LIFT-UP: Lifted First-Order Planning Under Uncertainty

Steffen Hölldobler and Olga Skvortsova
International Center for Computational Logic
Technische Universität Dresden, Dresden, Germany
{sh,skvortsova}@iccl.tu-dresden.de

Abstract

We present a new approach for solving first-order Markov decision processes combining first-order state abstraction and heuristic search. In contrast to existing systems, which start with propositionalizing the decision process and then perform state abstraction on its propositionalized version we apply state abstraction directly on the decision process avoiding propositionalization. Secondly, guided by an admissible heuristic, the search is restricted to those states that are reachable from the initial state. We demonstrate the usefulness of the above techniques for solving first-order Markov decision processes within a domain dependent system called FluCaP which participated in the probabilistic track of the 2004 International Planning Competition. Working toward a domain independent implementation we present novel approaches to θ-subsumption involving literal and object contexts.

1 Introduction

We are interested in solving probabilistic planning problems, i. e. planning problems, where the execution of an action leads to the desired effects only with a certain probability. For such problems, Markov decision processes have been adopted as a representational and computational model in much recent work, e.g., by [BBS95]. They are usually solved using the so-called dynamic programming principle [BDH99] employing a value iteration algorithm. Classical dynamic programming algorithms explicitly enumerate the state space and are thus exponenetial. In recent years several methods have been developed which avoid an explicit enuration of the state space. The most prominent are state abstraction [BDH99], heuristic search (e. g. [BBS95, DKKN95]) and a combination of both as used, for example, in symbolic LAO* [FH02].

A common feature of these approaches is that a Markov decision process is propositionalized before state abstraction techniques and heuristic search algorithms are applied within a value iteration algorithm. Unfortunately, the propositionalization step itself may increase the problem significantly. To overcome this problem, it was first proposed in [BRP01] to solve first-order Markov decision processes by applying a first-order value iteration algorithm and first-order state abstraction techniques. Whereas this symbolic dynamic programming approach was rooted in a version of the Situation Calculus [Rei91], we have reformulated and extended these ideas in a variant of the fluent calculus [HS04]. In this system, which is now called LIFT-UP, lifted first-order planning under uncertainty can be performed.

In the LIFT-UP system, states and actions are expressed in the language of the fluent calculus [HS90], which is slightly extended to handle probabilities. In addition, value functions and policies are represented by constructing first-order formulas which partition the state space into clusters, referred to as *abstract states*. Then, value iteration can be performed on top of these clusters, obviating the need for explicit state enumeration. This allows the solution of first-order Markov decision processes without requiring explicit state enumeration or propositionalization. In addition, heuristics are used to guide the search and normalization techniques are applied to eliminate redundant states. The LIFT-UP approach can thus be viewed as a first-order generalization of symbolic LAO* or, alternatively, as symbolic dynamic programming enhanced by heuristic search and state space normalization.

To evaluate the LIFT-UP system we have developed a domain-dependent implementation called FluCaP. It can solve probabilistic blocksword problems as they appeared, for example, in the colored blocksworld domain of the 2004 International Planning Competition. FluCaP is quite successful and outperforming other systems on truly first-order problems. On the other hand and working towards a domain-independent implementation we have studied θ-subsumption algorithms. θ-subsumption problems arise at various places in the LIFT-UP system: The normalization process requires to check whether one abstract state subsumes another one; the check whether an action is applicable to some abstract state and the computation of set of the successor or predecessor states also requires subsumption. One should observe that the latter application requires to compute a complete set of substitutions.

In this paper we give an overview of the LIFT-UP approach.

2 First-order Markov Decision Processes

A *Markov decision process*, is a tuple $(\mathcal{Z}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{C})$, where $\mathcal{Z}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, and $\mathcal{P} : \mathcal{Z} \times \mathcal{Z} \times \mathcal{A} \to [0, 1]$, written $\mathcal{P}(z'|z, a)$, specifies transition probabilities. In particular, $\mathcal{P}(z'|z, a)$ denotes the probability of ending up at state z' given that the agent was in state z and action a was executed. $\mathcal{R} : \mathcal{Z} \to \mathbb{R}$ is a real-valued reward function associating with each state z its immediate utility $\mathcal{R}(z)$. $\mathcal{C} : \mathcal{A} \to \mathbb{R}$ is a real-valued cost function associating a cost $\mathcal{C}(a)$ with each action a. A *sequential decision problem* consists of a Markov decision process and is the problem of finding a policy $\pi : \mathcal{Z} \to \mathcal{A}$ that maximizes the total expected discounted reward received when executing the policy π over an infinite (or indefinite) horizon. A Markov decision process is said to be *first-order* if the expressions used to define $\mathcal{Z}$, $\mathcal{A}$ and $\mathcal{P}$ are first-order.

The *value* $V_\pi(z)$ of a state z with respect to the policy π is defined as

$$V_\pi(z) = \mathcal{R}(z) + \mathcal{C}(\pi(z)) + \gamma \sum_{z' \in \mathcal{Z}} \mathcal{P}(z'|z, \pi(z)) V_\pi(z'),$$

where $0 \leq \gamma \leq 1$ is a discount factor. We take γ equal to 1 for indefinite-horizon problems only, i. e. when a goal is reached the system enters an absorbing state in which no further rewards or costs are accrued. A value function V is set to be *optimal* if it

satisfies

$$\mathcal{R}(z) + \max_{a \in \mathcal{A}} \{\mathcal{C}(a) + \gamma \sum_{z' \in \mathcal{Z}} \mathcal{P}(z'|z,a) V^*(z')\} \ ,$$

for each $z \in \mathcal{Z}$; in this case the value function is usually denoted by $V^*(z)$. The optimal policy is extracted from the optimal value function.

We assume that planning problems meet the following requirements:

1. Each problem has a goal statement, identifying a set of absorbing goal states.
2. A positive reward is associated with each action ending in a goal state; otherwise it is 0.
3. A cost is associated with each action.
4. A "done" action is available in all states.

The "done" action can be used to end any further accumulation of reward. Together, these conditions ensure that an MDP model of a planning problem is a positive bounded model as described by [Put94]. Such planning problems are also often called stochastic shortest path problems.

3 Probabilistic Fluent Calculus

States, actions, transition probabilities, cost and reward function are specified in a probabilistic and sorted extension of the fluent calculus [HS90, Thi98].

Fluents and States Let Σ denote a set of function symbols containing the binary function symbol $\circ$ and the nullary function symbol 1. $\circ$ is an AC1-symbol with 1 as unit element. Let $\Sigma^- = \Sigma \setminus \{\circ, 1\}$. Non-variable Σ^--terms are called *fluents*. Let $f(t_1, \ldots, t_n)$ be a fluent. The terms t_i, $1 \leq i \leq n$ are called *objects*. A *state* is a finite set of ground fluents. Let $\mathcal{D}$ be the set of all states.

Fluent Terms and Abstract States *Fluent terms* are defined inductively as follows: 1 is a fluent term; each fluent is a fluent term; if G_1 and G_2 are fluent terms, then so is $G_1 \circ G_2$. Let $\mathcal{F}$ be the set of all fluent terms. We assume that each fluent term obeys the *singularity condition*: each fluent may occur at most once in a fluent term. Because of the latter, there is a bijection $\cdot^M$ between ground fluent terms and states. Some care must be taken when instantiating a non-ground fluent term F by a substitution θ because $F\theta$ may violate the singularity condition. A substitution θ is *allowed* for fluent term F if $F\theta$ meets the singularity condition.

Abstract states are expressions of the form F or $F \circ X$, where F is a fluent term and X is a variable of sort fluent term. Let $\mathcal{S}$ denote the set of abstract states. Abstract states denote sets of states as defined by the mapping $\cdot^I : \mathcal{S} \to 2^{\mathcal{D}}$: Let Z be an abstract state. Then

$$[Z]^I = \{[Z\theta]^M \mid \theta \text{ is an allowed grounding substitution for } Z\}.$$