실시간 동적 프로그래밍에 기초한 확률 계획기의 설계 및 구현
Design and Implementation of a RTDP-based Probabilistic Planner

김현식, 김동현, 김인철

경기대학교 정보과학부 전자계산학과
경기도 수원시 영통구 이의동 산 94-6
Tel: +82-31-249-9670, Fax: +82-31-243-9673, E-mail: {advance7, saint23730, kic}@kyonggi.ac.kr

Abstract

전통적 계획방식은 결정적 효과를 가진 동작들로 이루어진 도메인을 다룬다. 따라서 전통적 계획기는 동작이 환경을 어떻게 변화시킬지 명확하게 예측할 수 있다. 그러나, 많은 실제 응용들에서는 불완전한 정보와 비-결정적 효과를 처리할 수 있는 계획방식을 요구한다. 확률적 계획방식은 확률적 효과를 가진 동작들을 포함함으로써 이러한 요구를 만족한다. 확률적 계획기는 일반적으로 목표상태에 도달하기 위한 하나의 행동정책을 찾아내며, 이는 (상태, 동작) 쌍들의 집합으로 표현된다. 그러나 확률적 효과를 포함시킴으로써 계획기들의 복잡도가 이전보다 증가 되었다. 본 논문에서는 효율적인 확률적 계획기의 설계와 구현에 대해 설명한다. 이 계획기는 표준 PPDDL 언어로 표현된 도메인 묘사를 입력으로 받아들이며, 실시간 동적 프로그래밍 알고리즘을 채용하고, 간략화한 문제로부터 추출된 휴리스틱 지식을 이용한다. 생성된 상태들과 행동정책을 효율적으로 저장하기 위해, 이 확률적 계획기는 해쉬 테이블을 이용한다.

Keywords:

Probabilistic Planner, PPDDL, Real-Time Dynamic Programming, Policy

1. 서론

결정적 효과를 가진 동작들로 이루어진 도메인을 다루는 전통적 계획방식은 동작이 환경을 어떻게 변화시킬 수 있는지 명확하게 예측할 수 있다. 그러나, 많은 실제 응용들에서는 불완전한 정보와 비-결정적 효과를 처리할 수 있는 계획방식을 요구하고 있으며, 이러한 불완전한 정보와 비-결정적 효과를 다루기 위한 조건부 계획생성이나 확률 계획생성과 같은 다양한 연구가 진행되고 있다.

본 논문에서는 이러한 연구 분야 중에서 확률적 계획방식을 이용하여 이러한 문제를 해결하고자 하였다. 확률적 계획방식은 확률적 효과를 가진 동작들을 포함함으로써 이러한 요구를 만족하게 되는데, 일반적으로 목표상태에 도달하기 위한 하나의 행동정책을 찾아내며, 이는 (상태, 동작) 쌍들의 집합으로 표현된다. 그러나 확률적 효과를 포함시킴으로 인하여 계획기들의 복잡도가 이전보다 증가 되어 효율적인 상태 관리가 요구된다. 본 논문에서는 효율적인 확률 계획기의 설계와 구현에 대해 설명한다. 이 확률 계획기는 행동을 통한 효과(effect)를 충분히 관찰(fully observable)할 수 있으며 행동을 취했을 때 발생하는 효과가 비-결정적(non-deterministic)이며 행동으로 인한 효과의 발생 확률(probabilistic)을 가지고 있는 환경을 가정하며, 불완전한 정보와 비-결정적 효과의 표현을 지원하는 계획생성 언어인 표준 PPDDL 언어로 표현된 도메인 묘사를 입력으로 받아들인다. 또한 실시간 동적 프로그래밍 알고리즘을 채용하고, 간략화한 문제로부터 추출된 휴리스틱 지식을 이용한다. 생성된 상태들과 행동정책을 효율적으로 저장하기 위해, 이 확률적 계획기는 해쉬 테이블을 이용한다.

2. 확률 계획

2.1 문제 정의

일반적으로 확률적 계획생성 문제는 환경에 대하여 발생할 수 있는 모든 계획(상황)을 고안하는 계획 방법으로서, 발생할 수 있는 모든 상태에 대한 행동 정책을 생성함으로써 다양한 상황에 대처할 수 있게 한다. 이러한 확률적 계획생성 문제를 해결하기 위하여 본 논문에서 제안하고 있는 확률 계획기는 행동을 통한 효과(effect)를 충분히 관찰(fully observable)할 수 있으며 행동을 취했을 때 발생하는 효과가 비-결정적(non-deterministic)인 환경을 가정한다. 이러한 확률적 계획 문제는 다음과 같이 하나의 MDP 문제로 정의 가능하다.

- S: 상태들의 집합

- A(s)⊆A : 상태 S에서 적용 가능한 동작들의 집합
- Pa(s'|s) : 상태전이확률, 이때 s∈A이고 a∈A(s)
- c(a, s) : 동작비용, c(a, s) > 0
- G : 목표 상태들의 집합, G⊆S

상태 s'은 상태 s에서 동작 a를 취해서 얻어지는 결과로서, 얻어지는 상태를 사전에 예측(predictable)할 수는 없지만 환경에서 동작을 취했을 때 발생할 수 있는 상태들은 모두 알 수 있다(fully observable). 그러므로 상태 s'에서 동작 a'을 선택하는 것이 가능하며, 결과적으로 MDP문제는 결과가 동작들의 순서가 아닌 행동정책으로 표현된다. 행동정책은 (상태, 동작)의 쌍으로 표현되며, 행동정책 π는 초기상태 s에서 전이 가능한 모든 상태에 대하여 전이확률을 부여한다. 비용은 상태에서 전이되는 확률에 대한 가중치로 계산되며 목표상태 G는 비용과 효과가 주어지지 않는다. 최적의 솔루션을 위한 행동정책은 모든 상태들에 대하여 비용을 최소로 하는 것이다.

2.2 예제

예제를 통하여 확률 계획생성 문제가 가지는 표현 방법에 대하여 설명하고 특징을 알아보고자 하였다. 설명하고 있는 예제는 블록쌓기 예제로써, 임의로 놓여있는 블록을 원하는 모형으로 쌓는 예제이다.

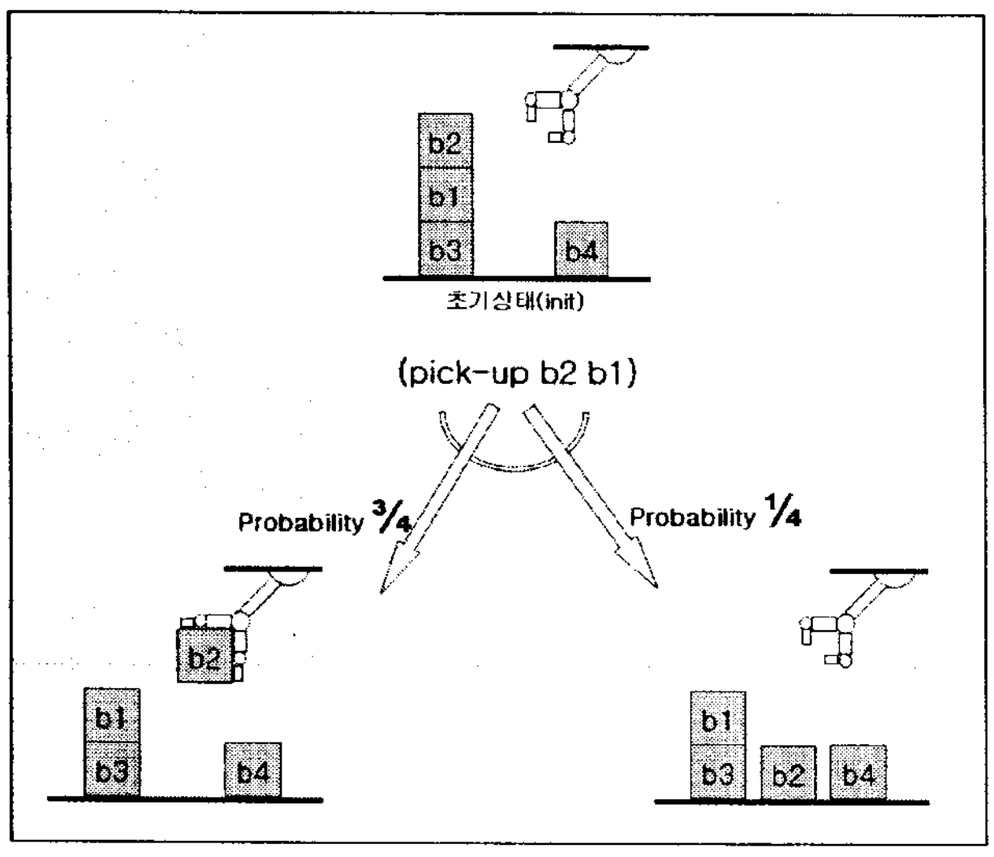

그림 1 - 비-결정적 동작을 적용한 경우

블록 도메인 문제는 주어진 초기상태에서 동작을 수행함으로써 목표상태로 나아가게 되는데, 비-결정적 동작이나 결정적 동작들을 통해 문제를 해결해 가게 된다. 도메인에서 사용 가능한 동작은 그림 1과 같은 비-결정성을 가지는 동작과 결정성을 가지는 동작으로 구분할 수 있으며, 비-결정적 동작으로는 블록 위에 블록이 있을 때 3/4확률로 블록을 들거나 1/4확률로 테이블에 내려놓는 pick-up과 같은 동작이 있다. 그림 1은 주어진 상태에서 비-결정적 동작 중에 하나인 pick-up을 적용했을 때 pick-up 동작을 통해 초기상태가 각각의 전이될 수 있는 상태를 확률과 함께 그림으로 표현한 예로서 비-결정적 동작이 상태에 어떠한 영향을 주게 되는지 쉽게 알 수 있다. 이러한 비-결정적 동작과 결정적 동작이 혼합된 환경에서 확률 계획기를 이용하여 주어진 문제를 해결해 가게 된다.

예제로 사용된 도메인은 pick-up과 함께 여러 개의 비-결정적 동작과 결정적 동작들로 이루어져 있다. 비-결정적 동작으로는 테이블 위의 블록을 들거나 동작을 실패하는 pick-up-from-table, 들고 있는 블록을 블록 위에 내려 놓거나 테이블에 내려 놓는 put-on-block, 블록들이 쌓여있는 경우, 중간 블록을 들어올리거나 실패하는 pick-tower, 쌓여있는 블록을 잡은 상태에서 블록 위에 내려놓거나 테이블에 놓는 put-tower-on-block이 있다. 결정성을 가지는 동작으로는 put-tower-down과 put-down 동작이 있으며 결정성을 가지는 동작의 확률은 1로서 한 상태로만 변화된다.

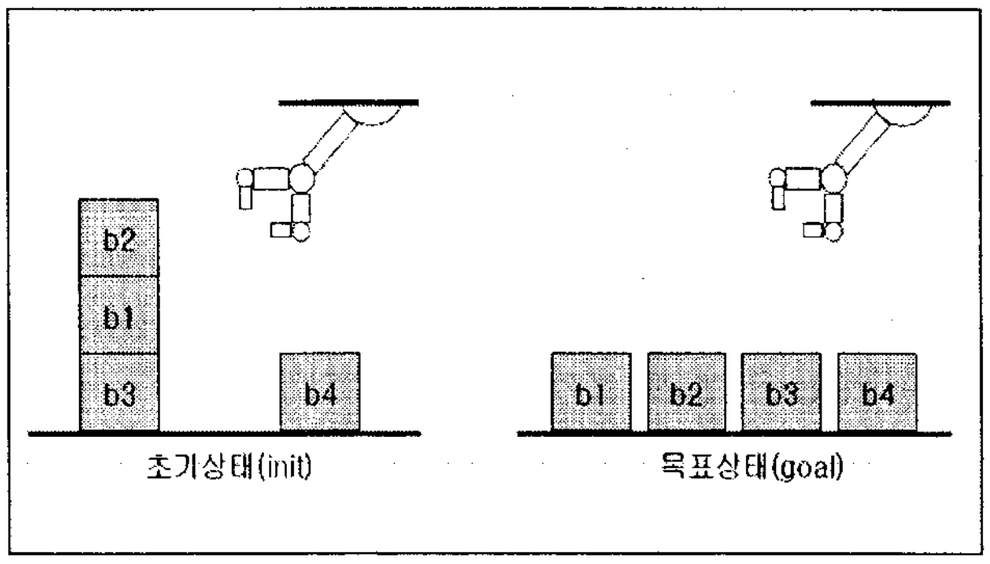

그림 2 - 계획문제의 초기상태와 목표상태

예제로 사용된 블록 도메인의 초기상태와 목표상태는 그림 2에서 보는 바와 같으며, 초기 상태는 4개의 블록이 있고 이 블록들 중에 3개가 쌓여있는 상태로 있다. 목표 상태는 이러한 블록들을 모두 쌓여 있지 않은 상태로 만드는 것이다.

2.3 문제 표현

PPDDL(Probabilistic Planning Domain Definition Language)[14]은 표준 계획지식영역언어인 PDDL(Planning Domain Definition Language)[5][3]을 확장한 형태로서 비-결정적인 상황과 확률(probability)을 표현할 수 있으며, 확률을 이용하여 환경의 불완전성과 행동의 비-결정성 표현을 함께 지원한다. 이러한 PPDDL은 확률을

이용하여 발생할 수 있는 상태를 확률 분포로 표현할 수 있으며, 행동의 비-결정적 상태 또한 확률 분포를 이용하여 표현할 수 있다.

표 1과 표 2는 위의 예제에서 설명한 내용을 PPDDL을 이용하여 계획기에서 입력 받을 수 있는 형태로 작성한 것이다. 표 1은 도메인을 기술한 내용으로 상태를 표현하기 위한 상태조건들과 환경에서 사용 가능한 동작들에 대한 내용이 기술되어 있으며, 표 2는 주어진 문제의 초기상태와 목표상태를 기술한 것이다.

표 1 - 도메인 표현

```
(define
(domain blocks-domain)
   (:requirements :conditional-effects
   :probabilistic-effects :equality :typing)
   (:types block)

(:predicates (holding ?b - block) (emptyhand)
             (on-table ?b - block) (on ?b1 ?b2 - block)
             (clear ?b - block))

(:action pick-up
:parameters (?b1 ?b2 - block)
:precondition (and (not (= ?b1 ?b2))
                (emptyhand) (clear ?b1) (on ?b1 ?b2))
:effect
(probabilistic 3/4 (and (holding ?b1) (clear ?b2)
                   (not (emptyhand)) (not (clear ?b1))
                   (not (on ?b1 ?b2)))
               1/4 (and (clear ?b2) (on-table ?b1)
                   (not (on ?b1 ?b2)))))

(:action pick-up-from-table
:parameters (?b - block)
:precondition (and (emptyhand) (clear ?b)
              (on-table ?b))
:effect (probabilistic 3/4 (and (holding ?b)
                           (not (emptyhand))
                           (not (on-table ?b)))))

(:action put-on-block
:parameters (?b1 ?b2 - block)
:precondition (and (holding ?b1) (clear ?b2))
:effect (probabilistic 3/4 (and (on ?b1 ?b2)
                           (emptyhand) (clear ?b1)
                           (not (holding ?b1))
                          (not (clear ?b2)))
                       1/4 (and (on-table ?b1) (emptyhand)
                           (clear ?b1) (not (holding ?b1)))))
……
```

표 2 - 계획문제 표현

```
(:init    (emptyhand) (on b1 b3) (on b2 b1)
          (on-table b3) (on-table b4) (clear b2)
          (clear b4))
(:goal    (and (emptyhand) (on-table b1) (on-table b2)
          (on-table b3) (on-table b4) (clear b1)
          (clear b2)  (clear b3) (clear b4)))
```

3. 확률 계획기의 설계

3.1 탐색 알고리즘

논문에서 제안하고 있는 확률 계획기인 JPCP는 동적 프로그래밍 알고리즘 중에서 하나인 LRTDP(Labeled Real-Time Dynamic Programming)[1][4] 알고리즘을 기반으로 한다. LRTDP 알고리즘은 RTDP(Real-Time Dynamic Programming) 알고리즘에 수렴표시(labeling)를 하여 탐색과정에서 수렴한 부분을 탐색하지 않음으로써 탐색 과정을 좀더 효율적으로 수행하는 알고리즘이다.

```
LRTDP(s : initialState, ϵ : float)
begin
        while ¬s.SOLVED do
                TRIAL(s, ϵ)
end

TRIAL(s : initialState, ϵ : float)
begin
        while ¬s.SOLVED do
                visitedStack:PUSH(s)
                if s.GOAL() then break
                s.GREEDYACTION(a)
                s.UPDATE()
                s = s.PICKNEXTSTATE(a)

        while visitedStack ≠ EMPTY do
                s = visitedStack.POP()
                if ¬LabeledCheck(s, ϵ) then break
end
```

그림 3 - LRTDP

LRTDP 알고리즘의 기본 탐색 과정은 RTDP 알고리즘과 매우 흡사하다. LRTDP 알고리즘은 RTDP 알고리즘과 같이, 가능한 행동들 중에 최적의 행동을 선택하고 이를 통해 현재 상태의 평가치를 갱신하는 과정을 거쳐 진행된다. 하지만 각각의 방문한 상태를 모두 기록하게 되며, 탐색 과정을 통해 목표에 도달하게 되면, 목표에 도달하는 동안 방문한 상태들의 평가치의 변경된 값이 주어진 수렴치(ϵ) 범위 내에서 변경되었는지 판단하고, 범위 내에 포함된 경우에 이 상태를 수렴했다고 판단하여 수렴여부를 표시(labeling)하게 된다. 이러한 과정을 통하여 초기상태가 수렴이 될 때까지 이를 반복하게 된다. 그림 3은 이러한 LRTDP 알고리즘의 수행과정을 나타내고 있다.

LRTDP는 이러한 탐색 과정을 통하여 최적은 아니지만, 사용자가 납득할 만한 최적에 가까운 솔루션을 구하게 된다. LRTDP 알고리즘은 수렴한 영역을 표시함으로써 이미 충분히 최적화 되었다고 판단되는 상태에 대해서 재 탐색을 하는 과정을 제외함으로써 탐색 시간을 줄일 수 있다. 또한

지속적인 평가치의 변동으로 인하여 다른 경로를 탐색하는 경우에도 기존에 탐색한 상태 중에서 수렴한 상태가 나타날 경우 더 이상 탐색을 수행하지 않게 된다. 이를 통하여 탐색 시간과 범위를 줄일 수 있게 된다. 이러한 수렴표시는 목표 상태부터 역방향으로 순서대로 진행되게 된다. 이렇게 진행된 수렴표시는 최종적으로 초기 상태에 대한 수렴표시를 함으로써 그 탐색을 종료하게 된다.

3.2 휴리스틱 측정

LRTDP 알고리즘의 경우 행동 비용(action cost)과 그 행동으로 인하여 발생할 수 있는 다음 상태의 평가치(state value)를 이용하여 현재 상태에 대한 평가치를 갱신하게 된다. LRTDP 알고리즘이 탐색을 처음 수행할 때, 각 상태에 대한 평가치가 존재하지 않기 때문에 이를 대신하기 위하여 휴리스틱을 이용한다. 이는 휴리스틱을 이용하지 않아서 각 상태에 대한 초기 평가치가 부여되지 않는 경우에 각 상태에 대한 평가치를 0 또는 일정한 값으로 고정하게 되는데, 이러한 경우에 가능한 모든 행동에 대한 평가치가 동일하게 측정되어 탐색 방법과 관계없이 무작위 탐색을 수행하는 것과 같은 결과를 가져오게 된다. 이러한 경우, LRTDP 알고리즘을 수행하게 되면 지속적인 반복을 통하여 문제 해결을 위한 솔루션을 구할 수 있기는 하지만 솔루션을 구하는데 까지 훨씬 더 많은 탐색 시간과 공간을 요구하게 된다. 이러한 탐색의 비효율성을 줄이기 위하여 휴리스틱을 이용한다.

JPCP는 단순화된 계획그래프를 이용한 휴리스틱 측정 방법과 현재상태와 목표상태간의 유사성을 이용한 휴리스틱 측정 방법을 제공한다.

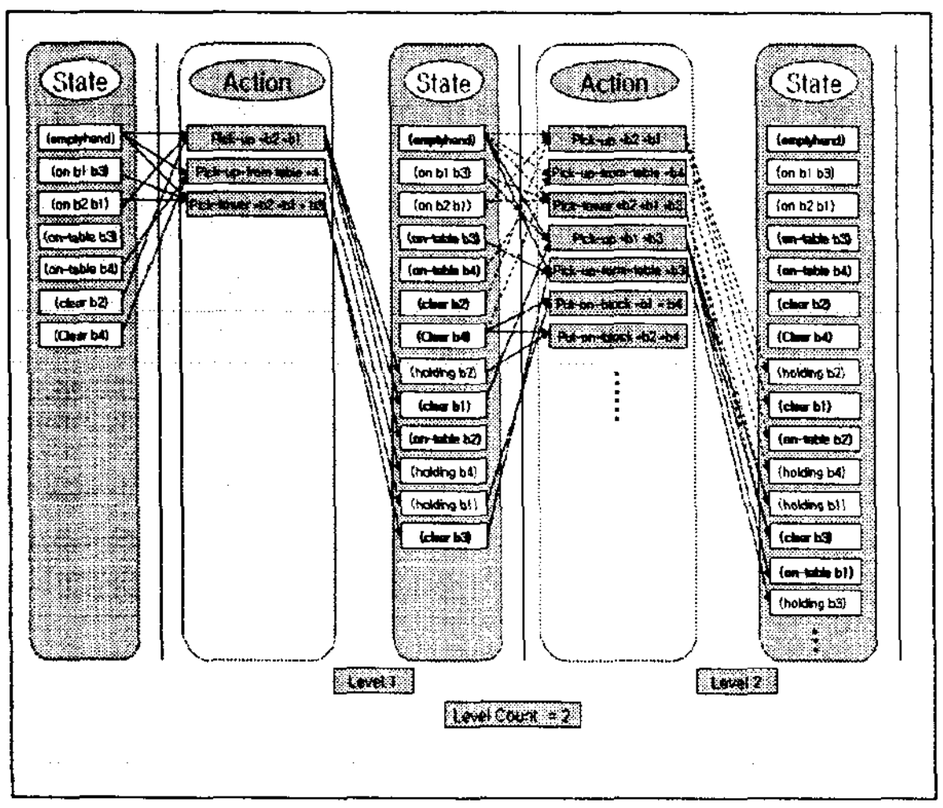

그림 4 - 단순화된 계획그래프

단순화된 계획그래프를 이용한 휴리스틱 측정 방법은 그림 4와 같이 목표 상태의 모든 조건을 만족 할 때까지 현재 상태에서 가능한 모든 동작을 적용하여 상태를 증가시키면서 목표 상태에 대한 조건을 모두 만족하는지 확인한다. 목표상태의 모든 조건을 만족하게 되면 그 진행 단계를 휴리스틱으로 이용한다. 단순화된 계획그래프는 발생하는 상태들 중에서 긍정적 상태만을 이용하며 비-결정적 행동의 경우 행동을 통해 발생하는 각각의 상태를 결정적 행동으로 구분한다.

계획그래프를 이용한 방법은 환경에서 이용 가능한 행동을 기반으로 함으로써 주어진 환경에서 좀 더 적합한 휴리스틱을 구할 수 있으며, 언제나 실제 탐색 거리보다 적은 값을 휴리스틱 값으로 구하게 된다. 또한 계산 방법이 단순하기 때문에 빠르게 휴리스틱 값을 구할 수 있다는 장점이 있다. 그러나 부정적 상태를 고려하지 않음으로써 적용 가능한 행동이 지속적으로 증가하고, 특정 도메인의 경우에 생성되는 상태가 매우 많으며 각 상태에 대한 휴리스틱 값의 차가 매우 적은 경우가 발생할 수 있다.

현재 상태와 목표 상태간의 유사성을 이용한 휴리스틱 측정 방법은 일반적으로 상태를 구성하는 조건이 상이한 경우일수록 더 많은 행위를 통하여 목표에 도달하게 된다는 가정을 기반으로 한다. 측정 방법은 현재 상태와 목표 상태의 조건을 비교하여 만족하지 않는 목표 상태의 조건의 수를 구하고 이 수를 휴리스틱 값으로 이용한다. 이 방법은 도메인에 정의된 행동을 이용하지 않고 상태 조건만을 비교하여 측정하기 때문에 환경 정보를 모두 이용하지 않는 문제점을 가지고 있다. 이 측정 방법은 예제에서 설명한 블록쌓기와 같이 특정한 도메인에 적합하다. 단순화된 계획그래프를 이용한 측정 방법의 경우, 측정된 휴리스틱 값의 차이가 매우 적어 비효율적인 경우가 있는데 이러한 특정 도메인의 경우에 이용하면 일반적으로 좀 더 탐색에 용이한 휴리스틱을 얻을 수 있다.

위에서 설명한 두 가지 휴리스틱 측정 방법은 기존 휴리스틱 측정 방법에 비하여 효율적이지 못하지만, 좀 더 빠르고 간단하게 그 휴리스틱 측정 결과를 얻을 수 있다. JPCP에서 휴리스틱 측정에 이러한 단순한 측정 방법을 이용하는 것은 적용하고 있는 알고리즘이 LRTDP 알고리즘이기 때문이다. 초기 평가치로 약한 휴리스틱이 주어진다 할지라도 LRTDP 알고리즘은 지속적으로 평가치를 갱신하여 수정하며 탐색을 진행하는 과정을 반복하기 때문에 결과적으로 일정한 값으로 수렴하게 된다. 또한 평가치가 주어지지 않는 경우보다 빠른 결과를 얻을 수 있으며, 일반적으로 초기 휴리스틱이 기존의 전통적인 계획기와 같이 강력한 휴리스틱이 아니라 할지라도 빠른 시간에 충분한 결과를 얻을 수 있다.

3.3 계획기 구조

위의 예제에서 설명한 것과 같은 확률계획 문제를 해결하기 위하여 제안하고 있는 확률 계획기인 JPCP의 구조는 그림 5와 같다.

JPCP는 도메인 파일과 문제 파일을 입력으로 받아들인다. 입력으로 받은 도메인 파일과 문제 파일은 Parser에서 분석하여 Domain, Problem, Predicate, Action과 행동의 비-결정성 부분을 관리하기 위한 ProbabilisticEffect로 구분하여 처리하며 Parser에서 분석된 내용을 바탕으로 문제를 풀기 위해 Grounding 과정이 수행되게 된다. 이 과정을 통해 환경 정보는 GPredDB, GState, GAction, GProbabilisticEffect로 구분되어 처리된다. StateInfo는 Parser와 Grounding을 통해 처리된 상태 정보를 관리하며 StateInfo를 통해 상태를 이용할 수 있다. Search 과정을 통해 목표를 만족하는 계획을 생성하게 되며, LRTDP 알고리즘을 이용하여 탐색 과정을 수행한다. Search 과정을 통해 계획생성 과정이 종료되면 PlanSyn에서 계획생성 과정을 수행하면서 생성된 상태들의 정보를 가지고 있는 ValueTable과 PolicyTable에 저장되어 있는 상태의 색인과 해당 상태의 정합과정을 진행하여 계획 결과를 출력하게 된다.

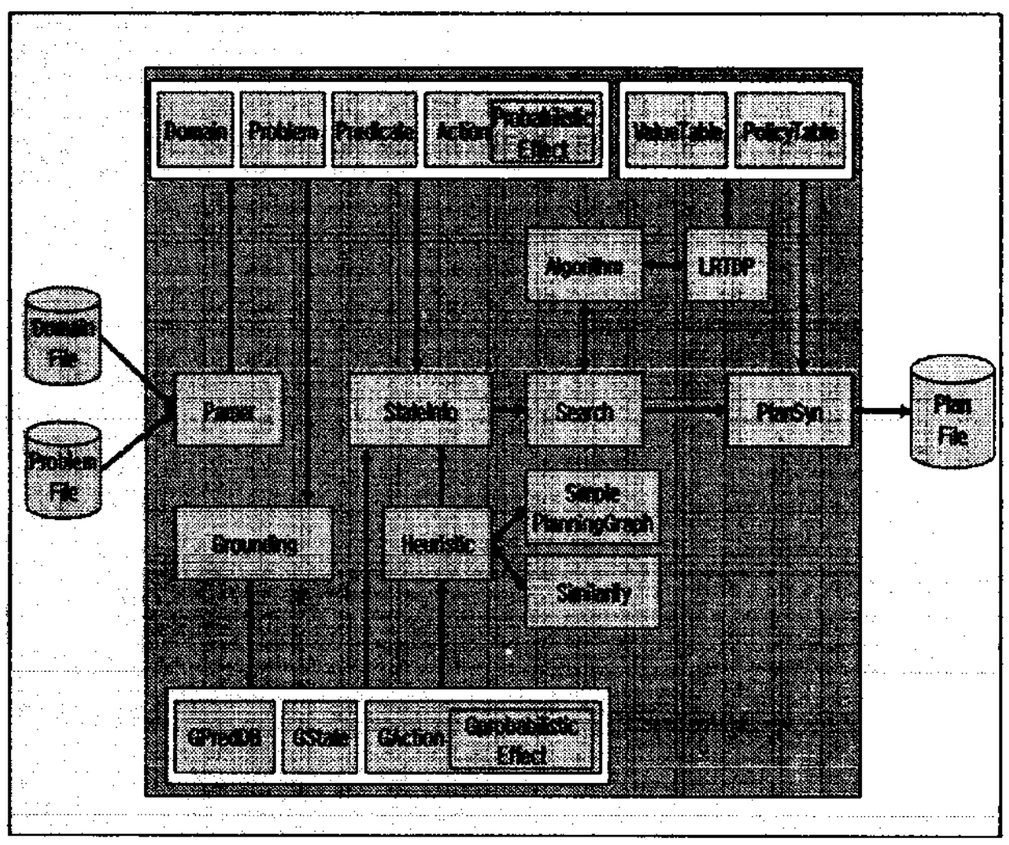

그림 5 - 확률 계획기 구조

4. 확률 계획기의 구현

4.1 상태와 동작 표현

JPCP는 상태를 표현하는 크기를 줄이기 위하여 내부적으로 상태를 구성하는 속성과 행동을 색인(index)으로 바꾸어 저장 관리하며, 이를 이용하여 그림 6, 그림 7과 같이 비-결정적 행동과 결정적 행동을 표현하게 된다.

```
pick-up*b2*b1*
(ID: 3
:precond (4) (26) (13)
:probability(0.75)
:effect +(1) +(25) -(4) -(26) -(13)
:probability(0.25)
:effect +(25) +(6) -(13)
```

그림 6 - 행동의 비-결정성 표현

```
candidate action 2 :
put-down*b2*
(ID: 29
:precond (1)
:probability(1.0)
:effect +(6) +(4) +(26) -(1)
```

그림 7 - 결정적 행동 표현

행동은 가질 수 있는 경우의 수를 확률(probability)과 효과(effect)의 쌍으로 표현한다. 각 상태는 그림 6, 그림 7과 같은 행동을 적용하여 다음 상태를 생성하게 되며 각 행동의 평가치(Q-value)를 측정하고 이를 이용하여 최적의 행동을 선택한다. 선택된 행동이 비-결정적인 경우 0에서 1사이의 난수(random number)를 발생시켜 그 값을 이용하여 각 상태의 확률과 비교하여 다음 상태를 결정하게 된다.

4.2 상태 관리

JPCP는 목표까지의 탐색을 진행하면서 상태에 대한 평가치를 갱신하게 되는데, 이러한 과정을 반복적으로 수행함으로써 목표까지의 솔루션을 구하게 된다. 이러한 과정을 통해 각 상태에 대한 평가치를 지속적으로 갱신하기 위해서는 각 상태에 대한 평가치를 지속적으로 저장, 관리할 필요성이 있다. 계획 생성 과정에서 많은 상태가 발생하게 되고, 이러한 상태들을 효율적으로 관리하고 이용하는 것이 계획기의 효율성에 직접적인 영향을 주게 된다. JPCP에는 이러한 탐색 과정에서 발생하는 상태들에 대한 정보를 해쉬테이블(hash table)을 이용하여 상태들에 대한 평가치 등의 정보를 저장 관리함으로써 빠르게 상태들에 대한 정보를 이용할 수 있게 하였다.

JPCP에서는 탐색과정에서 생성되는 모든 상태들에 대한 정보를 관리하는 Value_Table과 수렴한 상태 정보만을 추출하여 보관하는 Policy_Table을 이용하여 생성된 상태와 수렴한 상태를 관리한다. 상태를 구성하는 객체(Object)들의 색인(Index)을 정렬한 정보를 Value_Table과 Policy_Table의 키(Key)로 이용한다. 여기서 인덱스를

그대로 정렬하여 쓰게 되면 탐색이 종료한 후에 결과를 출력할 때, 숫자로 구성되어 있었기 때문에 상태로의 재변환이 어렵게 되는 문제를 갖게 된다. 이를 해결하기 위하여 인덱스를 정렬하여 키로 변환할 때, 인덱스 간에 구분자를 입력하여 키로 변환한다. 이러한 방법을 이용하여 상태를 구성하는 오브젝트를 키로 변환하여 사용하면 동일한 상태에 대한 정보가 해쉬테이블에 중복적으로 저장되거나, 서로 다른 상태가 동일한 키를 이용하는 경우와 같은 문제를 해결할 수 있다.

Value_table에는 각 상태에 대한 평가치와 그 상태의 수렴여부(True/False)를 쌍으로 저장하며 Policy_Table은 Value_Table에 저장된 상태 정보의 평가치가 수렴되게 되면 Value_Table의 수렴여부(flag) 정보가 false에서 true로 바뀌면서 그 때 선택된 행동을 Policy_Table에 저장하게 된다. 반복적인 탐색을 수행하면서 지속적으로 생성되는 상태가 기존 생성된 상태인지 아니면 새롭게 생성된 상태인지를 Value_Table을 통하여 검사하고 저장된 평가치를 이용하게 되며, 그 상태의 수렴 여부를 함께 확인하여 하위 상태를 지속적으로 탐색할 것인지를 판단하게 된다.

4.3 행동선택

한 상태는 그 해당 상태에서 취할 수 있는 가능한 행동들(applicable actions)을 이용하여 각 행동을 적용했을 때의 얻을 수 있는 평가치를 구하게 되고, 그 중에서 가장 평가치가 좋은 행동을 선택하게 된다. 한 상태는 다양한 행동을 취할 수 있으며 각 행동을 통한 평가치를 계산하여 보면 동일한 평가치를 가지게 되는 행동들이 다수 존재할 수 있는데, 이러한 동일한 평가치를 가지는 행동이 다수 존재하는 경우, 이중에서 한 행동을 선택하는 과정이 필요하다. 일반적으로 이러한 경우에 가능한 행동들 가운데 평가치가 동일한 행동과 그 행동에 대한 평가치를 모두 기억하고 있다가 그 중에서 하나를 임의로 선택하는 방법을 취하게 되는데, 이 경우에 평가치가 동일한 행동과 그에 대한 평가치를 모두 저장하고 있어야 하는 문제점이 발생한다.

JPCP에서는 한 상태에 대하여 동일한 평가치를 가지는 행동들을 모두 저장하는 이러한 경우의 문제점을 해결하기 위하여 등장하는 동일한 평가치를 가지는 행동들의 수에 따라 행동을 선택하는 확률을 변경시키는 방법을 이용함으로써 동일한 평가치를 가지는 행동이 다수가 등장하더라도 이를 모두 보관할 필요 없이 동일한 확률로 최적의 평가치를 가지는 행동을 선택할 수 있도록 하였다.

4.4 비-결정적 상태 전이

현재 상태에서 최적의 행동을 선택하게 되면 선택된 행동을 이용하여 다음 상태로 진행하게 된다. 그러나 선택된 행동이 하나의 다음 상태를 갖는 결정적 행동일 수도 있지만, 여러 개의 다음 상태를 가지는 비-결정적 행동이 대부분의 경우를 차지하게 된다. 이처럼 비-결정적인 행동의 경우에는 여러 개의 다음 상태를 가지게 되고, 각 상태는 (발생 확률, 상태)의 쌍으로 구성된다. 이러한 비-결정적 행동의 경우에 다음 상태를 선택하는 과정이 필요한데, 일반적으로는 발생할 수 있는 모든 상태에 대한 확률을 구하고 각 확률을 선택영역에 배정하고 난수(random number)를 발생시켜 다음 상태를 선택하는 방법을 이용한다.

5. 평가

JPCP를 이용하여 표 1과 표 2에서 설명한 예제를 풀어보고 결과를 확인해 보았다. 계획기는 목표를 만족하기 위해 탐색을 반복함으로써 그림 8에서 보는 것과 같이 상태들의 평가치가 지속적으로 갱신되게 되며 최적의 행동정책을 찾아가게 된다. 그림 8은 행동에 대한 평가치가 변경되어 가는 과정과 그 행동을 통해 생성되는 상태들을 확인하기 위하여 행동명과 전제조건, 확률, 효과와 이를 통해 생성되는 상태와 행동을 수행할 때의 평가치를 모두 보이고 있다. 이와 같이 JPCP는 반복되는 탐색을 통해 상태의 평가치를 측정하고 갱신하면서 목표를 만족하기 위한 행동 정책을 찾아가게 된다.

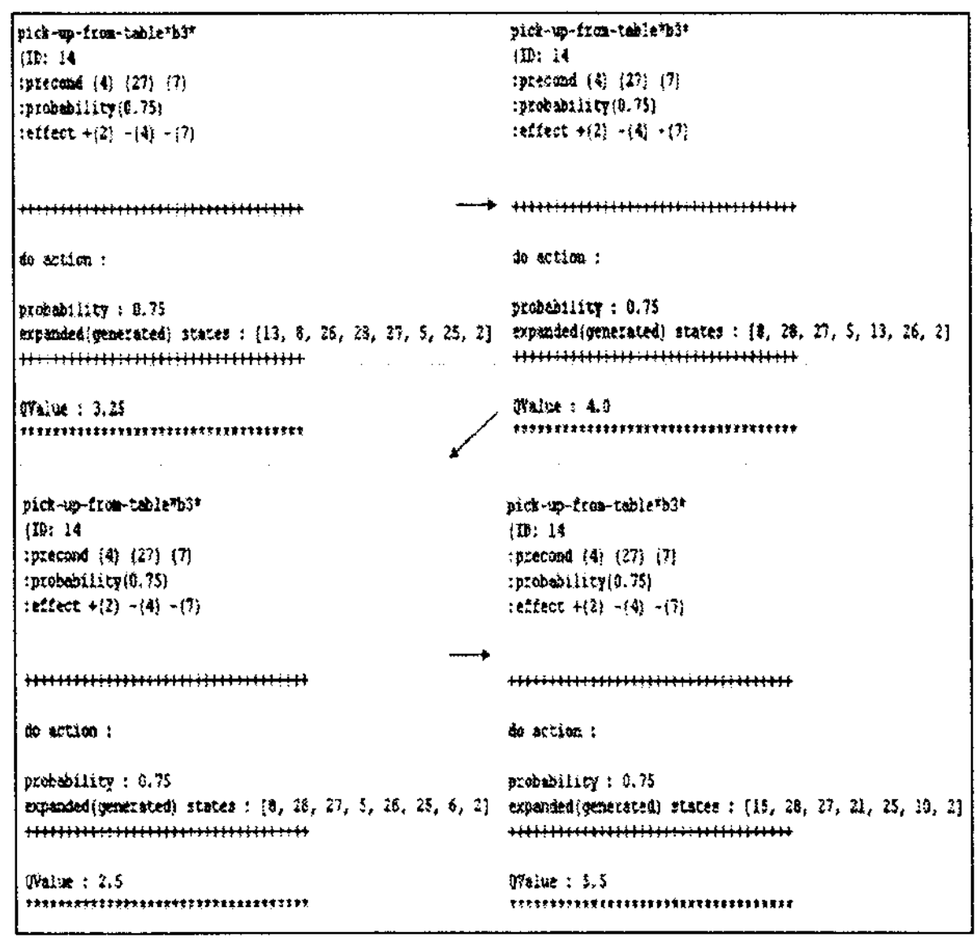

그림 8 - 행동을 통한 상태의 평가치 변동

표 3은 이러한 탐색 과정을 통해 얻어진 결과로서 목표에 도달하기 위한 행동정책을 나타내고 있다. 행동정책은 (상태, 행동)의 쌍으로

표현되며 표 3은 탐색을 통해 (상태, 행동)의 쌍으로 얻어진 결과를 상태조건과 동작으로 풀어서 기술한 내용이다. 그림 9는 이러한 결과를 알기 쉽게 트리를 이용하여 표현한 그림이다.

표 3 - 행동정책표

상태	상태조건	동작
S_0	(emptyhand) (on b1 b3) (on b2 b1) (on-table b3) (on-table b4) (clear b2) (clear b4)	(pick-up b2 b1)
S_1	(holding b2) (clear b1) (on b1 b3) (on-table b3) (on-table b4) (clear b4)	Put-down b2
S_2	(clear b1) (on-table b2) (emptyhand) (on b1 b3) (on-table b3) (on-table b4) (clear b2) (clear b4)	(pick-up b1 b3)
S_3	(holding b1) (on-table b3) (clear b3) (on-table b4) (on-table b2) (clear b2) (clear b4)	Put-down b1
S_g	(clear b3) (on-table b1) (clear b1) (on-table b2) (emptyhand) (clear b2) (on-table b3) (on-table b4) (clear b4)	<GOAL>

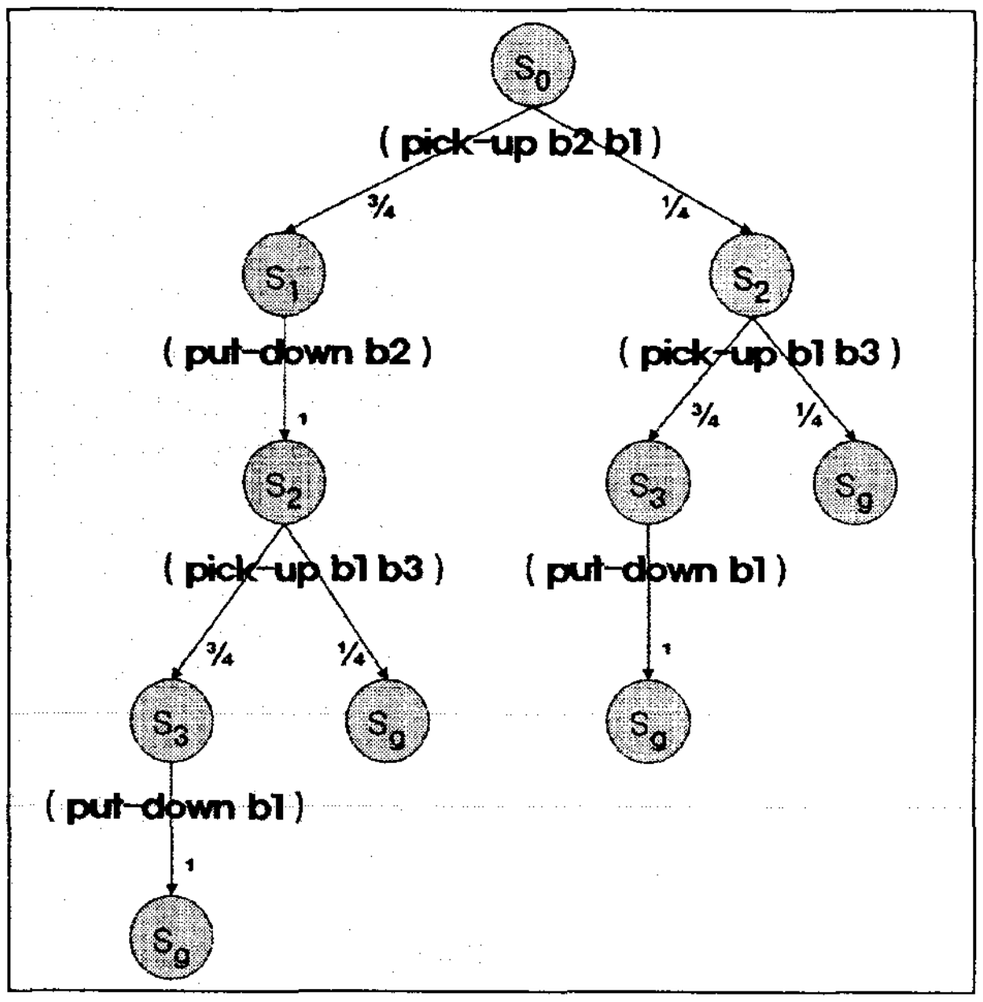

그림 9 - 트리를 이용한 행동정책 표현

6. 관련 연구

휴리스틱 기반의 탐색 방법인 LAO*[2] 알고리즘은 전통적인 알고리즘인 AO*의 확장된 형태로서 전체 상태 공간을 평가하지 않고 최적의 솔루션을 구하는 휴리스틱 탐색 알고리즘으로 루프 솔루션을 포함하고 있다. VI(Value Iteration)과 PI(Policy Iteration)과 같은 MDP를 위한 동적 프로그래밍 알고리즘을 대체할 있으며, and/or 그래프를 통해 직선으로 순차적인 한정적인 범위의 MDP를 표현한다. 가능한 상태들에 대하여 휴리스틱 함수를 사용하여 상태들을 평가하고, 그 다음에 최적의 부분 솔루션에 의해 방문한 상태들에 동적 프로그래밍을 수행하고, 그들의 값을 갱신하여 부분 솔루션을 개선한다. 이러한 과정을 완전한 솔루션이 발견될 때까지 반복하여 일어난다. 그러나 이러한 솔루션이 최적의 솔루션은 아닐 수 있다. LAO* 알고리즘은 한번의 탐색과정만을 수행하며, 수행과정에서 상태의 평가치가 수렴할 때까지 반복적인 평가치 갱신을 한다. LRTDP 알고리즘과 비교해 볼 때, LRTDP와는 달리 한번의 탐색과정만을 수행하지만 한번의 탐색과정이 매우 느리다는 특징을 가지고 있다.

VI(Value Iteration)[1] 알고리즘은 값을 지속적으로 갱신함으로써 그 결과를 찾아내는 방법으로서, 초기상태와 사용 가능한 오퍼레이터(operator)를 가지고 도달할 수 있는 상태 중에서 최선의 상태를 결정한다. 이러한 VI는 전체 상태를 평가함으로 인하여 효율성이 떨어지는 단점을 지니고 있으며, 전체 상태에 대한 평가치 갱신을 목표까지 도달한 후에 목표까지의 경로 전체를 동시에 갱신한다는 특징을 가지고 있다.

HDP(Heuristic Dynamic Programming)[1] 알고리즘은 사전에 모든 최적 계획(optimal plan)을 생성하는 방법이다. 문제를 해결하는 모든 행동을 선택하며, 성공한 다음 상태들을 바탕으로 갱신한다. HDP는 이러한 과정을 통하여 초기상태로부터 도달 가능한 모든 상태에 대하여 최적의 계획을 알게 된다. 초기 상태로부터 부분적인 최적 정책(partial optimal policy)을 이용하며 깊이 우선 탐색을 실행하면서 도달 가능한 상태에 대하여 값을 갱신하면서 진행된다. 탐색은 일치하지 않는 상태가 발견되지 않을 때까지 진행된다.

RTDP[1][4] 알고리즘은 초기상태로부터 시작하여 가능한 행동을 통하여 도달 가능한 상태들 중에서 목표에 가까운 최적의 행동을 선택하게 된다. 행동을 선택하면 그 행동을 수행하는 비용(action cost)과 상태에 대한 평가치(state value)를 바탕으로 현재 상태의 값을 갱신하고 다음 상태로 진행하게 된다. 이러한 지속적인 상태의 평가치 갱신 작업을 통하여 목표에 도달하면 목표까지 도달하는데 방문한 상태의 평가치의 변경된 값을 측정하여 오차범위 내에 수렴하였는지 평가하게 된다. 방문한 상태가 모두 오차범위 내에 들어오게 되어 전체 상태가 수렴하게 되면 알고리즘 수행을 중지하고 해당 결과를 되돌려 주게 된다. LRTDP는 이러한 RTDP를 개선한 방법으로서 RTDP의 경우 수렴한 상태를 지속적으로 탐색하고 평가함으로써 LRTDP에 비하여 비효율적인 측면을 가지고 있다.

7. 결론

지금까지 확률적 계획생성의 문제 해결 방법을 알아보고, 최근 요구되고 있는 불완전한 정보와 비-결정적 효과를 해결할 수 있는 확률 계획기인 JPCP를 제안하였다. 본 논문에서 제안하는 JPCP는 최근 계획 생성에서 요구되고 있는 불완전한 정보와 비-결정적 효과에 라는 현실 문제 반영 문제를 확률을 이용하여 표현하고 해결을 하고자 하였으며, 동적 프로그래밍 알고리즘에 기초하여 좀 더 빠르고 정확한 계획생성 결과를 얻고자 하였다. 향후 JPCP는 계획생성 과정에서 발생하는 상태 증가 문제를 좀 더 효율적으로 다루고 해결하기 위한 상태 표현 및 관리에 대한 연구를 진행할 예정이다. 이러한 연구는 계획생성에 있어서의 현실 문제를 반영하면서도 표현의 상태 수를 줄임으로써 좀 더 좋은 계획기의 성능을 얻을 수 있게 할 것으로 판단된다.

References

[1] B. Bonet and H. Geffner. (2005). mGPT: a Probabilistic Planner based on Heuristic Search. Journal of AI Research, Vol 24, pp 933-944.

[2] Eric A. Hansen and Shlomo Zilberstein. (2001). LAO* : A heuristic search algorithm that finds solutions with loops. Artificial Intelligence.

[3] Fox, Maria and Derek Long. (2003). PDDL2.1: An extension to PDDL for expressing temporal planning domains. Journal of Artificial Intelligence research. 20: 61-124.

[4] Blai Bonet and Hector Geffner. (2003). Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming. ICAPS.

[5] D. McDermott. (1088). PDDL – the planning domain definition language. AIPS'98 IPC Committee.

[6] E.P. Pednault. (1989). ADL: Exploring the middle-. ground between STRIPS and the Situation Calculus. In Proc. Int. Knowledge Representation Conf.

[7] N. Meuleau and D. Smith. (2003). Optimal limited contingency planning. In ICAPS - Workshop on Planning under Uncertainty and Incomplete Information.

[8] P. Bertoli, A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. (2001). MBP: a Model Based Planner. In Proc. of the IJCAI'01 Workshop on Planning under Uncertainty and Incomplete Information.

[9] R. Petrick and Fahiem Bacchus. (2002). A knowledge-Based approach to Planning with Incomplete Information and Sensing" In Proc. AIPS 2002.

[10]Richard Fikes, Nils J. Nilsson. (1971). STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. IJCAI.

[11]U. Dal Lago, M. Pistore, P. Traverso. (2002). Planning with a Language for Extended Goals. In Proc. AAAI 2002.

[12]Zhengzhu Feng and Eric A. Hansen. (2004). Symbolic Stochastic Focused Dynamic Programming with Decision Diagrams. IPC-4.

[13]A. Mediratta, and B. Srivastava. (2006). Applying Planning in Composition of Web Services with a User-Driven Contingent Planner. In IBM Research Report RI 06002.

[14]Younes, H. L. S., and Littman, M. L. (2004). PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-167.

[15]Blai Bonet and Robert Givan. (2005). 5th International Planning Competition: Non-deterministic Track Call for Participation. IPC-5.

[16]Michael L. Littman and Håkan L. S. Younes. (2004). Introduction to the Probabilistic Planning Track. IPC-4.