Anycase Subgoal을 위한 EBL 기반의 제어지식형 계획기의 확장

이 동 복, 이 수 원
숭실대학교 컴퓨터학부

Extending An EBL Based Control-Knowledge Planner for Anycase Subgoals

DongBok Lee, Soowon Lee
School of Computing, Soongsil University.

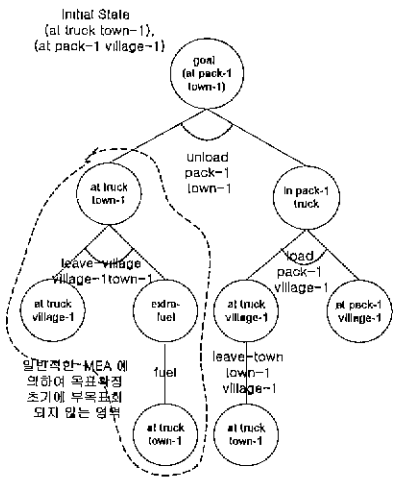
요 약

본 논문은 EBL 기반의 제어지식형 계획기에서 다양한 목표확장 방법을 사용하여 MEA의 불완전한 계획생성을 해결하는 새로운 방법을 제안한다. 계획기의 문제 공간을 탐색하는 방법 중 하나인 MEA는 현재상태와 목표상태의 차이를 줄이기 위하여 연산자를 선택한 후에, 연산자의 조건절을 현재상태가 만족하는지의 여부에 따라서 조건절의 부목표화를 결정한다. 그러나 이러한 목표확장 방법은 현재상태에서 만족된 부목표에 대한 목표확장을 하지않음으로써 문제공간 탐색에서 제한된 범위만을 탐색하므로 목표를 만족하는 최적의 계획을 생성할 수 없으며, 또한 문제를 해결하는 계획이 있음에도 불구하고 탐색범위의 제한으로 인해 계획을 생성하지 못하는 경우도 발생한다. 이와 같이 현재상태에서 만족되어 목표확장을 하지 않은 부목표를 Anycase Subgoal이라 한다. 본 논문에서 제안하는 목표확장 방법은 EBL기반의 제어지식형 계획기를 Anycase Subgoal을 위하여 확장하는 방법으로서, 초기의 문제공간 탐색에서 사용된 목표확장 방법에서 문제를 해결하지 못할 경우 탐색공간을 확장하여 문제를 해결하고, 문제에 적합한 목표확장 방법을 제어지식형 규칙으로 학습하여 유사한 문제에 대하여 효율적으로 계획을 생성한다

1. 서론

일반적으로 계획기는 주어진 상위레벨의 목표를 해결하기 위하여 문제공간의 탐색을 통하여 연산자의 순서를 결정하고 생성한다[5] 계획기는 계획표현 방법의 관점에 따라 전순서 계획(Total Order Planner), 부분순서 계획기(Partial Order Planner) 및 제어규칙형 계획기로 분류할수 있으며 최근 지능형 에이전트에 대한 연구가 활성화됨에 따라 이에 적합한 EBL 기반의 제어지식형 계획기에 대한 연구가 이루어지고 있다[2]. 제어지식형 계획기에서의 계획 결과는 계획중에 탐색을 제어하는 규칙의 형태로 나타나며, 목표확장방법은 일반적으로 MEA를 사용한다 MEA는 현재상태(Current State)와 목표상태(Goal State)사이의 차이를 발견하고 이를 줄이기 위하여 연산자를 선택한 후에, 현재상태가 연산자의 조건절을 만족하는지의 여부에 따라서 연산자의 조건절(Precondition)을 부목표(Subgoal)로 만든다 이러한 과정을 연산자 부목표화(Operator Subgoaling)라고 하는데, 부목표화를 통한 문제공간 탐색은 제한된 범위에서만 탐색을 하므로 최적의 해를 찾지 못하거나 또는 문제를 해결하지 못하는 경우도 발생한다 다음과 같은 문제를 고려해 보자

초기 상태에서의 화물(package)은 마을(village)에 있고 트럭은 도시(town)에 있으며, 목표는 트럭을 이용하여 화물을 마을에서 도시로 운반하는 것이며, 트럭의 연료탱크가 제한적이기 때문에 여분의 연료가 없으면 한번밖에 움직일 수 없고, 또한 주유소는 도시에만 존재한다는 제약이 있다 <그림 1>은 이 문제에서 MEA의 인신자 부목표화를 통한 목표계층을 보여주고 있다



<그림 1> MEA의 부목표화를 통한 목표계층

최상위의 Root 노드는 초기에 주어진 상위레벨의 목표를 나타내고 나머지 노드들은 연산자의 조건절로 인하여 생긴 부목표를 나타낸다. 아크는 연산자를 의미하며, 음영으로 된 노드들은 초기상태에서 이미 성취(achieved)된 목표를 나타내고 있다. 점선으로 된 영역은 문제해결을 위해서는 목표확장 초기에 고려되어

야 하나 일반적인 MEA에 의해여 초기에 부목표화되지 않는 부분이다. 그림에서와 같이 Unload연산자의 조건절 두 개 중에 하나는 성취된 상태(at truck town-1)이기 때문에 미성취된 조건절(in pack-1 truck)만을 고려하여 부목표화 한다 이와 같이 부목표화를 진행하다기 현재상태를 만족하는 연산자(leave-town town-1 village-1)를 적용하면, 기존의 성취된 목표(at truck town-1)를 미성취로 만들게 되며 현재상태를 변화(at truck village-1)시킨다 이 변화된 현재상태는 미성취된 목표의 부목표(Extra-Fuel)를 위한 연시자(Fuel)의 조건절(at truck town-1)을 만족할 수 없기 때문에, 결국 목표를 해결하는 계획을 생성하지 못하게 된다. 즉 MEA의 부목표화를 통한 목표확장 방법은 현재상태에서 이미 성취되어 있는 목표는 고려하지 않고 미성취된 목표만을 고려하기 때문에 성취된 목표에서 확장할 수 있는 부목표들을 탐색하지 못하게 되므로 완전한 계획을 생성할 수 없다 이와 같이 "**현재상태에서는 목표가 만족되어서 목표확장을 하지 않은 부목표를 Anycase Subgoal**"이라 한다[4].

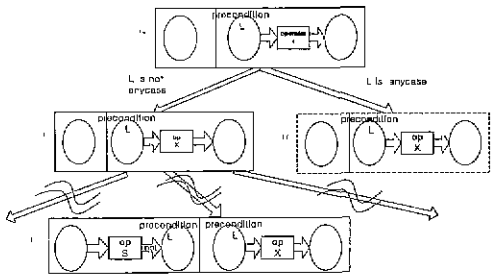
본 논문에서는 일반적인 MEA의 한계로서 발생하는 Anycase Subgoal을 해결하기 위하여 탐색공간이 확장된 EBL 기반의 제어지식형 계획기를 범용 인공지능 시스템인 Soar를 이용하여 구현하였다. 확장된 제이지식형 계획기는 문제에 대한 목표확장방법을 학습하여 유사한 유형의 문제에 대하여 학습된 제어지식을 이용할수 있다.

본 논문의 2절에서는 MEA의 한계로서 발생하는 Anycase Subgoal이 발생하는 문제를 여러 계획기에서 해결하는 방법들을 살펴본 후, 3절에서는 탐색공간이 확장된 EBL 기반의 제어지식형 계획기에서의 해결 방법을 기술한다 4절에서는 구현한 제어지식형 계획기에서 학습된 제어지식이 Anycase Subgoal 문제에 적용되는 결과를 살펴본다. 5절에서는 결론 및 향후 연구방향에 대해 기술한다.

2. 여러 계획기에서의 해결방법

2.1 양방향 계획기 - Prodigy, Rasputin

Prodigy는 양방향 계획기로서 하나의 노드는 Head-Plan과 Tail-Plan으로 구성되어 있다[3, 7] Head-Plan은 연산자의 적용을 나타내며, 초기상태에서 목표상태까지의 연산자의 전 순서를 의미한다. Tail-Plan은 목표상태로부터 연산자 부목표화를 통한 목표확장을 의미한다. 그러나 Prodigy의 Tail-Plan에서는 MEA를 사용한 목표확장을 하므로 Anycase Subgoal이 발생하는 문제를 해결할 수 없다 Prodigy에서 Anycase Subgoal을 해결하기 위해서는 모든 조건절과 최상위 레벨의 목표를 현재상태에 만족 여부에 관계없이 Anycase Subgoal로 간주하여, 목표확장하여 문제를 해결하여야 한다 그러나 이 방법은 부목표의 수가 많아질수록,특히 현재상태에 만족된 부목표의 수가 많을수록 불필요한 탐색을 하므로 효율성이 떨어진다. Rasputin[4]은 양방향 계획기의 이러한 불완전성을 보안하기 위하여 제안된 계획기이다 <그림 2>는 Rasputin에서 Anycase Subgoal을 해결하는 방법을 보여주고 있다

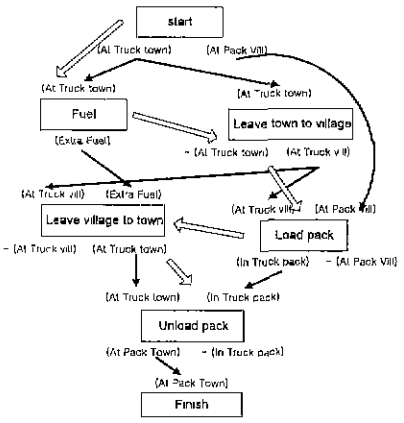


<그림 2> Anycase Subgoal 확장방법

그림의 (a)에서와 같이 Tail-Plan에 조건절 L을 갖는 연산자 X가 추가되었을 경우에는 그림에서의 (b)와같이 초기의 조건절 L은 Anycase Subgoal이 아니다. 그러나 문제를 해결하는 중에 (c)와 같이 Head-Plan에서 연산자 S를 적용한 결과가 이미 성취된 목표인 조건절 L을 위협하여 미성취 시키려 한다면 조건절 L은 Anycase Subgoal이라 표시된다. 만약 선택된 Branch에서, 탐색이 끝난 후에도 목표를 만족하는 계획을 생성하지 못했다면 Anycase Subgoal을 생성하는 분기점 (a)로 Backtracking이 발생하고 (d)방향으로 목표확장하고 탐색하여 계획을 생성한다 이 방법은 문제를 해결할 때에 얻은 의존적 방향 Backtracking에 대한 학습기능이 없기 때문에 유사한 문제에 대하여 반복된 Backtracking을 하며 또한 Anycase Subgoal의 개수가 많아질수록 Backtracking 횟수가 많아진다

2.3. 부분 순서 계획기 - UCPOP

대표적인 부분순서계획기인 UCPOP[1, 6]에서는 연산자(Leave town-1 to village-1)의 추가로 인하여 초기에 생성된 Causal link(start $\xrightarrow{\text{(at truck town)}}$ Unload pack)가 위협(Threat)을 받으나 새로운 연산자(Leave village to town)로 인하여 위협을 풀 수 있다 <그림 3>은 UCPOP에서 트릭 문제를 해결하며 생성된



<그림 4> UCPOP에서의 계획

계획을 보여주고 있다

그림에서 박스는 연산자를 의미하며, 점은 희살표는 Causal link, 흰 화살표는 연산자의 순서를 의미한다. 또한 음영으로 된 박스는 Causal link를 해결하는 연산자를 의미한다. 그러나 UCPOP은 학습기능이 없기 때문에 문제를 해결할 때마다 반복된 Backtracking을 한다.

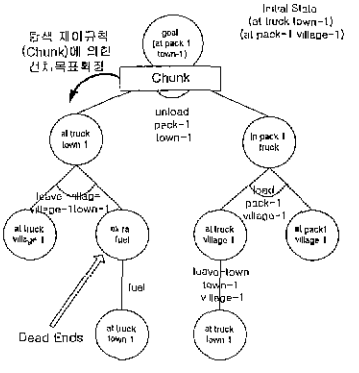
3. 탐색방법이 확장된 EBL 기반의 제어지식형 계획기에서의 해결방법

일반적인 EBL 기반의 제어지식형 계획기에서는 앞에서 설명한 Anycase Subgoal이 발생하는 문제영역에 대해서 목표상태로의 계획이 있음에도 불구하고 탐색공간을 편협하게 탐색됨으로써 계획을 찾지 못하는 경우가 생긴다. 이러한 한계를 극복하기 위해서 본 연구에서는 두 가지의 목표확장 방법을 이용한다 첫째는 계획기내에서 미성취된 목표만을 고려한 부분목표확장(partial goal expansion)방법이며, 둘째는 모든 목표들을 고려한 전체목표확장(whole goal expansion)방법이다. 초기에 최상위 레벨의 목표가 주어졌을 경우에는 먼저 미성취된 목표만을 고려한 부분목표확장 방법을 적용하여 목표가 성취되면 제어규칙형

계획을 생성하고, 그렇지 않다면 문제에 Anycase subgoal이 존재한다는 것을 의미하므로 모든 목표들을 고려한 전체목표확장 방법을 적용한다. 이와 같은 방법은 모든 계획 문제 영역에 똑같이 적용 가능한 방법으로서, 기존의 여러 계획기와는 달리 주어진 문제에 대한 목표확장방법을 학습할 수 있다. 학습된 제어지식형 계획기에서는 제한된 탐색공간에서 계획을 생성할 수 있다면 탐색시간을 절약할 수 있으며 그렇지 않다면, 문제영역에 대한 목표확장방법을 학습할 수 있으므로 문제영역과 유사한 문제영역에 대하여 학습된 제어지식을 이용하여 부분목표확장 방법에 따른 불필요한 탐색을 하지 않는다. 이러한 방법은 탐색공간을 확대하여($M_{부분목표확장} \rightarrow M_{전체목표확장}$) 문제를 효율적으로 해결하는 방법으로 *Multi Method Planner*에서 사용한 Bias-Relaxation과 유사한 방법이라고 할 수 있다[8]

4. 실험 및 결과

탐색공간이 확장된 EBL 기반의 제어지식형 계획기는 범용 인공지능 구조인 Soar에서 구현되었다. Soar에서 제어규칙은 제시된 연산자에 선호(Preference)를 첨가하여 표현하며, 선호의 종류에는 용인(Acceptable, +), 최상(Best, >), 최하(Worst, <)등이 있다. <그림 4>는 구현된 제어지식형 계획기에서 학습된 목표확장방법에 대한 제어지식이 문제에 적용되는 과정을 보여주고 있다. 초기의 목표확장 방법은 부분목표확장(In pack-1 truck)에 의한 탐색중에 Dead Ends 노드를 만나 탐색을 실패로 만든다. *실패한 목표확장방법은 제어지식형 규칙형태인 Chunk*로 만들어지, 계획기의 탐색방법을 전체목표확장으로 변환하여 탐색을 제어한다.



<그림 4> 학습된 목표확장방법의 적용

탐색실패로 인해 학습된 제어지식형 규칙은 <그림 5>의 (a)와 같다. 규칙의 조건절은 초기의 주어진 상태와 같이 트럭과 화물은 각각 도시와 마을에 있고 목표가 주어졌으며, 여분의 연료가 없을 경우를 의미하며, 결과절은 문제공간에 Anycase Subgoal이 존재하며 Subgoal의 상태 및 부모노드가 존재한다는 내용을 용인한다. <그림 5>의 (b)는 문제에 Anycase Subgoal이 존재할 때 목표확장방법을 제어하는 규칙을 나타내고 있다. 이와 같이 학습된 규칙은 조건절을 만족하는 유사한 문제에 대하여 바로 적용이 가능하며 제어지식형 계획기의 탐색 제어에 사용된다.

5. 결론 및 향후 연구 계획

본 논문에서는 Anycase Subgoal이 존재하는 문제를 통하여 계획기에서 주로 사용되는 MEA의 한계를 제조명하였으며, 이의 해결을 위하여 탐색공간을 확장할 수 있는 새로운 목표확장방법을 제안하였다. 제시된 방법은 Soar를 이용하여 EBL 기반의 제

```
만약
   문제공간은 드릭의 세계 ∧ 제시된 연산자는 부분목표확장
   ∧ <p1>은 화물 ∧ <t1>은 도시 ∧ <v1>은 마을
   ∧ 현재상태는 (at 트릭 <t1>) ∧ 현재상태는 (at <p1> <v1>)
   ∧ 현재상태에 (not extra-fuel)
   ∧ <u1>은 미성취된 목표 ∧ <u1>은 (at <p1> <t1>)
그러면
   현재상태에 (Anycase-Subgoal <a1>)이 존재
   ∧ <a1>는 (at 트릭 <t1>)
   ∧ <a1>의 부모노드는 <u1>

(a) 탐색 실패로 학습된 제어규칙

만약
   문제공간은 트릭의 세계 ∧ 제시된 연산자는 부분목표확장
   ∧ <p1>은 화물 ∧ <t1>은 도시 ∧ <v1>은 마을
   ∧ 현재상태는 (at 트릭 <t1>) ∧ 현재상태는 (at <p1> <v1>)
   ∧ 현재상태에 (Anycase Subgoal <a1>)이 존재
   ∧ <a1>은 (at 트릭 <u1>) ∧ <a1>의 부모노드는 <u1>
   ∧ <u1>은 미성취된 목표 ∧ <u1>은 (at <p1> <t1>)
그러면
   현재상태에서 부분목표확장 연산자에 대한 선호는 최하이다.

(b) 목표확장방법에 대한 규칙
```

<그림 5> Anycase Subgoal에 대한 탐색제어 규칙

지식형 계획기로 구현하였다.

향후 연구 방향은 첫째, Anycase Subgoal에 대한 실험적 평가가 필요하다. 현재 Anycase Subgoal이 발생하는 문제영역이 제한적이기 때문에 제안한 목표확장방법이 적용 가능한 다른 문제영역에 대한 연구가 필요하다. 둘째, 기존의 목표확장 방법을 분석하여 좀더 다양한 목표확장방법에 대한 연구가 필요하다. 현재의 목표확장방법은 부분목표확장 방법과 전체목표확장 방법, 두 가지로서 개념적으로 상호 메타적인 목표확장 방법이다. 이 두가지 방법의 중간적인 목표확장방법에 대한 연구가 필요하다.

5. 참고문헌

[1] Daniel S. Weld, An Introduction to Least Commitment Planning, *AI Magazine*, pp 27-61, 1994

[2] 이경현, 이수원, 지능형 에이전트를 위한 계획 표현 방법 1997년 *인지과학회 춘계학술 발표* 논문집

[3] Manuela Veloso, Peter Stone, FLECS- Planning with a Flexible Commitment Strategy, *Journal of Artificial Intelligence Research 3*, AI Access Foundation and Morgan Kaufmanu Publishers, 1995.

[4] Jim Blythe, Eugine fink, RASPUTIN' A Complete Bidirectional Planner, Fourth Conference on AI Planning Systems, Pittsburgh, 1998.

[5] David Chapman, Planning for Conjunctive Goals, *Artificial Intelligence*, 27(1) 97-109, 1987

[6] David McAllester David Rosenblitt, Systematic Nonlinear Planning, *Mathematical Foundation of Planning*, pages 634-639, 1991

[7] O. Etzioni, Why PRODIGY/EBL work. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 916-922, Boston, MA, 1990.

[8] Soowon Lee, Paul S Roenbloom, Granularity in multi-method planning In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 486-491, Washington D.C.,1993.