부분적으로 관측가능하고 비결정적인 계획문제를 풀기 위한 휴리스틱 탐색 알고리즘

(A Heuristic Search Algorithm for Solving Partially-Observable, Non-Deterministic Planning Problems)

김 현 식 † 박 찬 영 †
(Hyun-Sik Kim) (Chan-Young Park)

김 인 철 ††
(In-Cheol Kim)

요 약 본 논문에서는 불완전한 인식과 비결정적 동작을 함께 포함한 조건부 계획문제를 풀기 위한 새로운 휴리스틱 탐색 알고리즘 HSCP를 소개한다. HSCP 탐색 알고리즘은 하나의 완전한 해 그래프가 구해질 때까지 AND-OR 탐색시도를 반복한다. HSCP 알고리즘의 AND-OR 탐색시도는, 기존의 휴리스틱 AND-OR 탐색 알고리즘들인 AO*나 LAO*와는 달리, 오직 하나의 후보 해 그래프를 확장하는데 집중한다. 또한, 실시간 동적 프로그래밍 알고리즘들인 RTDP와 LRTDP와는 달리, 모든 상태들의 가치 평가치가 수렴할 때까지 미루지 않고 바로 해를 구한다. 따라서 HSCP 탐색 알고리즘은 양질의 조건부 계획을 매우 효율적으로 구해줄 수 있다는 장점이 있다.

키워드 : 조건부 계획문제, 휴리스틱 탐색, 믿음 상태

· 본 연구는 경기도의 경기도지역협력연구센터사업의 일환으로 수행하였음
· 이 논문은 2009 한국컴퓨터종합학술대회에서 '부분적으로 관측가능하고 비결정적인 계획문제를 풀기위한 휴리스틱 탐색 알고리즘'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 경기대학교 컴퓨터과학과
advance7@kyonggi.ac.kr
cyboys@kyonggi.ac.kr
†† 종신회원 : 경기대학교 컴퓨터과학과 교수
kic@kyonggi.ac.kr

논문접수 : 2009년 8월 13일
심사완료 : 2009년 9월 21일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제15권 제10호(2009.10)

Abstract In this paper, we present a new heuristic search algorithm, HSCP, that can solve conditional/contingent planning problems with nondeterministic actions as well as partial observations. The algorithm repeats its AND-OR search trials until a complete solution graph can be found. However, unlike existing heuristic AND-OR search algorithms such as AO* and LAO*, the AND-OR search trial conducted by HSCP concentrates on only a single candidate of solution subgraphs to expand it into a complete solution graph. Moreover, unlike real-time dynamic programming algorithms such as RTDP and LRTDP, the AND-OR search trial of HSCP finds a solution immediately when it possible without delaying it until the estimated value of every state converges. Therefore, the HSCP search algorithm has the advantage that it can find a sub-optimal conditional plan very efficiently.

Key words : Conditional Planning Problem, Heuristic Search, Belief State

1. 서 론

전통적인 인공지능 계획방식에서는 에이전트의 완전한 인식 능력과 결정적 동작 효과를 가정하였다. 이러한 가정에 따르면 에이전트는 늘 자신이 어떤 상태에 놓여 있든지 명확하게 알 수 있으며, 자신이 취할 행동의 결과도 정확하게 예측할 수 있다. 하지만 실세계 많은 계획 응용시스템들은 이러한 전통적인 인공지능 계획방식의 가정들을 만족하지 못하는 경우가 많다. 대부분의 실세계 응용들에서는 에이전트의 인식 한계가 존재하고, 환경의 가변성으로 인해 동작의 성공적인 실행을 보장할 수 없다. 그동안 인식의 불완전성과 동작의 비-결정성 문제를 해결해보고자 다양한 연구들이 진행되어 왔으며, 조건부 계획방식(conditional/contingent planning)은 이러한 연구들 중 대표적인 접근법이다[1]. 일반적으로 조건부 계획방식은 감지 동작(sensing action)의 존재와 일반 동작의 제한적인 비-결정성(bounded non-determinism)을 가정한다. 즉, 에이전트가 계획생성 단계에서는 환경의 상태에 대해 완전히 파악할 수 없어도, 계획실행 단계에 가서는 별도의 감지 동작을 실행함으로써 명확히 파악할 수 있다고 가정한다. 또, 미리 하나로 결정할 수는 없으나 비-결정적 동작의 실행결과는 계획생성 단계에서 예측 가능한 결과들 중의 하나가 될 것이라고 가정한다. 조건부 계획방식은 이러한 가정들을 기초로, 감지 동작들과 이들의 실행결과에 따라 조건부로 실행 가능한 동작들을 포함한 하나의 계획을 수립한다. 따라서 조건부 계획방식에서는 계획 실행 시 발생할 수 있는 다양한 상황들을 예측하고 미리 이 상황들에 대응할 수 있는 각각의 부분 계획을 포함하도록 하나의

계획을 수립한다.

조건부 계획방식은 불확실성이 내재된 환경에서 계획 실행의 실패 가능성과 재계획(replanning)의 필요성을 줄여주는 매우 효과적인 방법이다. 하지만 단일 해 경로(solution path)만 찾으면 되는 전통적인 계획방식과는 달리, 조건부 계획방식은 계획 내의 모든 분기 가지들이 목표에 도달 가능한 하나의 해 그래프(solution graph)를 구해야 하기 때문에, 탐색 공간이 매우 커지게 되고 탐색에 소요되는 계산시간과 메모리 양도 폭발적으로 늘어나는 문제가 있다.

본 논문에서는 인식의 불완전성과 동작의 비결정성이 함께 포함된 조건부 계획문제에 적합한 새로운 휴리스틱 탐색 알고리즘인 HSCP(Heuristic Search for Conditional Planning)를 소개한다. 본 논문에서 제안한 HSCP 탐색 알고리즘은 임의의 조건부 계획문제에 대해 양질의 해 계획을 매우 효율적으로 구해줄 수 있는 장점을 가지고 있다. 본 논문에서는 서로 다른 계획문제들에 대한 비교 실험을 통해 HSCP 알고리즘의 탐색 효율성을 분석해본다.

2. 계획 문제의 정형화

본 절에서는 계획문제의 상태와 동작들을 상태변수 표현법[2]에 따라 정의한다고 가정한다. 상태변수 표현법은 전통적인 술어논리 표현법[3][4]과는 달리, 하나의 상태를 유한개의 상태변수들(state variables, fluents)과 그들의 값(values)으로 표현한다.

일반적으로 불완전한 인식 능력과 동작의 비-결정성을 가지고 있는 한 에이전트가 자신의 행동을 계획하고자 할 때, 계획생성 초기단계나 계획생성 과정 중에 자신이 처한 현재 상태를 정확히 하나로 판단할 수 없는 경우가 많다. 이러한 상태들의 집합을 일반적으로 하나의 믿음 상태(belief state)라 부른다.

- 하나의 믿음 상태(belief state) bs는 각 상태변수 $v \in V$에 대한 값 할당을 나타내는 (v, x) 쌍들의 집합으로 표현된다. 하지만 이때 x는 기존의 상수 기호들 중의 하나인 $c \in C$가 될 수도 있고, 상태변수 v의 값을 현재는 정확히 알지 못한다는 것을 의미하는 메타 상수 unknown이 될 수도 있다. 상태변수 v의 값이 unknown인 하나의 믿음 상태 bs는 v에 할당 가능한 상수들이 c_1, c_2.., c_m이라면 각각 (v, c_1), (v, c_2), ..., (v, c_m)인 완전한 상태들(complete states)의 집합 S_C로도 표현할 수 있다. 즉 $bs_{v=unknown} = S_C = \{s_{v=c1}, s_{v=c2}, ..., s_{v=cm}\}$.

믿음 상태에 놓이는 에이전트에게는 현재 unknown인 상태변수의 값을 정확히 결정짓기 위한 별도의 감지 동작들(sensing actions)이 존재하는 것으로 가정한다. 이러한 감지 동작들은 환경의 상태를 변화시키는 일반 동작들(effective actions)과는 달리, 단지 현재의 환경 상태를 좀 더 분명히 알 수 있는 정보를 습득하는 효과를 가진다.

일반적으로 하나의 전통적 계획문제는 완전한 상태들의 집합과 결정적 일반 동작들만을 포함하지만, 인식의 불완전성과 동작의 비-결정성이 포함된 조건부 계획문제는 믿음 상태들의 집합과 감지 동작, 그리고 비-결정적 동작 등을 포함한다.

- 인식의 불완전성과 동작의 비-결정성이 포함된 조건부 계획문제 $P_{pond} = (bs_0, g, O_d \cup O_{nd} \cup O_s)$는 초기 믿음 상태 bs_0와 목표 g, 그리고 동작들의 집합 $O_d \cup O_{nd} \cup O_s$으로 주어진다. 이때 O_d는 결정적 일반 동작들(deterministic effective operators)의 집합을, O_{nd}는 비-결정적 일반 동작들(non-deterministic effective operators)의 집합을, O_s는 감지 동작들(sensing operators)의 집합을 각각 나타낸다.

본 논문에서는 인식의 불완전성과 동작의 비-결정성이 포함된 조건부 계획문제 P_{pond}에 대한 해 계획(solution plan)을 하나의 행동 정책(policy)으로 표현한다고 가정한다. 이러한 행동 정책은 조건부 계획을 나타내는 전통적인 계획 형태인 여러 개의 분기 가지(branch)를 가지는 해 그래프(solution graph)로 손쉽게 변환이 가능하다.

- 계획문제 P_{pond}에 대한 하나의 행동 정책 π_{bs}는 계획문제 P_{pond}의 믿음 상태 공간(belief state space) 위에서 정의되며, 믿음 상태 bs_i와 기본 동작 a_i의 쌍인 (bs_i, a_i)들의 집합으로 표현된다. 즉, $\pi_{bs} = \{(bs_i, a_i)\}$, i = 1, ..., n. 이러한 믿음 상태 공간상의 행동 정책 π_{bs}는 상태 공간상의 행동 정책 $\pi_s = \{(S_i, a_i)\}$, i = 1, ..., n로 변환 가능하다. 이때 집합 S_i는 행동 정책 π_{bs}의 한 믿음 상태 bs_i에 대응하는 완전한 상태들의 집합을 의미한다.

3. 탐색 알고리즘

인식의 불완전성과 동작의 비-결정성을 포함한 조건부 계획문제에 대한 하나의 해 계획은 탐색 경로가 모두 목표 상태에 도달 가능한 여러 개의 AND 분기 가지를 가진 하나의 그래프로 볼 수 있다. 따라서 이러한 해 계획을 얻기 위한 계획 알고리즘은 하나의 믿음 상태 공간 위에서 휴리스틱 AND-OR 탐색을 전개하는 알고리즘[5][6][7][8]으로 볼 수 있다. 본 논문에서 제안하는 탐색 알고리즘 HSCP(Heuristic Search for Conditional Planning)에서는 초기 믿음 상태에서 시작하여 휴리스틱 AND-OR 탐색을 통해 하나의 완전한 해 그래프를 찾아보는 일련의 과정을 한 번의 탐색시도

(search trial)로 본다. 하지만 HSCP에서 진행하는 탐색시도는 AO*와 같은 전통적인 휴리스틱 AND-OR 탐색과정과는 다르다. AO*와 같은 전통적인 휴리스틱 AND-OR 탐색에서는 한 번의 탐색시도 과정동안 다양한 여러 후보 해 그래프들을 서로 비교 평가하고, 그 중에서 가장 좋은 후보 그래프를 매번 선택하여, 이것의 말단 노드를 확장하는 과정을 계속한다. 따라서 전통적인 휴리스틱 AND-OR 탐색 알고리즘에서는 한 번의 탐색시도 동안에 다양한 후보 해 그래프들에 대한 평가와 확장, 그리고 해 그래프 전반에 걸친 재평가 작업을 반복함으로써, 단 한 번의 탐색시도를 통해 최적의 해 그래프를 찾아낸다.

하지만 HSCP 탐색 알고리즘의 AND-OR 탐색시도는 초기 믿음 상태에서 시작하여 모든 AND 분기 가지가 목표에 도달할 때까지 하나의 후보 해 그래프만을 확장해보는 탐색방식이다. 그리고 휴리스틱 평가치 갱신도 선택된 말단 노드에만 국한되어 발생하고, 후보 해 그래프 전체로 역전파 되지는 않는다. 이러한 독특한 탐색방식으로 인해 HSCP 알고리즘의 AND-OR 탐색시도는 전통적인 휴리스틱 AND-OR 탐색시도에 비해 매우 높은 탐색 효율성을 나타내지만, 한 번의 탐색시도만으로는 탐색의 최적성(optimality)과 완전성(completeness)을 보장할 수는 없다. 따라서 HSCP 알고리즘에서는 성공적으로 하나의 완전한 해 계획을 얻을 때까지 이러한 탐색시도를 계속해서 반복한다. 대신 HSCP 알고리즘은 매번 새로운 탐색시도를 전개할 때마다 이전의 탐색시도를 통해 습득한 상태 평가치를 기초로 새로운 후보 그래프를 확장해봄으로써 해 계획을 찾을 가능성을 높여 나간다.

HSCP 탐색 알고리즘은 표 1에 기술된 것처럼 하나의 해 그래프를 얻을 수 있을 때까지 탐색시도인 AND-OR-Search-Trial을 반복한다(표 1의 line 5~6). 이 알고리즘에서 solved는 한 번의 탐색시도인 AND-OR-Search-Trial을 통해, 해 그래프의 모든 AND 분기 가지가 목표까지 성공적으로 확장되었을 때 참이 된다. 반면에 이러한 AND 분기 가지 중 어느 하나라도 막다른 종료점(dead-end)이나 순환구조(cycle)를 만나 목표까지 확장할 수 없을 때는 solved를 거짓으로 반환한다.

표 1 HSCP 알고리즘

```
1.  HSCP(bs0 : initial belief state)
2.  Begin
3.     valueTable.CLEAR();
4.     solved = false;
5.     While solved = false do
6.        AND-OR-Search-Trial(bs0);
7.     return (policyTable.GET_POLICY());
8.  End
```

표 2 AND-OR-Search-Trial 알고리즘

```
1.  AND-OR-Search-Trial(bs0)
2.  Begin
3.    stateStack.CLEAR();
4.    policyTable.CLEAR();
5.    localHistory.CLEAR();
6.    stateStack.PUSH(bs0);
7.    While ¬stateStack.EMPTY() do
8.  /* if there remain any unexplored branches */
9.      bs = stateStack.POP();
10.     If bs.GOAL() then  /*bs is a goal state on
11. the end of one branch*/
12.       solved = true;
13.       bs.VALUE = 0;
14.       valueTable.UPDATE(bs, bs.VALUE);
15.       localHistory.CLEAR();
16.       continue;
17.     If localHistory.CONTAINS(bs) then /* bs is
18. a cycle state, not be included in a solution */
19.       solved = false;
20.       break;
21.     else
22.       localHistory.PUT(bs);
23.     If policyTable.CONTAINS(bs) then
24.       /* bs is already in a partial solution */
25.       localHistory.CLEAR();
26.       continue;
27.     a = bs.GREEDY_ACTION();
28.     If a = NULL then
29.       /* bs is a dead-end state */
30.       solved = false;
31.       bs.VALUE = MAX_VALUE
32.       valueTable.UPDATE(bs, bs.VALUE);
33.       break;
34.     policyTable.PUT(bs, a);
35.     If a is one of sensing actions then
36.       bs' = bs.P_NEXTSTATE(a);/*a positive*/
37.       bs" = bs.N_NEXTSTATE(a);/*a negative*/
38.       localHistory.CLEAR();
39.       stateStack.PUSH(bs');
40.       stateStack.PUSH(bs");
41.       bs.VALUE = cost(bs,a) +0.5*
42.  valueTable.GET(bs')+0.5*valueTable.GET(bs")
43.       valueTable.UPDATE(bs, bs.VALUE);
44.     else if a is one of non-deter actions then
45.       next_states = bs.ND_NEXTSTATE(a);
46.       localHistory.CLEAR();
47.       For each bs' in next_states do
48.          stateStack.PUSH(bs');
49.       bs.VALUE = cost(bs,a) +
50.            ∑bs'∈next_states  valueTable.GET(bs')  /
    |next_states|
51.       valueTable.UPDATE(bs, bs.VALUE);
52.     else if a is one of deter actions then
53.       bs' = bs.NEXTSTATE(a);
54.       stateStack.PUSH(bs');
55.  bs.VALUE = cost(bs,a) + valueTable.GET(bs')
56.       valueTable.UPDATE(bs, bs.VALUE);
57. End
```

AND-OR-Search-Trial을 통해 하나의 완전한 해 그래프로 확장이 가능한 경우, 이 AND-OR-Search-Trial 과정동안 policyTable에 기록된 행동 정책을 반환하고 종료한다(표 1의 line 7). 또한 HSCP 알고리즘은 AND-OR-Search-Trial 과정동안 방문하는 각 믿음 상태에 대한 휴리스틱 평가치를 저장할 valueTable을 한

차례 초기화한다(표 1의 line 3). 하지만 이후 반복되는 새로운 AND-OR-Search-Trial 때는 valueTable을 초기화하지 않고, 이전의 탐색시도 과정동안 갱신되어온 vaueTable내의 상태 평가치들을 그대로 사용한다.

표 2는 HSCP 알고리즘에서 반복하는 한번의 탐색시도 과정인 AND-OR-Search-Trial을 기술하고 있다. AND-OR-Search_Trial에서는 새로 생성되는 믿음 상태들을 저장하는 stateStack, 해 그래프의 일부이자 행동 정책의 일원으로 결정된 (믿음 상태, 동작)의 쌍들을 저장하는 policyTable, 순환구조 분석을 위해 이용되는 localHistory 등의 자료구조를 초기화한다(표 2의 line 3~5). AND-OR-Search-Trial은 초기 믿음 상태에서 시작하여, stateStack에 더 이상 확장할 믿음 상태가 남아 있지 않을 때까지 믿음 상태 공간상에서 휴리스틱 AND-OR 탐색을 계속한다(표 2의 line 7~56). 현재의 믿음 상태에서 실행 가능한 동작들 중 하나를 선택하는 것은 OR 노드 중 하나를 선택하는 것이고, 선택된 동작을 적용함으로써 발생 가능한 다수의 결과 상태들은 AND 노드들에 해당된다. stateStack에는 언제나 현재 후보 해 그래프의 미확장 AND 노드들이 저장된다. AND- OR-Search-Trial은 먼저 stateStack에 보관 중인 믿음 상태들 중에서 가장 마지막에 저장된 믿음 상태를 선택하고, 이를 stateStack에서 삭제한다(표 2의 line 9). 선택된 믿음 상태가 목표를 만족하는지(표 2의 line 10~16), 닫힌 순환 상태(closed cycle state)인지, 아니면 열린 순환 상태(open cycle state)인지(표 2의 line 17~26), 더 이상 확장이 불가능한 막다른 종료점(dead-end) 상태인지(표 2의 line 28~33) 검사하고, 각 경우에 합당한 처리과정을 수행한다. 만약 선택된 현재의 믿음 상태가 위의 어떤 경우에도 해당되지 않는다면, valueTable에 저장된 상태 평가치들을 기초로 현재 상태에 적용 가능한 가장 좋은 동작을 하나 선택하고(표 2의 line 27), (현재 믿음 상태, 동작)의 쌍을 해 계획의 일원으로 결정하고 policyTable에 저장한다(표 2의 line 34). 그리고 선택된 동작이 감지 동작인지, 비-결정적 일반 동작인지, 아니면 결정적 일반 동작인지를 판단하여, 각 경우에 따른 적절한 후속 상태들의 생성과 평가치 갱신 작업을 수행한다(표 2의 line 35~56).

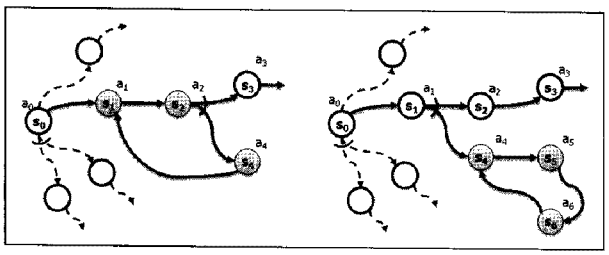

그림 1 순환구조의 유형

믿음 상태 공간상에서 AND-OR-Search-Trial 탐색을 진행하다보면 이전에 방문한 적이 있는 상태를 또 다시 만나는 즉, 순환구조(cycle)를 발견하게 된다. HSCP 알고리즘에서는 앞서 표 2에서 기술한 것처럼 현재의 믿음 상태가 policyTable에만 이미 들어있는 경우에는 그림 1의 왼쪽과 같이 목표에 도달할 가능성이 있는 열린 순환구조(open cycle)로 판단하고, 반면에 localHistory에 들어있는 경우에는 그림 1의 우측과 같이 목표에 도달할 수 없는 닫힌 순환구조(closed cycle)라고 판단한다.

HSCP 탐색 알고리즘은 AND-OR-Search-Trial을 반복함에 따라 탐색시도 과정동안 방문하는 각 믿음 상태에 대한 평가치는 계속해서 좀 더 정확한 값으로 갱신되면서 점진적으로 수렴하게 된다. 하지만 HSCP 탐색 알고리즘은 대표적인 실시간 동적 프로그래밍 알고리즘들인 RTDP나 LRTDP와 달리 관련 상태들의 평가치

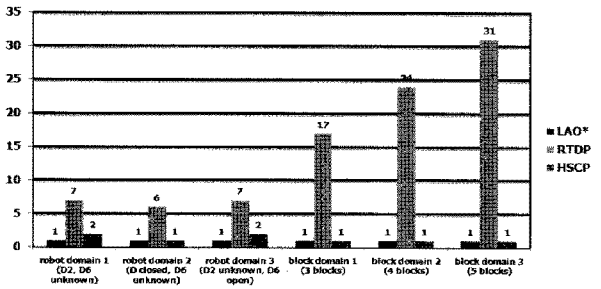

그림 2 탐색시도 횟수

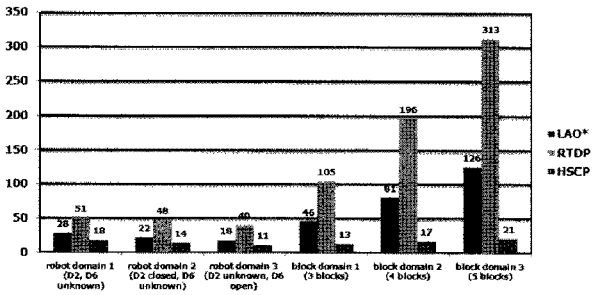

그림 3 상태 평가치 갱신 횟수

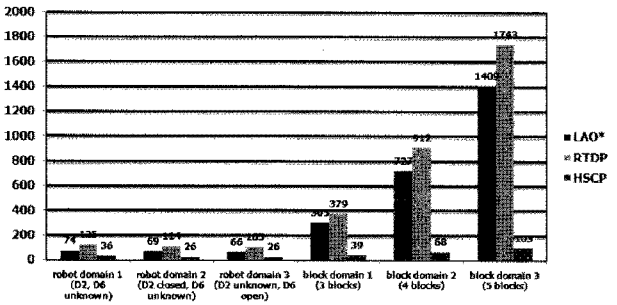

그림 4 생성한 상태 수

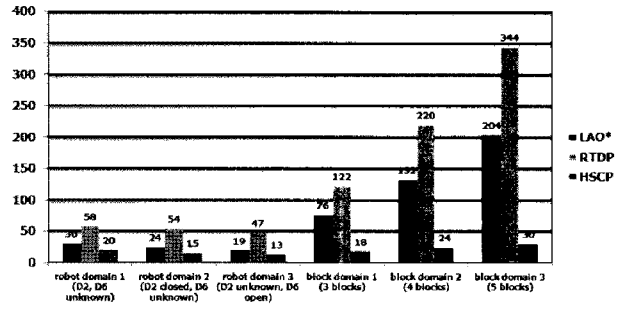

그림 5 방문한 상태 수

가 모두 충분히 수렴하기 전이라도 AND-OR-Search-Trial을 통해 성공적으로 하나의 해 계획을 찾을 수 있으면 종료한다.

4. 실험 및 평가

본 논문에서 제안한 HSCP 알고리즘을 대표적인 조건부 계획 알고리즘들인 LAO* 알고리즘, RTDP 알고리즘 등과 비교실험을 전개해봄으로써, HSCP 알고리즘의 효율성을 확인하고자 하였다. 실험에는 로봇 도메인과 블록 도메인을 이용하였으며, 각 도메인마다 임의로 생성한 3개의 계획 문제를 실험에 사용하였다. 총 탐색시도의 횟수, 총 상태 평가치 갱신의 횟수, 총 생성한 상태의 수, 총 방문한 상태의 수 등을 알고리즘의 효율성을 비교하기 위한 평가 척도로 사용하였다. 그림 2, 그림 3, 그림 4, 그림 5는 총 6 개의 서로 다른 계획문제를 풀어본 결과로서, 세 알고리즘의 총 탐색시도의 횟수, 총 상태 평가치 갱신의 횟수, 총 생성한 상태의 수, 총 방문한 상태의 수를 각각 나타내고 있다. 예상한대로 총 탐색시도는 LAO*나 HSCP에 비해 RTDP가 월등히 많다는 것을 확인할 수 있고, 이러한 현상은 블록 도메인 문제들에서 더 뚜렷이 나타났다. 근소한 차이지만 HSCP가 LAO*보다 탐색시도가 조금 더 많다는 것도 알 수 있다. 상태 평가치 갱신의 횟수를 비교해보면, RTDP가 가장 많고 그 다음이 LAO*, 그리고 HSCP가 가장 적은 것을 확인할 수 있다. 생성한 상태 수와 방문한 상태의 수를 비교해보면, HSCP가 다른 두 알고리즘에 비해 가장 적은 수의 상태를 생성하고 방문하였음을 확인할 수 있다. 또한 전체적으로 로봇 도메인보다 블록 도메인 문제에서 다른 두 알고리즘에 비해 HSCP 알고리즘의 탐색 효율성이 매우 높다는 것을 확인할 수 있다. 블록 도메인은 로봇 도메인보다 좀 더 많은 비-결정적 동작들을 포함하고 있고, 이들에 의해 발생하는 순환구조도 더 많은 특성을 지니고 있다. 따라서 HSCP 알고리즘의 효율성은 특히 이러한 특성을 지닌 문제 도메인에서는 더 큰 효과를 발휘할 수 있음을 알 수 있다.

5. 결 론

본 논문에서는 인식의 불완전성과 동작의 비결정성이 함께 포함된 조건부 계획문제를 정형화하고, 이 계획문제를 위한 표현법과 새로운 휴리스틱 탐색 알고리즘 HSCP를 소개하였다. HSCP 탐색 알고리즘은 임의의 조건부 계획문제에 대한 최적의 해 계획을 보장하지는 못하나, 양질의 해 계획을 매우 효율적으로 구해줄 수 있다. 로봇 도메인과 블록 도메인 등의 계획문제를 이용한 비교 실험을 통해 기존의 휴리스틱 AND-OR 탐색 알고리즘 LAO*와 실시간 동적 알고리즘 RTDP 보다 HSCP 알고리즘이 더 높은 탐색 효율성을 보임을 입증하였다. 또한 HSCP 알고리즘은 블록 도메인과 같이 많은 비-결정적 동작과 순환구조가 포함된 조건부 계획문제들에서 상대적으로 더 높은 효율성을 가짐을 확인하였다. 향후에는 HSCP 알고리즘의 완전성을 증명하고, 탐색과정에 발생하는 많은 믿음 상태들과 행동 정책을 보다 효율적으로 관리할 수 있는 표현법에 대해 연구할 필요가 있는 것으로 판단한다.

참 고 문 헌

[1] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 2nd Edition, Prentice Hall, 2003.

[2] M. Ghallab, D. Nau, and P. Traverso, Automated Planning: Theory and Practice, Morgan Kaufmann, 2004.

[3] H. L. S. Younes and M. Littman, "PPDDL1.0: An Extension to PDDL for Expressing Planning Domains wit Probabilistic Effects," *Technical Report CMU-CS-04-167*, 2004.

[4] M. Fox and D. Long, "PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains," *Journal of Artificial Intelligence Research*, vol.20, pp.61-124, 2003.

[5] E. Hansen and S. Zilberstein, "LAO*: A Heuristic Search Algorithm that Finds Solutions with Loops," *Artificial Intelligence*, vol.129, no.1-2, pp.35-62, 2001.

[6] B. Bonet and H. Geffner, "Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming," *Proceeding of the ICAPS'03*, pp.12-21, 2003.

[7] U. Kuter, D. Nau, E. Reisner and R. Goldman, "Conditionalization: Adapting Forward-Chaining Planners to Partially Observable Environments," *Proceeding of the ICAPS'07*, 2007.

[8] U. Kuter and D. Nau, "Forward-Chaining Planning in Nondeterministic Domains," *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp.513-518, 2004.