실시간 그래프 탐색 알고리즘을 이용한 공간 탐사

최은미° 김인철
경기대학교 정보과학부 전자계산학과
{sogmi°, kic}@kyonggi.ac.kr

Space Exploration Using Real-time Graph Search Algorithms

Eunmi Choi° Incheol Kim
Department of Computer Science, Kyonggi University

요 약

본 논문에서는 자율 에이전트에 의해 미지의 공간을 탐사하는 실시간 그래프 탐색 알고리즘 DFS-RTA*와 DFS-PHA*를 제안하고 그 효율성을 비교한다. 두 알고리즘들은 모두 깊이-우선 탐색(DFS)을 기초로 하고 있으며, 직전 노드로의 빠른 후진(backtrack) 을 위해 각각 실시간 최단 경로 탐색 방법인 RTA*와 PHA*를 적용하는 것이 특징이다. 본 논문에서는 대표적인 3차원 온라인 게임 환경인 Unreal Tournament 게임과 지능형 캐릭터 에이전트인 KGBot를 이용한 실험을 통해 두 탐색 알고리즘의 완전성과 효율성을 분석해본다.

1. 서 론

3차원 온라인 게임에서 활동하는 캐릭터 에이전트들이나 실세계 공간을 탐사하는 로봇에게 가장 중요한 문제 중의 하나는 어떻게 하면 주어진 미지의 공간을 효과적으로 돌아다니며 이동 가능한 경로들을 파악하느냐 하는 공간 탐사 문제(space exploration problem)이다. 전통적으로 이 문제는 그래프 기반의 공간 표현법과 그래프 탐색 알고리즘들로 해결하려고 노력해 왔다. 따라서 많은 효율적인 공간 탐사를 위한 그래프 탐색 알고리즘들이 제안되어 왔으나, 이들은 대부분 양방향으로 이동 가능한 무향 그래프(unidirectional graph)를 가정하거나 실제 특정 노드를 방문하지 않고도 그 노드에 이웃한 노드들을 사전에 모두 알 수 있다고 가정하고 있다. 공간 탐사를 위한 알고리즘에 요구되는 대표적인 성질은 탐색의 완전성과 효율성이다. 본 논문에서는 미지의 유향 공간 그래프를 빠짐없이 모두 방문하는 완전성과 중복 방문 노드들의 수를 최소화할 수 있는 실시간 탐색 알고리즘 DFS-RTA* 알고리즘과 DFS-PHA* 일고리즘을 제안한다.

2. 실시간 최단 경로탐색

Dijkstra알고리즘이나 A*와 같은 전통적인 최단 경로 탐색 알고리즘들은 대부분 이미 전체 그래프를 다 알고 있거나 적어도 실제 특정 노드를 방문하지 않고도 그 노드에 이웃한 노드들을 사전에 모두 알 수 있다고 가정하고 있다. 하지만 실세계 환경에서는 에이전트가 직접 이동하여 특정 지역을 방문하지 않고는 주변 지역과의 연결관계를 미리 알 수 없는 경우가 대부분이다. 최근 들어 에이전트가 직접 실세계 공간에 놓여 이동과 경로탐색을 펼치는 실시간 경로 탐색 알고리즘들이 활발히 연구되고 있다.

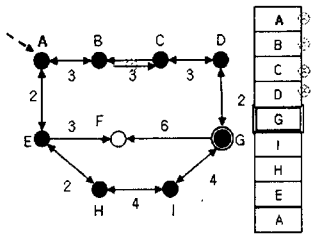
2.1 RTA* 알고리즘

RTA*(Real-Time A*) 알고리즘은 대표적인 실시간 경로 탐색 알고리즘으로서, 이웃한 노드 중에서 현재 노드로부터 목표 노드까지의 추정거리가 가장 짧은 노드를 선택하여 이동하는 과정을 반복한다. RTA* 알고리즘에서는 각 이웃 노드 y에 대한 평가함수 f(y)를 다음과 같이 계산한다. f(y) = k(x, y) + h(y). 즉 현재 노드 x에서 노드 y까지의 거리 k(x, y)에 노드 y에서 목표 노드까지의 추정 거리 h(y)를 더한 값을 평가치로 삼아 이러한 평가치가 가장 낮은 이웃 노드를 다음 노드로 선택한다. 이 알고리즘은 매 단계 현재 노드의 이웃한 노드만을 대상으로 국지적인 관점(local view)에서 다음 노드를 선택하게 되어 출발 노드부터 목표 노드까지의 최단 경로를 보장 하지는 못하나, 빠른 시간 내에 최선 경로(optimal path) 혹은 차선 경로(suboptimal path)를 찾아낼 수 있는 효율적인 탐색 알고리즘 이다[4].

2.2 PHA* 알고리즘

또다른 실시간 최단 경로 탐색 알고리즘인 PHA*은 A* 알고리즘처럼 그래프 전체를 대상으로 하는 OPEN-LIST 중에서 평가치 f(x)=g(x)+h(x) 값이 우수한 노드를 다음 방문 노드로 선택하는 방식을 취한다. 그래프 전체 노드를 대상으로 다음 방문할 노드를 결정하기 때문에 다음 방문할 노드가 현재 위치에서 멀리 떨어지는 경우도 발생하며, 현재 위치에서 이곳까지 실제 이동해가기 위해서는 이미 방문한

적이 있는 노드들을 중복 방문하기도 한다. 하지만 A* 알고리즘과 같이 언제나 최적의 경로 탐색을 보장한다. PHA* 알고리즘은 두 단계로 구성되며, 윗 단계에서는 A* 알고리즘과 동일하게 전역적 관점에서 평가치 f(x)가 가장 낮은 노드를 다음 방문 노드로 선택하는 일을 담당하고, 아랫 단계에서는 선택된 노드까지 실제로 이동해가는 일을 담당한다[2].



3. 공간 탐사 알고리즘

3.1 SimpleDFS 알고리즘

SimpleDFS 알고리즘은 무정보 그래프 탐색(uninformed graph search) 알고리즘인 DFS(Depth First Search)를 미지의 공간 탐사를 위해 적용한 하나의 공간 탐사 알고리즘이다. 원래 DFS 그래프 탐색은 탐색에 특별한 휴우리스틱(heuristic)을 이용할 수 없기 때문에 목표 노드를 만날 때까지는 체계적으로 모든 노드들을 다 방문해보는 일종의 전역 탐색(exhaustive search)의 성질을 가진다. 하지만 잘 알려진 바대로 DFS는 BFS(Breadth First Search)와 같은 다른 전역 탐색 알고리즘들에 비해 탐색 중에 기억하고 있어야 하는 노드 수가 적고 중복 방문 노드의 수도 적어 매우 효율적이다. 따라서 이러한 특징을 지닌 DFS를 전체 맵(map)이 알려져 있지 않은 미지의 공간 탐사에 적용하면 효과를 얻을 수 있다[3]. SimpleDFS는 기본적으로 DFS에 따라 탐색을 전개하되, 현재 노드에서 임의의 이웃한 비방문 노드를 다음 노드로 선택하지 않고, 현재 노드로부터 가장 가까운 근거리에 있는 이웃 노드를 우선 선택한다. 또 SimpleDFS는 실세계 공간의 특성을 반영하여 지형상 현재 노드에서 직전 노드로 바로 되돌아 갈 수 없는 경우, 다른 경로로 우회해서라도 최단 경로를 따라 직전 노드로 되돌아 가는 탐색을 펼친다는 것이 기본 DFS와 다른 점이다. SimpleDFS의 탐색과정 중 만약 방문하지 않은 이웃 노드가 존재하지 않은 경우에는 현재 노드에서 더 이상 탐색을 계속할 수 없다. 우리는 이와 같은 상황을 "STUCK"이라 부르고, STUCK이 발생하면 방문 기록을 보관하고 있는 스택(stack)으로부터 되돌아갈 노드를 찾아낸다. 되돌아갈 노드는 아직 비방문 이웃 노드(unvisited neighbor node)를 포함하고 있는 가장 최근 방문 노드가 된다. 일단 되돌아갈 노드가 선택되었으면 그 노드까지 가장 짧은 경로를 통해 이동한다. 후진이 성공하였으면 그 노드에서부터 SimpleDFS 탐색은 계속된다. 만약 스택의 모든 노드를 꺼내도 비방문 이웃 노드를 가진 노드를 발견할 수 없다면, 이때는 이미 그래프 상의 모든 노드를 다 방문한 것으로 판단하고 알고리즘을 종료한다.

```
Function : simpleDFS(S_NODE)
C_NODE = S_NODE    /* start node */
N_NODE, Y_NODE = null
do
   while(exhausted(C_NODE) != true)
      do
        N_NODE = selectNextNode(C_NODE)
        advanceTo(N_NODE)
        push( HISTORY, C_NODE ) /* history stack */
        C_NODE = N_NODE
      end
   do
      Y_NODE = pop(HISTORY)
   until((exhausted(Y_NODE) != true) or
         (empty(HISTORY) = true))
   if (exhausted(Y_NODE) != true),
     moveShortestPath(C_NODE, Y_NODE)
     C_NODE = Y_NODE
until (empty(HISTORY) = true)

Function : exhausted(C_NODE )
for each neighbor Y_NODE of C_NODE
   if unvisited(Y_NODE), return false
return true

Function : selectNextNode(C_NODE)
f = 10000    /* a large value */
for each neighbor Y_NODE of C_NODE
      if distance(C_NODE, Y_NODE) <  f,
        B_NODE = Y_NODE
        f = distance(C_NODE, Y_NODE)
return B_NODE
```

[그림 2] simpleDFS 알고리즘

[그림 1]은 특히 simpleDFS 탐색 중 STUCK이 발생한 상황을 나타내고 있다. A노드를 시작으로 하여 A-E-H-I-G-D-C-B 를 거쳐 A노드에 도착 하였을 때 STUCK이 발생하게 된다. 이때 스택에 저장된 노드를 하나씩 꺼내어 주변 노드에 방문하지 않은 노드의 유무를 확인한다. A, B, C, D 노드는 이웃한 노드가 모두 탐색 된 노드이기 때문에 고려하지 않는다. 그 다음 노드인 G 노드의 이웃한 노드가 탐색 되지 않았기 때문에 G를 다음 방문할 노드로 선택한다. STUCK이 발생한 B 노드를 시작으로 하여 G 노드로 최적 경로 탐색 알고리즘을 이용하여 이동 한다. [그림 2]는 이와 같은 SimpleDFS 알고리즘을 요약한 것이다.

3.2 DFS-RTA*와 DFS-PHA* 알고리즘

DFS-RTA*와 DFS-PHA* 알고리즘은 각각 앞서 소개한 공간 탐사를 위한 SimpleDFS를 기초로 후진노드(bactrack node)까지 이동을 위해 실시간 최단 경로 탐색 알고리즘인 RTA*와 PHA*를 적용한 공간 탐사 알고리즘들이다. [그림 3]과 [그림 4]는 각각 RTA*와 PHA*를 나타내는 함수들로서, [그림 2]의 SimpleDFS 알고리즘에서 호출하는 함수 moveShorestPath(C_NODE, Y_NODE)를 대신할 수 있다.

```
Function : RTA(S_NODE, G_NODE)
C_NODE = S_NODE ; best = 0
do
   for each neighbor Y_NODE of C_NODE
       f = h(Y_NODE,G_NODE) + k(C_NODE, Y_NODE)
       if f < best, B_NODE = Y_NODE ; best = f
   advanceTo(B_NODE)
   C_NODE = B_NODE
until  C_NODE = G_NODE
```

[그림 3] RTA* 알고리즘

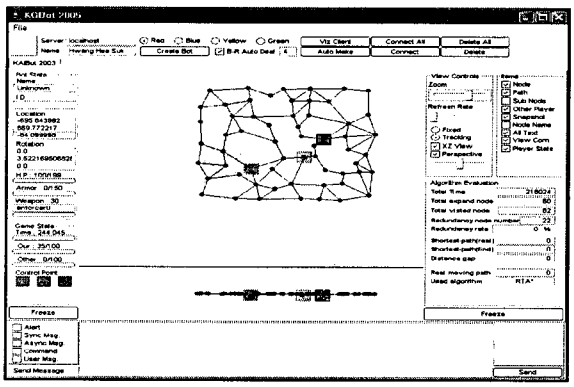
```
Function : PHA(S_NODE, G_NODE)
C_NODE = S_NODE ; best = 0
do
  N_NODE = best node from OPEN_LIST
  if N_NODE = explored
     lowerLevel(C_NODE , N_NODE)
     C_NODE = N_NODE
until  C_NODE = G_NODE


Function : lowerLevel(C_NODE, G_NODE)
S_NODE = C_NODE
f = 0
do
   for each neighbor Y_NODE of C_NODE
     f = g(Y_NODE) + h(Y_NODE,G_NODE)
       if f < best, B_NODE = Y_NODE ; best = f
   moveTo(B_NODE)
   C_NODE = B_NODE
until  C_NODE =  G_NODE
```

[그림 4] PHA* 알고리즘

4. 구현 및 실험

본 논문에서는 대표적인 3차원 온라인 게임 환경인 Unreal Tournament 게임[1]과 지능형 캐릭터 에이전트인 KGBot를 이용하여 두 공간 탐사 알고리즘들의 완전성과 효율성을 알아보기 위한 실험을 전개하였다. [그림 5]는 실험에 사용된 KGBot의 실행화면을 보여주고 있다.



[그림 5] 공간 탐사 중인 KGBot의 실행화면

실험에 사용된 맵은 6개이며, 각각 노드 수가 60, 90, 120, 150, 180, 210 으로 구성 되어 있다. 실험은 각각의 맵마다 임의 선택된 서로 다른 5개의 시작 노드에서 시작하여 실험 하였고, 총 소요시간, 방문노드 수, 이동거리에 대하여 두 알고리즘을 비교하였다. 실험결과를 나타내는 [표1]을 살펴보면, 두 알고리즘 모두 주어진 맵상의 모든 노드를 빠짐없이 다 방문하였으나, 탐색 효율성 면에서 약간의 차이를 보였다. 탐색에 소요된 방문 노드수, 총 소요시간, 총 이동거리에서 평균적으로 DFS-PHA* 알고리즘이 DFS-RTA* 알고리즘보다 우수한 결과를 얻을 수 있었다. 특히 후진할 노드까지 최단 경로 탐색에서 RTA* 알고리즘에 비해 PHA* 알고리즘은 언제나 최적의 경로를 찾아 낼 수 있었다. 이것은 RTA*에 비해 더 많은 계산량을 요구하는 PHA* 알고리즘의 단점을 상쇄하는 장점으로 파악된다.

[표 1] 실험 결과

노드수	시작노드	총 소요시간		총 방문 노드수		총 이동거리	
		DFS-PHA	DFS-RTA	DFS-PHA	DFS-RTA	DFS-PHA	DFS-RTA
60	PathNode95	221,108	220,767	91	86	36,059	36,425
	PathNode83	212,486	220,767	83	82	34,080	34,868
	PathNode126	214,619	213,637	79	91	34,426	34,352
	PathNode110	213,076	213,627	79	83	33,190	34,103
	PathNode124	218,093	226,225	87	96	34,821	37,997
90	PathNode23	273,824	284,398	116	118	36,481	37,133
	PathNode148	276,367	276,167	117	116	36,571	37,111
	PathNode160	286,661	286,983	120	120	37,589	37,589
	PathNode201	272,672	272,671	114	116	35,726	36,266
	PathNode176	278,981	285,591	122	122	38,277	38,074
120	PathNode23	415,287	429,087	191	197	63,911	69,269
	PathNode50	401,968	408,429	166	172	60,693	62,560
	PathNode100	421,196	422,027	191	197	66,044	66,523
	PathNode56	400,356	411,221	176	177	61,385	63,460
	PathNode88	431,530	422,427	191	195	68,492	68,324
150	PathNode23	581,188	598,637	211	231	99,556	107,240
	PathNode113	598,620	595,777	219	263	104,368	123,989
	PathNode2	617,168	642,564	235	249	109,332	114,889
	PathNode64	671,522	630,377	206	242	98,424	114,188
	PathNode80	598,881	637,126	218	244	104,235	116,531
180	PathNode23	753,183	779,641	271	291	135,863	143,701
	PathNode64	776,537	797,472	274	292	141,387	143,288
	PathNode80	715,067	726,625	256	260	129,352	130,131
	PathNode113	779,170	816,433	280	311	143,983	154,600
	PathNode120	711,372	715,929	255	259	128,584	128,969
평균		149,413	153,008	53	56	27,164	28,028

5. 결 론

본 논문에서는 자율 에이전트에 의해 미지의 공간을 탐사하는 실시간 그래프 탐색 알고리즘 DFS-RTA*와 DFS-PHA*를 제안하고, 3차원 온라인 게임 환경을 이용하여 그 효율성을 비교, 분석하였다.

참고문헌

[1] R. Adobbati et al., "Gamebots : A 3D virtual world test-bed for multi-agent research", Proceedings of Agents-01, 2001.

[2] Makoto Yokoo and Yasuhiko Kitamura, "Multiagent Real-time A* with Selection: Introducing Competition in Cooperative Search." Proceedings of ICMAS-96, p409-416, 1996.

[3] Stephe Kwek, "On a Simple Depth-First Search Strategy for Exploring Unknown Graphs", LNCS. 1272, p 345-353, 1997.

[4] A. Felner, R. Stern, A. Ben-Yair, S. Kraus and N. Netanyahu, "PHA*: Finding the Shortest Path with A* in an Unknown Physical Environment", JAIR, Vol.21, p631-670, 2004.