멀티 에이전트를 이용한 협력적 공간 탐사

최은미[O] 김인철
경기대학교 정보과학부 전자계산학과
{sogmi[O], kic}@kyonggi.ac.kr

Collaborative Exploration Using Multi Agents

Eunmi Choi[O] Incheol Kim
Department of Computer Science, Kyonggi University

요 약

본 논문에서는 자율적인 다수의 에이전트들이 협력하여 미지의 공간을 탐사하는 실시간 그래프 탐색 알고리즘인 MADFS를 제안하고 그 효율성을 분석한다. 이 알고리즘은 깊이-우선 탐색(DFS)에 기초한 단일 에이전트 공간 탐사 알고리즘인 DFS-RTA* 와 DFS-PHA* 를 멀티 에이전트 환경에 적합하게 에이전트 간 정보공유 방식과 방문노드 선택전략을 설계하여 확장하였다. 본 논문에서는 대표적인 3차원 온라인 게임 환경인 Unreal Tournament 게임과 지능형 캐릭터 에이전트인 KGBot를 이용한 실험을 통해 알고리즘의 완전성과 효율성을 분석해본다.

1. 서 론

3차원 게임 환경에서 활동하는 캐릭터 에이전트들이나 실세계 공간에서 활동하는 로봇에게 가장 중요한 문제 중의 하나는 어떻게 하면 주어진 미지의 공간을 효과적으로 돌아다니며 이동 경로를 파악하느냐 하는 공간 탐사 문제(space exploration problem)이다. 이 문제는 공간 표현법에 따라 접근법이 크게 달라지는데 인공지능 분야에서는 전통적으로 그래프 기반의 공간 표현법과 그래프 탐색 알고리즘들로 이 문제를 해결하려고 노력해 왔다. 공간 탐사를 위한 알고리즘에 요구되는 대표적인 성질은 탐색의 완전성(completeness)과 효율성(efficiency)이다.

기존의 공간 탐사 알고리즘들은 주로 단일 에이전트에 기초한 방법들이었다. 그러나 일반적으로 멀티 에이전트를 이용하면 각 에이전트가 서로 다른 영역을 탐사함으로써 단일 에이전트의 경우보다 좀 더 빠른 시간 안에 전체 공간을 탐사할 수 있다. 또한 각 에이전트가 습득하는 불완전한 정보를 통합함으로써 좀 더 정확하고 완전한 환경 정보를 얻을 수 있다. 하지만 멀티 에이전트의 경우, 동일한 공간 안에서 움직이는 에이전트 서로 간의 충돌과 간섭효과, 그리고 탐사 영역의 중복성 문제 등이 발생할 수 있으므로 효율적인 에이전트 간 조정(coordination)이 매우 중요하다.

본 논문에서는 하나의 유향그래프(directed graph)로 표현되는 미지의 공간을 다수의 에이전트들이 협력하여 효율적으로 탐사하기 위한 공간 탐사 알고리즘을 제안한다. 이 알고리즘은 탐사에 참여하는 각 에이전트의 자율성을 최대한 보장하면서도 중복 탐사를 최소화함으로써 탐색의 효율성을 높일 수 있다는 점이 특징이다.

2. 관련연구

Reid Simmons[1]는 멀티 에이전트를 이용한 효율적인 협력적 공간 탐사를 위해 이미 탐사된 공간 영역과 미 탐사 영역 사이의 경계 셀(frontier cell)들에 대해 각 에이전트별로 각자의 정보습득량(information gain)을 계산하고 이를 기초로 입찰(bid)하는 경매방식을 통해 각 에이전트에게 적절한 경계지역을 탐사대상으로 할당하는 방법을 제안하였다. Robert Zlot[3]의 연구 역시 협력적 탐사를 펼치는 멀티 에이전트의 조정을 위해 시장 경제원리(market economy)를 적용하였다. 즉, 이 연구에서는 각 에이전트마다 각자 D* 탐색 알고리즘을 이용하여 이동 가능한 셀들을 찾아내고 이 셀들에 대한 자신의 이익(Profit)을 계산한 다음 이것을 기초로 입찰(bid)하도록 설계하였다. Yasuhiko Kitamura[5]는 공간탐사(exploration)를 위한 앞선 두 연구와는 달리 특정 목적지까지 최단 경로 탐색(shortest-path finding)을 위해 멀티 에이전트를 이용하는 방법을 제시하였다. 이 연구에서는 각자 실시간 최단 경로 탐색 알고리즘인 RTA* 나 LRTA*에 따라 탐색을 펼치는 에이전트들의 노력을 효과적인 결합하기 위해서는 문제의 유형에 따라 때로는 다른 에이전트로부터 멀어지도록 유도하는 repulsion 방법과, 때로는 가까워지도록 유도하는 attraction방법이 필요하다는 것을 설명하고 있다.

3. 공간 탐사를 위한 그래프 탐색

3.1 공간 탐사 문제

유향 그래프(directed graph) G=(N, E)를 이용한 미지의 공간 탐사 문제는 다음과 같이 정의 할 수 있다. 한 에이전트는 현재 노드에서 출력 에지(outgoing edge)들 중 하나를 선택하고 이것을 방문하는 과정을 반복함으로써, 하나의 강 연결 유향 그래프(strongly connected directed graph)로 표현된 미지의 공간을 탐사하여야 한다. 이때, 에이전트는 자신이 한번이라도 방문한 적이 있는 모든 에지와 노드들은 기억할 수 있으며, 따라서 그들을 재방문할 경우에는 재방문 사실을 스스로 인식할 수 있다. 하지만 에이전트는 그래프상에 얼마나 많은 수의 노드들과 에지들이 존재하는지 미리 알지 못하고, 또 아직 방문하지 않은 각 에지들이 어떤 노드들과 연결되는지 알지 못한다. 에이전트가 그래프상의 모든 에지들과 모든 노드들을 방문하였을 때, 우리는 이 에이전트가 그래프로 표현된 미지의 공간을 모두 탐사하였다고 말한다. 에이전트의 목표는 가능한 최소의 에지와 노드들을 방문함으로써 주어진 그래프 공간을 모두 탐사하는 것이다.

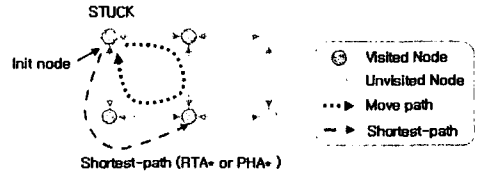
3.2 SimpleDFS 알고리즘

Kwek[4]등에 의해 발표된 SimpleDFS는 깊이- 우선 탐색(Depth-First Search, DFS)을 공간 탐사에 적용한 알고리즘으로서, 최대 $\min(mn, dn^2+m)$개의 - 여기서 m은 그래프상의 총 에지의 수를, n은 총 노드의 수를 가리킴 - 에지만을 방문하는 매우 효율적인 공간 탐사 알고리즘이다. SimpleDFS는 기본적으로 깊이-우선탐색에 따라 탐색을 전개하되, 현재 노드와 이웃한 비방문 노드들(unvisited neighbor nodes) 중 하나를 선택하여 이동한다. 탐색과정 중 만약 방문하지 않은 이웃 노드가 존재하지 않은 경우에는 현재 노드에서 더 이상 탐색을 계속할 수 없다. 우리는 이와 같은 상황을 "고착상태(stuck)"라 부르고, 고착상태가 발생하면 방문 기록을 보관하고 있는 스택(stack)으로부터 되돌아갈 노드를 찾아낸다. 이때 되돌아갈 노드는 비방문 이웃 노드를 적어도 하나 포함하고 있는 가장 최근 방문 노드

* 본 논문은 산업자원부 지원으로 수행하는 21세기 프론티어연구개발사업(인간기능 생활지원 지능로봇 기술개발사업)의 일환으로 수행되었습니다.

가 된다. 일단 되돌아 갈 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 더 이상 발견할 수 없으면, 이미 그래프 상의 모든 노드를 다 방문한 것으로 판단하고 알고리즘을 종료한다.

3.3 DFS-RTA* 와 DFS-PHA* 알고리즘

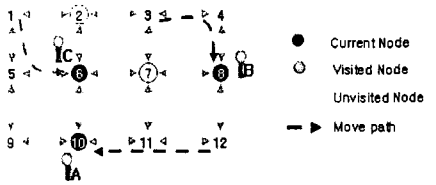


[그림 1] DFS-RTA* 와 DFS-PHA*의 예

DFS-RTA*와 DFS-PHA*[6]는 SimpleDFS에 기초한 단일 에이전트 공간 탐사 알고리즘들이다. 이들의 특징은 매번 다음 방문 노드를 결정할 때 비방문 이웃노드들 중 현재 노드에 가장 가까운 노드를 우선으로 선택하고, 또 후진노드(bactrack node)까지의 빠른 이동을 위해 실시간 최단 경로 탐색 알고리즘들인 RTA*와 PHA*를 각각 적용한다는 것이다.

4. 협력적 공간 탐사

[그림 2]는 3명의 에이전트들을 이용하여 동일한 미지의 공간을 탐사하는 한 예를 나타내고 있다. 그림에서 에이전트 A, B, C는 각기 독립적으로 공간탐사 알고리즘인 DFS-RTA*를 적용함으로써 현재 10번 노드, 8번 노드, 6번 노드를 각각 방문 중에 있다고 가정하며, 비방문 이웃노드들 중의 하나를 다음 방문 노드로 선택하여 탐사를 계속해야 한다. 이때 에이전트들간에 서로 어떤 정보를 교환하고 공유함으로써 서로 불필요한 중복 방문과 충돌을 줄이면서도 완전한 탐사를 이루어낼 수 있는지가 중요한 문제이다. 또한 각각의 에이전트는 이러한 공유정보를 바탕으로 다음 방문할 노드를 어떻게 선택해야 하는지 또한 해결해야 할 중요한 문제이다.



[그림 2] 협력적 공간 탐사의 예

4.1 에이전트간 정보 공유

다수의 에이전트들을 이용하여 미지의 공간을 탐사할 때 일반적으로 각각의 에이전트들은 서로 떨어져 독립적으로 동작하며, 많은 경우 그들간의 통신환경 또한 제한적이다. 따라서 각 에이전트가 매순간 감지하는 모든 환경정보나 로컬 맵(local map) 전부를 통신을 통해 실시간으로 교환하는 방식 등은 사실상 불가능하다고 가정한다. 하지만 반대로 너무 축약된 정보만을 간헐적으로 교환하는 경우, 이러한 정보를 바탕으로 실시간 제한성을 만족하는 효과적인 에이전트간 조정은 불가능하다. 따라서 어느정도 일정한 간격으로 에이전트간 조정에 꼭 필요한 적당 양의 정보를 에이전트간에 주고받는 정보교환 방식이 바람직하다.

협력적 탐사를 위한 에이전트간 조정에 꼭 필요한 필수 정보로는 매순간 각 에이전트의 현재 위치 정보를 들 수 있다. 이 정보를 기초로 각 에이전트는 자신과 다른 에이전트들이 이미 방문했던 노드에 대한 불필요한 재방문을 줄일 수 있고, 각자 탐사해야 할 영역을 동적으로 균등하게 분할할 수 있다. 에이전트간에 교환되는 각자의 위치 정보 메시지는 다음과 같이 정의 하였다.

MYPO agent_name, node_id, x, y, z, time_stamp

MYPO는 위치 정보임을 알려주고, agent_id는 에이전트의 이름을, node_id와 x, y, z은 현재 노드의 식별자와 절대 위치 좌표값을, time_stamp는 메시지 발송시각을 각각 나타낸다. 각 에이전트는 일정한 간격으로 교환되는 이러한 다른 에이전트의 위치정보를 지속적으로 기록함으로써, 자신의 방문 히스토리(history) 뿐만 아니라 다른 에이전트들의 대략적인 탐색경로도 자체적으로 파악하고 유지한다. 이와 같이 매순간 다른 에이전트의 위치정보를 추적하여 구성하는 탐색경로 정보를 자취 맵(trace map, T_MAP)이라고 부른다. 이러한 다른 에이전트에 대한 자취 맵은 실제로 해당 에이전트가 DFS-RTA* 공간탐사를 계속하기 위해 스스로 유지하는 자신의 로컬 맵(local map, L_MAP)과는 다르다. 따라서 본 연구에서는 협력적 공간 탐사에 참여하는 모든 에이전트는 각각 내부에 다음과 같은 서로 다른 두 종류의 맵을 유지 관리하는 것으로 가정한다.

$$\begin{cases} L\_MAP : \text{각 에이전트가 방문을 통하여 획득한 맵} \\ T\_MAP : \text{다른 에이전트의 자취를 기록한 맵} \end{cases}$$

[그림2]의 예에서는 에이전트 A, B, C간의 위치 정보 교환을 통해 다음과 같이 각 에이전트의 T_MAP을 구성하고 있는 것으로 가정한다.

$$T\_MAP_A = \{1,3,4,5,6,8\}$$
$$T\_MAP_B = \{1,5,6,10,11,12\}$$
$$T\_MAP_C = \{3,4,8,10,11,12\}$$

참고로, 본 연구에서는 그래프에 존재하는 각 노드는 서로 식별 가능하다고 가정하기 때문에, 일반적으로 로봇 탐사에서 상대적인 정보를 이용하여 다른 에이전트와 맵을 공유하고자 할 때 발생하는 맵 통합(map merge)문제와 각자의 현재 위치 파악(localization)에는 큰 어려움이 없다.

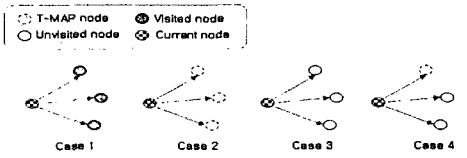
4.2 방문노드 선택전략

각 에이전트가 매순간 다음 방문 노드를 결정하는 방문노드 선택전략은 다음과 같은 몇 가지 사항을 고려하여야 한다. 첫째는 어떤 에이전트에 의해서도 탐사되지 않는 미탐사 공간영역이 남아서는 않된다. 둘째는 공간탐사에 참여하는 에이전트들간에 불필요한 중복 방문을 최소로 줄여야 한다. 셋째는 탐사에 참여하는 각 에이전트에게 탐사해야 할 공간 분할이 가능한 균등하게 이루어져야 한다. 마지막으로, 가장 중요한 사항은 탐사에 참여하는 각 에이전트의 자율성을 최대한 보장해야 한다. 본 연구에서는 협력적 공간탐사에 참여하는 모든 에이전트는 각자 독립적으로 DFS-RTA* (혹은 DFS-PHA*)에 따라 공간탐사를 수행하는 것으로 가정하였다. 그리고 이러한 가정하에 본 연구에서는 각 에이전트의 현재 위치를 기준으로 상대거리에 따라 각 이웃노드에 대한 효용성(utility)을 다음과 같이 계산한다.

$$utility_i(x') = 1/\frac{d(x', x_i)}{\sum_{j=1}^{N} d(x', x_j)}$$

즉, 에이전트 i가 현재 노드 x에서 다음 방문 할 후보노드인 이웃노드 x'에 대한 효용성(utility)을 계산할 때, 현재 자신을 포함해 다른 모든 에이전트 j들로부터 노드x'에 이르는 상대거리의 총합을 자신으로부터의 상대거리로 나눈 결과값을 그 노드의 효용성으로 본다. 따라서 자신의 현재 위치에서 방문대상 노드까지 거리가 가까울수록 그 노드에 대한 효용성은 크게 평가된다.

매순간 각 에이전트는 자신의 로컬 맵(L_MAP)과 다른 에이전트의 자취 맵(T_MAP)을 기초로, 자신이 현재 처한 상황을 파악하고 [표 1]과 같은 전략에 따라 다음에 방문할 노드를 선택한다. [그림 3]은 서로 다른 4 가지 유형의 탐색상황들을 예시하고 있다.



[그림 3] 4 가지 유형의 탐색상황들

	탐색 상황	선택 노드
Case 1	이웃한 비방문 노드가 없다.	고착상태로 인식, DFS-RTA*에 따라 후진노드 선택
Case 2	이웃한 비방문 노드가 모두 T-MAP에 속한다.	고착상태로 인식, DFS-RTA*에 따라 후진노드 선택
Case 3	비방문 노드 모두가 T-MAP에 속해 있지 않다.	효용성(utility)을 기준으로 다음 노드 선택
Case 4	비방문 노드 일부가 T_MAP에 속해 있다.	T_MAP노드들을 제외한 비방문 노드들의 효용성(utility)을 기준으로 다음 노드 선택

[표 1] 방문노드 선택전략

이와 같은 선택 전략을 이용하면 다른 에이전트의 탐사 영역인 T_MAP를 재방문하지 않을 수 있으며, 가능한 비방문 노드중 효용성(utility)이 높은 노드를 선택하여 이동할 수 있게 된다. 예컨대, [그림 2] 예의 경우 에이전트 A는 4번째 유형의 상황에 놓인 것으로 판단되며 따라서 T_MAP에 속한 비방문 노드를 제외한 2번,7번 중에서 효용성(utility)이 높은 노드를 선택하게 된다. 다른 두 에이전트 B, C 또한 4번째 유형의 상황에 놓여 효용성이 높은 노드를 선택하게 된다.

4.3 MADFS 알고리즘

[그림 4]의 MADFS(Multi-Agent DFS)알고리즘은 DFS-RTA*와 DFS-PHA* 알고리즘들을 앞서 설명한 방식대로 멀티 에이전트 환경에 적합하게 확장한 협력적 공간탐사 알고리즘이다.

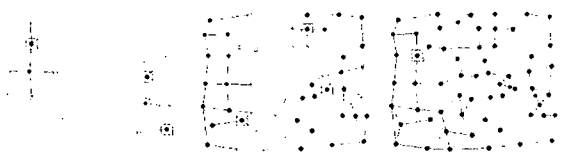
```
T_MAP = null /* trace map */
Function : exhausted(C_NODE)
for each neighbor Y_NODE of C_NODE
   if unvisited(Y_NODE) and ! contain(Y_NODE, T_MAP)
        return false
return true

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

[그림 4] MADFS 알고리즘

MADFS 알고리즘에 따르면, 각 에이전트는 자신의 로컬 탐색 알고리즘인 DFS-RTA*와 DFS-PHA*의 완전성에 힘입어 다른 에이전트가 방문하지 않은 탐사 영역에 대해서는 자동적으로 자신의 탐사 영역으로 인식함으로써 공간내의 모든 영역을 빠짐 없이 탐사 할 수 있다.

5. 구현 및 실험

본 논문에서는 대표적인 3차원 온라인 게임 환경인 Unreal Tournament(UT) 게임과 지능형 캐릭터 에이전트인 KGBot을 이용하여 제안한 멀티에이전트 공간 탐사 알고리즘인 MADFS 알고리즘의 완전성을 실험하였고, 멀티에이전트 탐사가 효과적인지를 비교하기 위하여 단일 에이전트의 경우와 에이전트간 조정이 없는 멀티 에이전트의 경우와 비교 실험하여 효율성을 알아보았다. [그림 5]는 특정 UT게임맵으로 주어지는 미지의 공간에서 3명의 KGBot 캐릭터 에이전트들이 협력하여 공간을 탐사하고 스스로 공간에 대한 맵을 완성해가는 모습을 보여준다.



[그림 5] 3명의 에이전트를 이용한 협력적 공간탐사

No	Total Nodes	Single agent exploration (DFS-RTA*)			Multi-Agent exploration (3 agents)					
					non-coordinated			coordinated		
		Consumed Time	Visited nodes	Moved distance	Consumed Time	Visited nodes	Moved distance	Consumed Time	Visited nodes	Moved distance
1	90	284,999	118	37,133	294,109	347	113,501	105,343	120	36,997
2	90	276,157	119	37,111	281,828	280	83,296	151,809	136	42,489
3	90	286,963	120	37,589	239,594	275	88,521	116,841	130	41,409
4	90	272,071	116	36,266	257,703	293	98,028	104,989	117	35,838
5	90	285,581	122	38,074	295,140	346	367,077	132,219	116	36,377
average		281,039	119	37,235	272,475	304	149,645	122,150	124	38,822
1	120	429,067	197	69,269	516,656	456	153,826	178,312	182	64,897
2	120	406,428	172	62,550	395,235	373	130,617	199,281	160	62,676
3	120	422,027	187	68,523	379,500	373	130,513	150,125	171	59,512
4	120	411,221	177	63,460	349,344	363	133,011	174,408	186	65,125
5	120	423,427	195	68,324	349,016	402	140,117	234,937	177	62,520
average		418,634	186	66,025	397,550	394	137,617	187,412	179	62,946
1	150	509,637	231	107,240	519,187	440	199,311	387,875	232	109,146
2	150	865,777	263	123,989	570,031	481	219,183	214,969	212	95,689
3	150	642,554	249	114,609	587,500	542	253,447	255,975	229	104,046
4	150	630,377	242	114,188	541,375	527	240,163	275,578	244	112,536
5	150	697,126	244	116,531	600,078	605	281,901	287,359	252	117,149
average		637,094	246	115,511	583,634	519	236,801	280,231	234	107,714
1	180	779,541	291	143,701	630,719	656	336,636	368,406	286	137,869
2	180	787,472	282	145,298	658,172	603	300,112	280,625	252	130,068
3	180	726,625	260	130,131	621,959	780	377,106	364,312	272	135,264
4	180	615,433	311	154,600	748,125	646	324,216	310,437	267	133,674
5	180	715,929	258	128,969	679,281	750	378,369	313,672	300	145,951
average		765,000	280	140,138	767,531	685	343,292	327,490	275	136,371

[표 2] 실험결과

[표 2]는 단일 에이전트, 조정되지 않은 멀티 에이전트, 조정된 멀티 에이전트의 경우(MADFS)를 각각 비교 실험한 결과를 보여준다. 실험에는 총 노드수가 90,120,150,180개로 서로 크기가 다른 4개의 맵을 이용하였고, 각 맵당 5회씩 서로 다른 시작노드를 이용하여 실험하였다. 각 탐사방법에 대한 효율성 척도로서, 총 소요시간, 총 방문 노드수, 총 이동거리 등을 측정한 결과를 보여주고 있다.

실험결과를 살펴보면, 실험에 사용된 모든 방법이 맵상의 모든 노드를 빠짐없이 방문하는 완전성을 보였고, 탐사 효율성면에서는 차이를 보였다. 총 소요시간 면에서는 " 단일 에이전트 > 조정되지 않은 멀티에이전트 > 조정된 멀티에이전트" 순으로 점점 적은 시간에 탐사를 완료했다. 단일에이전트와 조정되지 않은 멀티에이전트는 시간 면에서는 거의 차이를 보이지 않지만 방문 노드수, 이동 거리 면에서는 단일 에이전트가 월등히 높은 성능을 발휘하여 조정되지 않은 멀티에이전트가 얼마나 비효율 적인지를 보여주어 멀티 에이전트에 있어서 조정이 기능이 얼마나 중요한지를 말해주고 있다. 본 논문에서 제안한 MADFS 알고리즘이 총 소요시간 면에서 가장 높은 성능을 보여주었고, 방문 노드수와 이동 경로 면에서는 단일에이전트와 비슷한 수준의 결과를 보여 줬다. 멀티 에이전트의 탐사시 약간의 중복이 발생하지만 방문 노드수와 이동경로가 단일에이전트와 비슷하게 측정되었다는 것은 다수의 에이전트를 이용한 탐사 방법이 효율적이었음을 의미한다.

6. 결론

본 논문에서는 자율성을 가지는 멀티에이전트를 이용하여 미지의 공간을 탐사하는 실시간 그래프 탐색 알고리즘인 MADFS 알고리즘을 제안하였고, 3차원 온라인 게임 환경을 이용한 실험을 통하여 모든 노드들과 에지들을 빠짐없이 탐색하는 완전성을 확인하였고, 단일에이전트의 경우와 조정되지 않은 멀티에이전트와의 비교를 통하여 효율성을 입증하였다.

참고문헌

[1] Reid Simmons, et al."Coordination for Multi-Robot Exploration and Mapping."Proceedings of National Conference on Artificial Intelligence, 2000.

[2] R. Adobbati, et al."Gamebots : A 3D Virtual World Test-Bed for Multi-Agent Research", Proceedings of International Conference on Autonomous Agents, 2001.

[3] Robert Zlot, et al."Multi-Robot Exploration Controlled by a Market Economy", Proceedings of IEEE International Conference on Robotics and Automation,2002.

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

[5] Yasuhiko Kitamura, "Organizational Strategies for Multiagent Real-Time Search", Proceedings of International Conference on Multi-Agent Systems,pp.150-156, 1996.

[6] 최은미, 김인철 "공간탐사를 위한 실시간 그래프 탐색" , 한국지능정보시스템학회 논문지, 제11회 제1호, pp.153-167, 2005