는 알고리즘인 DH-Min 휴리스틱을 사용하는 그리드 제거법의 경우 도로 상황 정보를 이용하는 알고리즘인 A* Weight 30의 경우와 비슷한 수준의 경로를 탐색하지만 탐색비용이 약 2배가량 높음을 알 수 있다.

그러나 이 실험은 단순히 경로를 탐색하는 비용만을 계산한 것이다. 즉, 실시간 도로 상황 정보를 전송받는 비용은 제외된 것이므로, 항상 DH-Min의 방법이 경로의 전체 탐색 비용이 높다고 할 수 없다. 왜냐하면, A* Weight 30과 A* Weight 60의 경우 경로를 탐색하기 위해서는 지도상의 모든 도로의 속력 정보를 갱신해야 하기 때문이다. 실험에서 사용된 도로의 개수는 약 10만개 가량 되지만, 그리드의 개수는 크기를 0.5km 기준으로 약 7천여 개지만 실제 도로가 지나가는 그리드는 전체 그리드의 약 40%인 3천여 개에 불과하다. 즉, 전송받아야 할 데이터의 단순 수치만 97%가 차이가 난다. 따라서 제안하는 알고리즘의 경우 그리드의 정보만을 전송받으면 경로탐색이 이루어 질 수 있기 때문에 전체 탐색비용은 오히려 제안하는 알고리즘이 낮을 가능성이 높다.

5. 결론 및 향후 과제

이 논문에서는 최저 속력 동적 휴리스틱을 사용하여 그리드 제거법을 이용한 경로탐색 기법에 관해 다루었다. 동적 휴리스틱을 사용하지 않는 A* 알고리즘과의 성능평가에서 경로의 질이 최대 22% 가량 향상됨을 보였다. 특히 도로가 원활한 소통이 이루어질 때 보다 정체구간이 있을 때 탐색된 경로의 질이 향상되었다. 반면 탐색비용은 증가하여 노드접근 및 검색시간에서 비효율적임이 보였다. 이는 그리드 제거를 통해 경로의 질은 높였지만 특정한 형태의 맵에서는 탐색비용이 매우 증가하였기 때문이다.

실시간 도로 데이터를 사용하는 알고리즘과의 성능평가에서는 휴리스틱의 비중이 다소 높은 알고리

즘과 비슷한 성능의 경로를 탐색함을 보였으나, 노드접근 및 탐색시간이 경로의 질에 비해 높은 수준으로 측정되었다. 그러나 실험에서 도로 상황 정보를 전송하는 비용이 포함되지 않았기 때문에 전송하는 데이터의 양이 적은 DH-Min을 사용하는 알고리즘이 전체 탐색 비용이 낮을 가능성이 있음을 보였다.

이 논문에서 제안된 방법은 동적 휴리스틱을 전송받았을 경우를 가정한 상태에서 제안하는 알고리즘이다. 만약 서버와의 통신이 원활하고 추가 비용이 들지 않는다면 기존의 서버 기반의 경로탐색 시스템이 더 효율적일 수 있다. 그러나 제안하는 방법은 서버와의 통신으로 데이터를 주고받는 것이 아닌 TPEG과 같은 방송용 채널을 통해 서버에서 도로 데이터를 전송하기만 하는 환경을 전제로 한다. 이 경우 추가적인 통신비용이 들지 않으며 사용자는 송신채널만을 가지면 된다.

향후 최저속력뿐만 아니라 최고속력을 고려한 동적 휴리스틱 개발이 추가로 요구된다. 또한, 서버와의 데이터 전송비용을 포함한 정확한 실험이 요구되며, 노드접근 및 검색시간을 줄이는 기법에 대한 연구가 필요하다.

참고문헌

1. Hart, P. E., Nilsson, N. J. and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths in Graphs," IEEE Trans. on Systems Science and Cybernetics, Vol. SSC-4, No. 2, 1968, pp. 100-107.
2. Agrawal, R., and Jagadish, H. V., "Materialization and Incremental Update of Path Information," Proc. Fifth Int'l Conf. Data Eng., 1989., pp. 374-383.
3. Huang, Y.-W., Jing, N., and Rundensteiner, E. A., "A Semi-Materialized View Approach for Route Guidance in Intelligent Vehicle Highway Systems," Pro. Second ACM Workshop Geographic Information System, 1994, pp. 144-151.
4. Huang, Y.-W., Jing, N., and Rundensteiner, E. A., "Hierarchical Encoded Path Views for Path Query Processing: An Optimal Model and Its Performance Evaluation," IEEE Trans. Knowledge and Data Eng., Vol. 10, No. 3, 1998, pp. 409-432.
5. Goldman, R., Shivakumar, N., Venkatasubramanian, S., and Garcia-Molina, H., "Proximity Search in Databases," VLDB, 1998
6. Jung, S., and Pramanik, S., "An Efficient Path Computation Model for Hierarchically Structured Topographical Road Maps," IEEE Trans. Knowledge and Data Eng., Vol. 14, No. 5, 2002, pp. 1029-1046.
7. Dijkstra, E. W., "A note on two problems in connection with graphs," Numerische Mathematik, Vol. 1, 1959, pp. 260-271.
8. Yang, T. A., Shekhar, S., Hamidzadeh, B., and Hancock, P. A., "Path planning and evaluation in IVHS databases," VNIS, 1991, pp. 283-290.
9. Huang, Y.-W., Jing, N., and Rundensteiner, E. A., "Hierarchical Path Views: A Model Based on Fragmentation and Transportation Road Types," Pro. Third ACM Workshop Geographic Information System, 1995, pp. 93-100.
10. Jagadeesh, G. R., Srikanthan, T., and Quek, K. H., "Heuristic Techniques for Accelerating Hierarchical Routing on Road Networks," IEEE Trns. Intelligent Transportation Systems, Vol. 3, No 4, 2002, pp. 301-309.
11. Karimi, H. A., "Real-time optimal route computation: a heuristic approach," ITS

J., Vol. 3, No. 2, 1996, pp. 111-127.
12. Jacob, R., Marathe, M. V., and Nigel, K., "A Computational study of routing algorithms for realistic transportation networks," the Second Workshop on Algorithmic Engineering, NJ, 1998
13. Huang, Y., and Jing, N., "Evaluation of hierarchical path finding techniques for ITS route guidance," Proc. Annu. Meeting of ITS America, Vol. 1, 1996, pp. 340-350.
14. Botea, A., MÜller, M., and Schaeffer, J., "Near Optimal Hierarchical Path-Finding," Journal of Game Development, Volume 1, Issue 1, 2005, pp 7-28.
15. Lanctot, M., Sun, N. N. M., and Verbrugge, C., "Path-finding for Large Scale Multiplayer Computer Games", GameOn-NA 2006, September 2006, pp. 26-33.

조대수
1995년 부산대학교 컴퓨터공학과 졸업(공학사)
1997년 부산대학교 컴퓨터공학과 졸업(공학석사)
2001년 부산대학교 컴퓨터공학과 졸업(공학박사)
2001년~2004년 한국전자통신연구원 텔레매틱스연구단 선임연구원
2004년~현재 동서대학교 컴퓨터정보공학부 조교수
관심분야 : GIS, 공간데이터베이스, LBS, 스트림 데이터처리 등
e-mail : dscho@dongseo.ac.kr

문대진
2002~현재 동서대학교 컴퓨터정보공학부 학사과정
관심분야 : 텔레매틱스, 비동기 통신, 크로스 플랫폼
e-mail : wizardyk@nate.com