Probabilistic Planning via Linear Value-approximation of First-order MDPs

Scott Sanner
University of Toronto
Department of Computer Science
Toronto, ON, M5S 3H5, CANADA
ssanner@cs.toronto.edu

Craig Boutilier
University of Toronto
Department of Computer Science
Toronto, ON, M5S 3H5, CANADA
cebly@cs.toronto.edu

Abstract

We describe a probabilistic planning approach that translates a PPDDL planning problem description to a first-order MDP (FOMDP) and uses approximate solution techniques for FOMDPs to derive a value function and corresponding policy. Our FOMDP solution techniques represent the value function linearly w.r.t. a set of first-order basis functions and compute suitable weights using *lifted*, first-order extensions of approximate linear programming (FOALP) and approximate policy iteration (FOAPI) for MDPs. We additionally describe techniques for automatic basis function generation and decomposition of universal rewards that are crucial to achieve autonomous and tractable FOMDP solutions for many planning domains.

From PPDDL to First-order MDPs

It is straightforward to translate a PPDDL [12] planning domain into the situation calculus representation used for first-order MDPs (FOMDPs); the primary part of this translation requires the conversion of PPDDL action schemata to *effect axioms* in the situation calculus, which are then compiled into *successor-state axioms* [8] used in the FOMDP description. In the following algorithm description, we will assume that we are given a FOMDP specification and we will describe techniques for approximating its value function linearly w.r.t. a set of first-order basis functions. From this value function it is straightforward to derive a first-order policy representation that can be used for action selection in the original PPDDL planning domain.

Linear Value Approximation for FOMDPs

The following explanation assumes the reader is familiar with the FOMDP formalism and operators used in Boutilier, Reiter and Price [2] and extended by Sanner and Boutilier [9]. In the following text, we will refer to function symbols $A_i(\vec{x})$ that correspond to parameterized actions in the FOMDP; for every action and fluent, we expect that a successor state axiom has been defined. The reader should be familiar with the notation and use of the $rCase$, $vCase$, and $pCase$ *case statements* for representing the respective FOMDP reward, value, and transition functions. The reader should also be familiar with the *case operators* $\oplus$, $\ominus$, $\cup$, and $Regr(\cdot)$ [2] as well as $FODTR(\cdot)$, $B^{A(\vec{x})}(\cdot)$, and $B^{A}(\cdot)$ [9].

Value Function Representation

Following [9], we represent a value function as a weighted sum of k *first-order basis functions* in case statement format, denoted $bCase_j(s)$, each containing a *small* number of formulae that provide a first-order abstraction of state space:

$$vCase(s) = \oplus_{i=1}^{k} \; w_i \cdot bCase_i(s) \tag{1}$$

Using this format, we can often achieve a reasonable approximation of the exact value function by exploiting the additive structure inherent in many real-world problems (e.g., additive reward functions or problems with independent subgoals). Unlike exact solution methods where value functions can grow exponentially in size during the solution process and must be logically simplified [2], here we maintain the value function in a compact form that requires no simplification, just discovery of good weights.

We can easily apply the FOMDP backup operator $B^{A(\vec{x})}$ [9] to this representation and obtain some simplification as a result of the structure in Eq. 1. Exploiting the properties of the $Regr$ and $\oplus$ operators, we find that the backup $B^{A(\vec{x})}$ of a linear combination of basis functions is simply the linear combination of the first-order decision-theoretic regression (FODTR) of each basis function [9]:

$$B^{A(\vec{x})}(\oplus_i \; w_i bCase_i(s)) = \\ rCase(s,a) \oplus (\oplus_i \; w_i FODTR(bCase_i(s), A(\vec{x}))) \tag{2}$$

A corresponding definition of B^A follows directly [9]. It is important to note that during the application of these operators, we never explicitly ground states or actions, in effect achieving *both state and action space abstraction*.

First-order Approximate Linear Programming

First-order approximate linear programming (FOALP) was introduced by Sanner and Boutilier [9]. Here we present a linear program (LP) with first-order constraints that generalizes the solution from MDPs to FOMDPs:

$$\begin{aligned} &\text{Variables: } w_i \; ; \;\; \forall i \le k \\ &\text{Minimize: } \sum_{i=1}^{k} w_i \sum_{\langle \phi_j, t_j \rangle \in bCase_i} \frac{t_j}{|bCase_i|} \\ &\text{Subject to: } 0 \ge B^{A}_{\max}(\oplus_{i=1}^{k} \; w_i \cdot bCase_i(s)) \\ &\qquad\qquad \ominus (\oplus_{i=1}^{k} \; w_i \cdot bCase_i(s)) \; ; \; \forall \; A, s \end{aligned} \tag{3}$$

The objective of this LP requires some explanation. If we were to directly generalize the objective for MDPs to that of FOMDPs, the objective would be ill-defined (it would sum over infinitely many situations s). To remedy this, we suppose that each basis function partition is chosen because it represents a potentially useful partitioning of state space, and thus sum over each case *partition*.

This LP also contains a first-order specification of constraints, which somewhat complicates the solution. Before tackling this, we introduce a general *first-order LP* format that we can reuse for approximate policy iteration:

$$\begin{aligned} &\text{Variables: } v_1, \ldots, v_k \ ; \\ &\text{Minimize: } f(v_1, \ldots, v_k) \\ &\text{Subject to: } 0 \geq case_{1,1}(s) \oplus \ldots \oplus case_{1,n}(s) \ ; \ \forall \ s \\ &\qquad\qquad \vdots \\ &\qquad\qquad 0 \geq case_{m,1}(s) \oplus \ldots \oplus case_{m,n}(s) \ ; \ \forall \ s \end{aligned} \tag{4}$$

The variables and objective are as defined in a typical LP, the main difference being the form of the constraints. While there are an infinite number of constraints (i.e., one for every situation s), we can work around this since case statements are finite. Since the value t_i for each case partition $\langle \phi_i(s), t_i \rangle$ is piecewise constant over all situations satisfying $\phi_i(s)$, we can explicitly sum over the $case_i(s)$ statements in each constraint to yield a single case statement. For this "flattened" case statement, we can easily verify that the constraint holds in the finite number of piecewise constant partitions of the state space. However, generating the constraints for each "cross-sum" can yield an exponential number of constraints. Fortunately, we can generalize constraint generation techniques [10] to avoid generating all constraints. We refer to [9] for further details. Taken together, these techniques yield a practical FOALP solution to FOMDPs.

First-order Approximate Policy Iteration

We now turn to a first-order generalization of approximate policy iteration (FOAPI). Policy iteration requires that a suitable first-order policy representation be derivable from the value function $vCase(s)$. Assuming we have m parameterized actions $\{A_1(\vec{x}), \ldots, A_m(\vec{x})\}$, we can represent the policy $\pi Case(s)$ as:

$$\pi Case(s) = \max(\bigcup_{i=1 \ldots m} B^{A_i}(vCase(s))) \tag{5}$$

Here, $B^{A_i}(vCase(s))$ represents the values that can be achieved by any instantiation of the action $A_i(\vec{x})$ and the $\max$ case operator ensures that the highest possible value is assigned to every situation s. For bookkeeping purposes, we require that each partition $\langle \phi, t \rangle$ in $B^{A_i}(vCase(s))$ maintain a mapping to the action A_i that generated it, which we denote as $\langle \phi, t \rangle \rightarrow A_i$. Then, given a particular world state s at run-time, we can evaluate $\pi Case(s)$ to determine which policy partition $\langle \phi, t \rangle \rightarrow A_i$ is satisfied in s and thus, which action A_i should be applied. If we retrieve the bindings of the existentially quantified action variables in ϕ (recall that B^{A_i} existentially quantifies these), we can easily determine the instantiation of action A_i prescribed by the policy.

For our algorithms, it is useful to define a set of case statements for each action A_i that is satisfied only in the world states where A_i should be applied according to $\pi Case(s)$. Consequently, we define an action restricted policy $\pi Case_{A_i}(s)$ as follows:

$$\pi Case_{A_i}(s) = \{\langle \phi, t \rangle | \langle \phi, t \rangle \in \pi Case(s) \text{ and } \langle \phi, t \rangle \rightarrow A_i\}$$

Following the approach to approximate policy iteration for factored MDPs provided by Guestrin *et al* [4], we can generalize approximate policy iteration to the first-order case by calculating successive iterations of weights $w_j^{(i)}$ that represent the best approximation of the fixed point value function for policy $\pi Case^{(i)}(s)$ at iteration i. We do this by performing the following two steps at every iteration i: (1) Obtaining the policy $\pi Case(s)$ from the current value function and weights ($\sum_{j=1}^{k} w_j^{(i)} bCase_j(s)$) using Eq. 5, and (2) solving the following LP in the format of Eq. 4 that determines the weights of the Bellman error minimizing approximate value function for policy $\pi Case(s)$:

$$\begin{aligned} &\text{Variables: } w_1^{(i+1)}, \ldots, w_k^{(i+1)} \\ &\text{Minimize: } \phi^{(i+1)} \\ &\text{Subject to: } \phi^{(i+1)} \geq \Big| \pi Case_A(s) \oplus \oplus_{j=1}^{k} [w_j^{(i+1)} bCase_j(s)] \\ &\qquad\qquad \ominus \oplus_{j=1}^{k} w_j^{(i+1)} (B^A_{\max} bCase_j)(s) \Big| ; \ \forall A, s \end{aligned} \tag{6}$$

We've reached convergence *if* $\pi^{(i+1)} = \pi^{(i)}$. If policy iteration converges, the loss bounds from [4] generalize directly to the first-order case.

Greedy Basis Function Generation

The use of linear approximations requires a good set of basis functions that span a space that includes a good approximation to the value function. While some work has addressed the issue of basis function generation [7; 5], none has been applied to RMDPs or FOMDPs. We consider a basis function generation method that draws on the work of Gretton and Thiebaux [3], who use inductive logic programming (ILP) techniques to construct a value function from sampled experience. Specifically, they use regressions of the reward as candidate building blocks for ILP-based construction of the value function. This technique has allowed them to generate fully or k-stage-to-go optimal policies for a range of Blocks World problems.

We leverage a similar approach for generating candidate basis functions for use in the FOALP or FOAPI solution techniques. If some portion of state space ϕ has value $v > \tau$ in an existing approximate value function for some nontrivial threshold τ, then this suggests that states that can reach this region (i.e., found by Regr(ϕ) through some action) should also have reasonable value. However, since we have already assigned value to ϕ, we want the new basis function to focus on the area of state space not covered by ϕ. Consequently, we negate ϕ and conjoin it with $Regr(\phi)$ yielding the new basis function $[\neg\phi \wedge Regr(\phi) : 1; \ \phi \vee \neg Regr(\phi) : 0]$. The "orthogonality" of newly generated basis functions also allows for computational optimizations since many combinations of basis function partitions are mutually exclusive and thus need not be examined.

Handling Universal Rewards

In first-order domains, we are often faced with *universal reward expressions* that assign some positive value to the world states satisfying a formula of the general form $\forall y \ \ \phi(y, s)$, and 0 otherwise. For instance, in a logistics problem, we can use a predicate $Dst(t, c)$ to indicate that truck t is at city c and a fluent $TAt(t, c, s)$ to indicate that truck t is at city c in situation s. Then a reward may be given for having all trucks at their assigned destination: $\forall t, cDst(t, c) \rightarrow TAt(t, c, s)$. One difficulty with such rewards is that our basis function approach provides a piecewise constant approximation to the value function (i.e., each basis function aggregates state space into regions of equal value, with the linear combination simply providing constant values over somewhat smaller regions). However, the value function for problems with universal rewards typically depends (often in a linear or exponential way) on the *number* of domain objects of interest. For instance, in our example, value at a state depends on the number of trucks not at their proper destination (since that impacts the time it will take to obtain the reward). Unfortunately, *this cannot be represented* concisely using the piecewise constant decomposition offered by first-order basis functions. As noted by Gretton and Thiebaux [3], effectively handling universally quantified rewards is one of the most pressing issues in the practical solution of FOMDPs.

To address this problem we adopt a decompositional approach, motivated in part by techniques for additive rewards in MDPs [1; 11; 6; 7]. Intuitively, given a goal-oriented reward that assigns positive reward if $\forall y G(y, s)$ is satisfied, and zero otherwise, we can decompose it into a set of ground goals $\{G(\vec{y_1}), \ldots, G(\vec{y_n})\}$ for all possible $\vec{y_j}$ in a ground domain of interest. If we reach a world state where all ground goals are true, then we have satisfied $\forall y G(y, s)$.

Of course, our methods solve FOMDPs without knowledge of the specific domain, so the set of ground goals that will be faced at run-time is unknown. So in the offline solution of the MDP we assume a a *generic* ground goal $G(\vec{y}^*)$ for a "generic" object vector $\vec{y}^*$. It is easy to construct an instance of the reward function $rCase(s)$ for this single goal, and solve for this simplified generic goal using FOALP or FOAPI. This produces a value function and policy that assumes that $\vec{y}^*$ is the only object vector of interest (i.e., satisfying relevant type and preconditions) in the domain. From this, we can also derive the optimal Q-function for the simplified "generic" domain (and action template $A_i(\vec{x})$): $Q_{G(\vec{y}^*)}(A_i(\vec{x}), s) = B^{A_i}(vCase(s))$.[1] Intuitively, given a ground state s, the optimal action for this generic goal can be determined by finding the ground $A_i(\vec{x}^*)$ for this s with max Q-value.

With the solution (i.e., optimal Q-function) of a generic goal FOMDP, we now address the online problem of action selection for a *specific* domain instantiation. Assume a set of ground goals $\{G(\vec{y_1}), \ldots, G(\vec{y_n})\}$ corresponding to a specific domain given at run-time. If we assume that (typed) domain objects are treated uniformly in the uninstantiated FOMDP, as is the case in many logistics and planning problems, then we obtain the Q-function for any goal $G(\vec{y_j})$ by replacing all ground terms $\vec{y}^*$ with the respective terms $\vec{y_j}$ in $Q_{G(\vec{y}^*)}(A_i(\vec{x}), s)$ to obtain $Q_{G(\vec{y_j})}(A_i(\vec{x}), s)$.

Action selection requires finding an action that maximizes value w.r.t. the original universal reward. Following [1; 6], we do this by treating the *sum of the Q-values* of any action in the subgoal MDPs as a measure of its Q-value in the joint (original) MDP. Specifically, we assume that each goal contributes uniformly and additively to the reward, so the Q-function for a entire set of ground goals $\{G(\vec{y_1}), \ldots, G(\vec{y_n})\}$ determined by our domain instantiation is just $\sum_{j=1}^{n} \frac{1}{n} Q_{G(\vec{y_j})}(A_i(\vec{x}), s)$. The action selection (at run-time) in any ground state is realized by choosing that action with maximum joint Q-value. Naturally, we do not want to explicitly create the joint Q-function, but an efficient scoring technique that evaluates potentially useful actions by iterating through the individual Q-functions is very straightforward.

While this additive and uniform decomposition may not be appropriate for all domains with goal-oriented universal rewards, we have found it to be highly effective for the *Box-World* logistics domain from the ICAPS 2004 probabilistic planning competition. And while this approach can only currently handle rewards with universal quantifiers, this reflects the form of many practical planning problems.

[1] Since the B^A operator can often retain much of the additive structure in the linear approximation of $vCase(s)$ [9], representation and computation with this Q-function is *very* efficient.

References

[1] C. Boutilier, R. I. Brafman, and C. Geib. Prioritized goal decomposition of Markov decision processes: Toward a synthesis of classical and decision theoretic planning. *IJCAI-97*, pp.1156–1162, Nagoya, 1997.

[2] C. Boutilier, R. Reiter, and B. Price. Symbolic dynamic programming for first-order MDPs. *IJCAI-01*, 2001.

[3] C. Gretton and S. Thiebaux. Exploiting first-order regression in inductive policy selection. *UAI-04*, 2004.

[4] C. Guestrin, D. Koller, R. Parr, and S. Venktaraman. Efficient solution methods for factored MDPs. *JAIR*, 2002.

[5] S. Mahadevan. Samuel meets amarel: Automating value function approximation using global state space analysis. *AAAI-05*, pp.1000–1005, Pittsburgh, 2005.

[6] N. Meuleau, M. Hauskrecht, K. Kim, L. Peshkin, L. P. Kaelbling, T. Dean, and C. Boutilier. Solving very large weakly coupled Markov decision processes. *AAAI-98*, 1998.

[7] P. Poupart, C. Boutilier, R. Patrascu, and D. Schuurmans. Piecewise linear value function approximation for factored MDPs. *AAAI 02*, pp.292–299, Edmonton, 2002.

[8] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[9] S. Sanner and C. Boutilier. Approximate linear programming for first-order MDPs. *UAI 2005)*, Edinburgh, 2005.

[10] D. Schuurmans and R. Patrascu. Direct value approximation for factored MDPs. *NIPS-2001*, Vancouver, 2001.

[11] S. P. Singh and D. Cohn. How to dynamically merge Markov decision processes. *NIPS-98*, 1998.

[12] H. Younes and M. Littman. PPDDL: The probabilistic planning domain definition language, 2004.