제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

아일랜드 탐색 기반 인공지능 계획자 설계

임 재걸

동국대학교 경주캠퍼스 컴퓨터학과 교수

yim@mail.dongguk.ac.kr

A Design of an AI Planner Based on Island Search

Yim, Jaegeol

Dept. of Computer Science, Dongguk University at Kyung-Ju

yim@mail.dongguk.ac.kr

요 약

인공지능 로봇 계획 문제는 초기상태, 동작, 목적상태로 구성되며, 초기상태를 목적상태로 변화시키는 일련의 동작을 찾는 문제로서, 지수 복잡도 문제이다. 이러한 문제에 대한 접근 방법으로 인공지능 탐색이 널리 쓰인다. 본 논문에서는 아일랜드 탐색을 사용하는 방법을 소개한다. 아일랜드 탐색을 적용하려면 초기상태에서 목적상태로 변환하는 도중 꼭 거쳐야 하는 아일랜드를 제공해야 한다. 그러나 그러한 아일랜드를 찾는 것은 불가능한 일이다. 그러므로, 본 논문에서는 선취관계를 이용하여 적당한 아일랜드를 생성하여 사용한다. 로봇 계획 문제의 목적 상태를 구성하는 부분목적 사이에 어떤 부분 목적이 반드시 다른 어떤 부분 목적 보다 먼저 성취되어져야 하는 관계를 선취관계라 한다. 아일랜드를 프로세서 수만큼 생성하여, 각 프로세서에 하나의 아일랜드를 원래의 계획 문제와 함께 할당한다. 각 프로세서는 초기상태에서 아일랜드까지 가는 문제를 휴리스틱 방법으로 풀고, 아일랜드에서 목적상태로 도달하는 문제를 또한 휴리스틱 방법으로 풀어 결합함으로써 원래의 문제에 대한 해를 구하여 주 프로세서에게 되돌려 준다. 주 프로세서는 되돌아 온 해 중에서 가장 효율적인 해를 선택하여 최적해를 찾는다.

1. 서론

인공지능 계획자는 초기상태, 목적상태, 로봇이 할 수 있는 동작들이 주어질 때, 초기상태를 목적상태로 바꾸는 일련의 동작을 찾아주는 컴퓨터 시스템이다. 최초의 인공지능 계획 시스템으로 GPS[1]를 들 수 있는데, GPS는 로봇이 수행할 수 있는 동작의 수가 유한하며, 로봇이 작업하는 환경이 로봇 동작의 함수로 결정되고, 오류가 발생하는 일은 없다라고 가정하였다. 초기의 계획 시스템들은 모두 이러한 가정하에 연구되었는 데, 근래에는 이러한 가정이 너무 비현실적이라 하여 다른 개체에 의하여 작업 환경이 수시로 바뀔 수 있다든지, 로봇이 작업 수행 중에 실수를 저지를 수 있다든지 하는 등 문제를 더욱 실제와 비슷하게 만들어 나가는 경향이 있다. 이렇게 더욱 복잡하게 된 문제와 구별하여 원래 GPS가 다룬 류의 문제를 "고전적 인공지능 계획 문제"라고 한다[2].

고전적 인공지능 계획 문제를 표현하는 방법으로 STRIPS[3] 방법이 오래도록 사용되어 왔다. STRIPS 방법에서는 문제가 술어 논리(predicate logic)로 표현된다. 로봇의 동작은 동작이 적용되려면 충족되어져야 하는 조건(precondition)을 나타내는 일군의 술어(predicate), 동작을 수행하면 부정이 되는 일군의 술어들(delete condition), 그리고 동작 수행의 결과 참으로 되어지는 일군의 술어들(add condition)로 정의된다. 로봇이 작업을 수행하는 환경도 역시 술어로 표현된다.

본 논문은 STRIPS 방법으로 표현된 인공지능

계획 문제를 아일랜드 탐색과 휴리스틱 탐색을 혼용하여 분산시스템에서 푸는 방법을 제안한다. 아일랜드 탐색을 적용하려면 초기상태에서 목적상태로 변환하는 도중 꼭 거쳐야하는 아일랜드를 제공해야 한다. 그러나 그러한 아일랜드를 찾는 것은 불가능한 일이다. 그러므로, 선취관계를 이용하여 적당한 아일랜드를 임의로 만든다. 로봇 계획 문제의 목적 상태를 구성하는 부분목적 사이에 어떤 부분 목적이 반드시 다른 어떤 부분 목적 보다 먼저 성취되어져야 하는 관계를 선취관계라 한다.

아일랜드를 프로세서 수만큼 생성하여, 각 프로세서에 하나의 아일랜드를 원래의 계획 문제와 함께 할당한다. 각 프로세서는 초기상태에서 아일랜드까지 가는 문제를 휴리스틱 방법으로 풀고, 아일랜드에서 목적 상태로 도달하는 문제를 또한 휴리스틱 방법으로 풀어 결합함으로써 원래의 문제에 대한 해를 구하여 주 프로세서에게 되돌려 준다. 주 프로세서는 되돌아 온 해 중에서 가장 효율적인 해를 선택하여 최적해를 찾는다

2. 걸림관계

인공지능 로봇 계획 문제는 로봇이 작업 환경을 목적 상태로 변화시키기 위하여 수행해야 할 일련의 동작을 찾는 것이다. 작업 환경의 상태는 논리적 and로 연결된 일군의 술어로 묘사될 수 있으며, 로봇의 동작도 술어를 사용하여 정의한다. 로봇이 작업하는 환경의 초기 상태를 I라 하고 목적 상태를 G, 로봇이 취할 수 있는 동작들의 집합을 A라 하면 인공지능 로봇 계획 문제 AIP=<I, A, G>이다. A의 임의의 원소 동작 a는 precondition, delete condition, add condition으로 구성되며 이 condition들은 작업 환경의 상태를 묘사하는 술어를 원소로 하는 집합들이다. 목적 상태 G를 구성하는 술어를 g_1, g_2, ...로 표기하며 이들을 각각 부분목적이라고 한다.

동시에 성취할 수 있는 동작이 제공되지 않은 어떤 두 개의 부분 목적 g_1과 g_2를 독립적 부분목적이라 하며, g_1과 g_2를 모두 포함하는 동작의 addition condition이 존재하지 않으면 이들은 독립적 부분목적이다.

정의 1: G의 원소 g_1과 g_2에 대하여 이들 둘을 모두 add condition으로 포함하는 동작이 존재하지 않을 때 서로 독립적이라 한다.

동작의 precondition은 동작을 수행하려면 만족되어야 하는 술어의 집합이다. 동작에 대한 precondition과 같은 개념을 부분목적에 적용하여, 부분목적 g_1을 성취하기 위한 최소한의 조건을 나타내는 술어의 집합을 g_1의 "조건"이라 한다. 주어진 부분 목적에 대한 조건은 유일하지 않다. 부분목적 g_1을 성취하려면 우선 g_1을 add condition의 원소로 하는 동작이 존재하여야 한다. 이러한 동작은 한 개 이상일 수 있으며, 그 중 한 개만 수행하면 g_1을 성취한다. 이들 동작들의 precondition이 각각 g_1의 조건이다. 이 precondition의 원소인 술어 p에 대하여 다시 이것을 add condition에 포함하는 동작을 찾고 각 동작의 precondition을 다시 p로 하는 작업을 반복함으로써 더욱 기본적인 조건을 찾을 수 있다. 이러한 작업의 진행 과정을 가시화 한 그림을 "조건도"라 한다. 조건도는 부분목적을 root로 하고, 이를 add condition에 포함하는 동작들을 child로 한다. 즉, 조건도의 level 1은 부분목적 g_1으로, level 2는 g_1을 add condition에 포함하는 동작들로, 그리고 level 3는 이들 동작들의 precondition들로, level 4는 동작들로 ... 구성된다. 본 논문에서 조건도를 도입하는 목적은 이를 이용하여 두 개의 부분목적 사이에 어떤 부분 목적의 성취가 다른 부분목적 성취에 방해가 되는 간섭 현상을 찾는데 있다.

독립적인 두 개의 부분 목적 g_1과 g_2에 대하여, g_1을 성취한 상태가 다른 부분 목적(g_2)을 성취하는 동작의 조건을 부정시키면 두 부분 목적간에 모종의 간섭 관계가 존재한다. 본 논문에서는 이러한 관계를 걸림관계라 정의한다.

정의 2: 독립적인 두 개의 부분 목적 g_1과 g_2에 대하여, g_1을 성취한 상태가 g_2의 조건을 부정(negate)하면 걸림(g_1, g_2)라 한다.

걸림(g_1, g_2) 관계가 성립하는 목적 상태가 주어질 때, g_1을 g_2 보다 먼저 성취하는 계획은 추후 g_2를 성취하기 위하여 g_1 성취를 취소(undo)하는 동작을 수행하여 g_2의 precondition을 충족시킨 후, 다시 g_1을 재성취 한다. 그러므로 최적계획은 g_2를 먼저 성취하고 g_1을 나중에 성취한다.

사실 1: 걸림(g_1, g_2)일 때 최적 계획은 g_2를 성취한 후 g_1을 성취한다.

<증명> 이 사실의 증명은 [4]에 소개된 다음과 같은 정리를 사용한다. "명제 p가 어떤 상태 s에서 참이기 위한 필요 충분 조건은 다음 두가지 조건을 동시에 만족하는 것이다: 1) s에 선행하는 어떤 상태 t에서 p가 참이다. 그리고 2) s에 이르기 전에 수행되는 모든 동작 C에 대하여 그리고 p=q가 될

수 있는 C가 부정하는 q에 대하여 C와 s 사이에 p=q 일 때 마다 p=r인 r을 참으로 하여 주는 동작 W가 수행되어야 한다."

이 정리에 의하면 논리적 and로 연결된 독립적인 두 부분목적 g_1, g_2로 형성된 명제 ($g_1 \wedge g_2$)를 만족하는 상태 s에 도달하는 계획은, 초기 상태에서 이들이 이미 성취되어 있지 않은 한, g_1을 add condition에 포함하는 동작과 g_2를 add condition에 포함하는 두 동작을 포함한다. 더 나아가서 결림(g_1, g_2)이므로, ($g_1 \wedge g_2$)를 만족하는 최초의 상태 s에 도달하는 계획은 add condition에 g_1을 포함하는 동작으로 끝난다(왜냐하면 g_1이 성취된 상태에서는 g_2를 성취하는 동작을 수행할 수 없으므로). 그러므로 사실 1임.

사실 1은 결림(g_1, g_2)이면 계획은 g_2를 g_1보다 먼저 성취한다는 것을 서술한다. 이를 선취(g_2, g_1)이라 한다. 선취관계는 추이관계임을 쉽게 알 수 있다. 본 논문이 제안하는 바는 임의로 생성된 아일랜드에 대하여 선취관계에 어긋나는 아일랜드를 제거하는 것이다.

선취관계를 파악하려면 먼저 결림관계를 파악하여야 한다. 결림(g_1, g_2)를 파악하는 알고리즘은 표1과 같다. 이 알고리즘의 입력은 g_1을 성취하면 참이 되는 명제들의 집합인 상태1과 g_2이다.

<표 1> 결림관계를 파악하는 알고리즘

입력: 상태1: 부분목적 g_1이 성취되었을 때 필연적으로 참이되는 명제의 집합.

g_2

변수: p-list: 술어의 집합;

```
/* procedure 시작 */
p-list = {g2}; /* p-list를 초기화 */
while p-list에 marked 되지 않은 원소가 존재하는 동안
  p-list의 mark되지 않은 임의의 원소를 p로 함.
  p에 mark함;
  unachievable(p)이면 결림(g1, g2)임.  return(1);
  for (p를 add condition으로 하는 모든 동작 a에 대하여)
    imperformable(a) then continue;
    a를 p의 child로 함.
    for (a의 precondition q에 대하여)
      q가 p-list의 원소이면 continue;
      not(possibly-inconsistent(q, 상태1)) 이면 continue;
      p-list = p-list ∪ {q};
    endfor
  endfor
endwhile
return(2)  /* 결림관계라고 밝히지 못함, end of algorithm */

unachievable(p)   /* 함수 시작 */
for (p를 add condition으로 하는 모든 동작 a에 대하여)
    imperformable(a) then continue
                  else return(false);
return(true)   /* The end of unachievable */

imperformable(a)      /* 함수 시작 */
for (a의 precondition q에 대하여)
  inconsistent(q, 상태1) then return(true);
                else  continue;
return(false);      /* end of imperformable */

possibly-inconsistent(q, 상태1) /* 함수 시작 */
  q에 변수가 사용되지 않고, 사용된 상수가 모두 상태1에 사용
      되면 return(true);
  return(false);
/* possibly-not-inconsistant(q, 상태1) 의 끝 */
```

결림(g_1, g_2)이면 선취(g_2, g_1)이므로 두 부분목적간의 결림관계를 파악하면 이들간의 선취관계도 역시 파악한 것이 된다. 다음 단계에는 선취관계가 추이관계임을 고려하여 부분목적간의 모든 선취관계를 파악한다.

표1에 보이는 알고리즘은 반드시 종료한다. 어떤 술어 p에 대하여 이것이 unachievable이면 결림(g_1, g_2)라 하고 종료하며, 그렇지 않으면 p-list의 원소들이 모두 marked 되었을 때 결림관계임을 밝히지 못하였다는 결론을 내고 종료한다. 그런데 p-list에 첨가되는 원소는 possibly-inconsistent(q, 상태1)인 명제 q라야 한다. 함수 possibly-inconsistent(q, 상태1)을 살펴보면 q에 변수가 사용되지 않고, 사용된 상수가 모두 상태1에 사용되어 있을 때 참 값을 함수값으로 한다. 상태1에 사용되는 상수의 수를 n, precondition에 사용되는 술어 중 인수의 수가 1인 술어의 수를 m1, 인수의 수가 2인 술어의 수를 m2, 인수의 수가 3인 술어의 수를 m3, ... 라고 하면, possibly-inconsistent(q, 상태1)을 참으로 하는 q의 가지 수는 $m1*n + m2*n^2 + m3*n^3 + \ldots$ 이다. 더구나 p-list는 집합이고, 따라서 원소가 유일하므로 표1의 알고리즘이 p-list에 추가하는 명제의 수는 유한하다. 예를 들어 어떤 문제를 정의하는데 사용된 술어의 수가 m3 개이고 이들의 인수의 수가 모두 3이라 하며, 상태1에 사용된 상수의 수가 n이라 하면 p-list의 원소의 수는 최대 $m3*n^3$이다. 로봇이 수행할 수 있는 동작의 수가 k라 하면 표1 알고리즘의 while loop내의 "for (p를 add condition으로 하는

모든 동작 a에 대하여)" 문장은 최대 k번 반복될 수 있다. 다른 for loop나, 함수 unachievable과 imperformable도 동작의 수나 명제의 수만큼만 명령어를 수행함으로 결국 표1의 알고리즘은 복잡도가 다항식임을 알 수 있다.

표1에 보이는 알고리즘이 1을 return하면 걸림 (g_1, g_2)이다. 그 이유는 다음과 같다. 본 알고리즘이 1을 return하는 필요-충분 조건은 g_2를 성취하려면 반드시 참이 되어야 하는 어떤 명제가 unachievable 한 것이다. 그런데 unachievable(p)는 p를 성취할 수 있는 모든 동작이 imperformable할 때 참이고, 어떤 동작 a는 상태1과 inconsistent한 precondition을 포함 할 때 imperformable하므로 결국 g_1이 참인 상태(상태1)에서는 g_2를 성취할 수 없을 때 본 알고리즘은 1을 return한다.

본 알고리즘이 모든 걸림관계를 다 찾는 것은 아니다. 함수 possibly-inconsistent는 "q에 변수가 사용되지 않고, 사용된 상수가 모두 상태1에 사용되면 return(true)"를 하는데 사실은 q에 변수가 사용되더라도 이 변수에 어떤 상수가 대입되느냐에 따라 inconsistent할 수도 있고, 또 상태1에 사용되지 않은 상수도 사용되는 동작에 어떤 상수를 대입하느냐에 따라 inconsistent한 상황이 발생 할 수도 있다. 그러나 본 알고리즘은 이러한 경우를 모두 배제하였으므로 모든 걸림관계를 다 찾지는 못 한다. 즉, 본 알고리즘은 sound하지만 complete하지는 못하다.

3. 혼합탐색 방법

본 논문이 제안하는 혼합탐색 방법은 아일랜드 탐색과 휴리스틱 탐색을 분산 환경에서 수행하는 것이다. 혼합탐색 방법은 다음과 같다:

1) 선취관계를 이용하여, 주어진 계획문제에 대한 아일랜드를 생성한다.

2) 생성된 아일랜드와 계획 문제를 임의의 프로세서에 할당한다.

3) 각 프로세서는 휴리스틱 탐색으로 계획을 탐색하여 주 프로세서에게 반환한다.

4) 주 프로세서는 반환된 계획들을 비교하여 최적해를 찾는다.

<예 1> 아래에 보이는 상자세계 문제의 Sussman의 anomaly 문제에 혼합탐색 방법 적용. 상자세계 문제는 precondition과 delete condition이 동일한 특징이 있으므로 표2에는 P & D라 하여 함께 표기한다.

<표 2> Sussman의 anomaly 문제

초기 상태 I: CLEAR(C), CLEAR(B), ONTABLE(A), ONTABLE(B), ON(C, A).

목적 상태 G: ON(A, B), ON(B, C)

상자세계 문제의 로봇 동작

pickup(x)
  P & D: ONTABLE(x), CLEAR(x), HANDEMPTY
  A: HOLDING(x)

putdown(x)
  P & D: HOLDING(x)
  A: ONTABLE(x), CLEAR(x), HANDEMPTY

stack(x, y)
  P & D: HOLDING(x), CLEAR(y)
  A: HANDEMPTY, ON(x, y), CLEAR(x)

unstack(x, y)
  P & D: HANDEMPTY, CLEAR(x), ON(x, y)
  A: HOLDING(x), CLEAR(y)

---

1) 걸림(ON(A, B), ON(B, C))이므로, 상태 ON(B, C)를 아일랜드로 생성.

2) 주어진 문제와 아일랜드를 프로세서 P에 할당.

3) P는 ON(B, C)를 성취한 다음, ON(A, B), ON(B, C)를 모두 성취하는 계획을 차례로 찾아 반환함.

4. 결론

본 논문은 아일랜드 탐색에 기반을 둔 계획자 설계 방법을 제안하였다. 아일랜드 탐색에서 성공의 관건은 적당한 아일랜드를 제공하는 것이다. 본 논문에는 선취관계를 고려하여 적당한 아일랜드를 찾는 방법이 소개되었다. 이를 위하여 선취관계를 파악하는 알고리즘을 제안하고, 제안된 알고리즘의 효율성을 분석하였다.

적당한 아일랜드를 사용하면 탐색 공간은 급격히 축소된다. 향후에는 초기상태와 아일랜드 사이의 아일랜드를 생성하는 방법에 대하여 연구함으로써 획기적으로 효율적인 인공지능 계획자를 구현하려고 한다.

참고문헌

[1] G.W. Ernst and A. Newell, "GPS: A Case Study in Generality and Problem Solving," Academic Press, New York, 1969.

[2] D. McDermott and J. Hendler, "Planning: What it is, What it could be, An introduction to the Special Issue on Planning and Scheduling," Artificial Intelligence 76 (1995) pp 1-16.

[3] R.E. Fikes and N.J. Nilsson, "Learning and Executing Generalized Robot Plans," Artificial Intelligence 3(4) 1972 pp349-371.

[4] D. Chapman, "Planning for Conjunctive Goals," Artificial Intelligence 32 (1987) pp. 333-377.