Practical Linear-value Approximation Techniques for First-order MDPs

Scott Sanner & Craig Boutilier

University of Toronto

UAI 2006
Why Solve First-order MDPs?

- Relational desc. of (prob) planning domain in (P)PDDL:

```
(:action load-box-on-truck-in-city
  :parameters (?b - box ?t - truck ?c - city)
  :precondition (and (BIn ?b ?c) (TIn ?t ?c))
  :effect (and (On ?b ?t) (not (BIn ?b ?c)))
```

Box World:

- Can solve a ground MDP for each domain instantiation:
  - 3 trucks: ![Trucks]
  - 2 planes: ![Planes]
  - 4 boxes: ![Boxes]

- Or solve first-order MDP for all domain inst. at once!
  - Lift PPDDL MDP specification to first-order (FOMDP)
  - Soln makes value distinctions for all dom. instantiations!
Background / Talk Outline

1) Symbolic DP for first-order MDPs (BRP, 2001)
   - Defines FOMDP / operators / value iteration
   - Requires FO simplification for compactness 😞

2) First-order approx. linear prog. (SB, 2005)
   - Approximate value with linear comb. of basis funs.
   - No simplification → project onto weight space 😊

3) Many practical questions remaining (SB, 2006)
   - Other algorithms – first-order API?
   - Where do basis functions come from?
   - How to efficiently handle universal rewards?
   - Optimizations for scalability?
FOMDP Foundation: SitCalc

- **Deterministic Actions:** $loadS(b,t)$, $unloadS(b,t)$, ...
- **Situations:** $S_0$, $do(loadS(b,t), S_0)$, ...
- **Fluents:** $BIn(b,c,s)$, $TIn(t,c,s)$, $On(b,t,s)$

**Successor-state axioms (SSAs) for each fluent $\mathcal{F}$:**
- Describe how action affects fluent (like det. FO-DBN)
- **Ex:** $BIn(b,c,do(a,s)) \equiv$
  
  1. $Bin(b,c,s)$ AND $a \neq loadS(b,t)$
  OR 2. for some $t$: $a = unloadS(b,t)$ AND $TIn(t,c,s)$

- **Regression Operator:** $Regr(\varphi) = \varphi'$
  - Takes a formula $\varphi$ describing a *post-action* state
  - Uses SSAs to build $\varphi'$ describing *pre-action* state
  - Crucial for backing up value fun to produce Q-fun!
FOMDP Case Representation

- **Case:** Assign value to first-order state abstraction
  - E.g., can express reward in *BoxWorld* FOMDP as…

\[
\text{rCase}(s) = \begin{array}{c|c}
\forall b,c. \text{Dest}(b,c) \Rightarrow B\text{In}(b,c,s) & 1 \\
\neg \forall b,c. \text{Dest}(b,c) \Rightarrow B\text{In}(b,c,s) & 0
\end{array}
\]

- **Operators:** Define unary, binary case operations
  - E.g., can take “cross-sum” $\oplus$ (or $\otimes$, $\ominus$) of two cases…

\[
\begin{array}{c|c}
\exists x. A(x) & 10 \\
\neg \exists x. A(x) & 20
\end{array} \oplus \begin{array}{c|c}
\exists y. A(y) \land B(y) & 3 \\
\neg \exists y. A(y) \land B(y) & 4
\end{array} = \begin{array}{c|c}
\exists x. A(x) \land \exists y. A(y) \land B(y) & 13 \\
\exists x. A(x) \land \neg \exists y. A(y) \land B(y) & 14 \\
\neg \exists x. A(x) \land \exists y. A(y) \land B(y) & 23 \\
\neg \exists x. A(x) \land \neg \exists y. A(y) \land B(y) & 24
\end{array}
\]

- Must remove inconsistent elements (i.e., red bar ———— )
FOMDP Actions and FODTR

- SitCalc is deterministic, how to handle probabilities?
  - User’s stochastic actions: load(b,t)
  - Nature’s deterministic choice: loadS(b,t), loadF(b,t)
  - Probability distribution over Nature’s choice:

\[
P(\text{loadS}(b,t) \mid \text{load}(b,t)) = \begin{array}{c|c}
\text{snow}(s) & .1 \\
\neg \text{snow}(s) & .5 \\
\end{array}
\]

\[
P(\text{loadF}(b,t) \mid \text{load}(b,t)) = 1 \oplus P(\text{loadS}(b,t) \mid \text{load}(b,t))
\]

- First-order decision-theoretic regression (FODTR):
  - Given value fun \( v\text{Case}(s) \) and user action, produces first-order description of “Q-fun” (modulo reward)

\[
\text{“Q-Fun”} = \text{FODTR}[v\text{Case}(s), \text{load}(b,t)] = \text{Regr}[v\text{Case}(\text{after loadS...})] \otimes P(\text{loadS...} \mid \text{load...})
\oplus \text{Regr}[v\text{Case}(\text{after loadF...})] \otimes P(\text{loadF...} \mid \text{load...})
\]
FOMDP Backup Operators

In fact, there are 3 types of “Q-funs”/backup operators:

1) \( B^{A[x]}[\text{vCase}(s)] = r\text{Case}(s) \oplus \gamma \cdot \text{FODTR}[\text{vCase}(s)] \)

Let \( B^{\text{load}(b,t)}[\text{vCase}(s)] = \)

\[
\begin{array}{c|c}
\phi(b,t) & .9 \\
\neg \phi(b,t) & 0 \\
\end{array}
\]

Think of as \( Q(A(x),s) \), note the free vars!

2) \( B^A[\text{vCase}(s)] = \exists x. B^{A(x)}[\text{vCase}(s)] \) (action abstraction!)

\( B^{\text{load}}[\text{vCase}(s)] = \)

\[
\begin{array}{c|c}
\exists b,t. \phi(b,t) & .9 \\
\exists b,t. \neg \phi(b,t) & 0 \\
\end{array}
\]

Think of as \( \sim Q(A,s) \), no free vars but now overlap!

3) \( B^{A_{\text{max}}}[\text{vCase}(s)] = \max( B^A[\text{vCase}(s)] ) \)

\( B^{\text{load}_{\text{max}}}[\text{vCase}(s)] = \)

\[
\begin{array}{c|c}
\exists b,t. \phi(b,t) & .9 \\
\neg(\exists b,t. \phi(b,t)) \land \exists b,t. \neg \phi(b,t) & 0 \\
\end{array}
\]

Think of as \( Q(A,s) \), \textit{no} free vars and \textit{no} overlap!
First-order Approx. Linear Prog. (FOALP)

- Represent value fn as linear comb. of k basis fns:

\[ v_{\text{Case}}(s) = w_1 \oplus \cdots \oplus w_k \]

\begin{align*}
\exists b, c & \text{ BIn}(b, c, s) & 1 \\
\neg \exists b, c & \text{ BIn}(b, c, s) & 0 \\
\exists t, c & \text{ TIn}(t, c, s) & 1 \\
\neg \exists t, c & \text{ TIn}(t, c, s) & 0
\end{align*}

- Reduces MDP solution to finding good weights…

generalize approx. LP used by (van Roy, GKP, SP):

<table>
<thead>
<tr>
<th>Vars: ( w_i; \ i \leq k )</th>
</tr>
</thead>
</table>

Minimize:

\[ \sum_s \sum_{i=1..k} w_i \cdot \text{bCase}_i(s) \]

Subject to:

\[ 0 \geq B_{\text{max}} \left[ \bigoplus_{i=1..k} w_i \cdot \text{bCase}_i(s) \right] \]

\[ \ominus \bigoplus_{i=1..k} w_i \cdot \text{bCase}_i(s); \quad \forall a \in A, s \]

- FOALP issues resolved in (SB, 2005):

  - \( \infty \) sum in objective: We give principled approximation
  - \( \infty \) constraints: Only finite set of distinct constraints, solve exactly & efficiently w/ constraint gen. (SP)
First-order Approx. Policy Iter. (FOAPI)

- Need an explicit representation of a policy:
  - $\pi_{\text{Case}}(s) = \max( \cup_{i=1..m} B_i^a [v_{\text{Case}}(s)] )$
  - Each case partition should retain mapping to $A_i$

- Now separate partitions in $A_i$-specific policies:
  - $\pi_{\text{Case}_{A_i}}(s) = \{ \text{part} \in \pi_{\text{Case}}(s) \text{ s.t. part} \rightarrow A_i \}$
  - Specifies states where policy would apply $A_i$

- FOAPI: Direct generalization of GKP (exact objective!)
  - Start w/ $w_i^0 = 0, \pi_{\text{Case}}^0(s)$; iterate LP soln until $\pi^{j+1} = \pi^j$:

Vars: $w_i^{(j+1)}; i \leq k$

Minimize: $\phi^{(j+1)}$

Subject to: $\phi^{(j+1)} \geq | \pi_{\text{Case}_{a}}(s) \oplus B_{a_{\text{max}}}(\oplus_{i=1..k} w_i^{(j+1)} \cdot b_{\text{Case}_{i}}(s))$
  $\oplus \oplus_{i=1..k} w_i^{(j+1)} \cdot b_{\text{Case}_{i}}(s) |; \forall a \in A, s$

- Use cgen; if converges, obtain bounds on policy (GKP)!
Generating Basis Functions

- **Where do basis functions come from?**
  - Major question for automation!
  - Huge candidate space if systematically building basis functions for all first-order formulae

- **Idea (GT, 2004):** Regressions from goal make good candidate basis functions!
  - Given initial basis function for reward: \( \exists b. \text{Bin}(b,P,s) \)
  - Regr w/ unload: \( \exists b. \text{Bin}(b,P,s) \lor (\exists b^*, t^*. \text{In}(t^*, P, s) \land \text{On}(b^*, t^*, s)) \)

- Render basis *disjoint* from parents, will use later

- **Iteratively solve FOMDP**
  - Retain all basis functions with wgt. > threshold \( \tau \)
  - Generate new basis fns from retained set
Problems w/ Universal Reward

- Universal rewards are difficult for FOMDPs, e.g.
  - Given reward:
    
    \[
    r\text{Case}(s) = \begin{cases} 
      \forall b,c. \text{Dest}(b,c) \Rightarrow \text{BIn}(b,c,s) & 1 \\
      \neg" & 0 
    \end{cases}
    \]

  - Exact n-stage-to-go value function has form:
    
    \[
    v\text{Case}^n(s) = \begin{cases} 
      \forall b,c. \text{Dest}(b,c) \Rightarrow \text{BIn}(b,c,s) & 1 \\
      1\text{ box not at dest} & \gamma \\
      \ldots & \ldots \\
      n-1\text{ boxes not at dest} & \gamma^{n-1} 
    \end{cases}
    \]

  - Exact value function has infinitely many values!
  - Cannot compactly represent such structure with piecewise-constant case approximation of value fn
Additive Goal Decomposition

- Solution for universal rewards:

  \[ \forall b,c. \text{Dest}(b,c) \Rightarrow \text{BIn}(b,c,s) \]

  Solve FOMDP for: \( \text{BIn}(b^*,c^*,s) \)

  Given solution, gen. Q-funs \( Q(A,s)_{b^*,c^*}(s) \) for \( \forall a \in A \)

- At run-time: Given concrete domain, e.g.

  - Instantiation: \( \{ \text{Dest}(b_1,c_1), \text{Dest}(b_2,c_2), \text{Dest}(b_3,c_3) \} \)

  - Let overall \( Q(A,s) = Q(A,s)_{b_1,c_1}(s) + Q(A,s)_{b_2,c_2}(s) + Q(A,s)_{b_3,c_3}(s) \) for \( \forall a \in A \)

  - To execute policy: select action that maximizes sum of values across all Q-funs, i.e., \( Q(A,s) \)

  - Only heuristic: works in many, but not all cases
Optimizations

- Exploiting disjointness in basis functions:
  - Worst case for set $\mathcal{B}$ of basis functions: must examine $2^{|\mathcal{B}|}$ case partitions in constraint generation.
  - But for any pairwise disjoint set $\mathcal{B}'$ of basis functions, need examine only $|\mathcal{B}'|$ case partitions in cgen.
  - Basis generation enforces disjointness b/w child/parent!

- Exploiting implicit max in constraint generation:
  - In constraints, substitute $0 \geq B^a_{\max}$ ... with $0 \geq B^a$ ...

- Removing internal redundancy/inconsistency w/ BDDs:
  - Given: $(\exists x \ A(x)) \land (\exists x \ A(x)) \land (\exists x \ A(x) \land B(x))$

<table>
<thead>
<tr>
<th>Prop Var</th>
<th>FOL Mapping</th>
<th>Impl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$\exists x \ A(x) \land B(x)$</td>
<td>$a \Rightarrow b$</td>
</tr>
<tr>
<td>$b$</td>
<td>$\exists x \ A(x)$</td>
<td>$\neg b \Rightarrow \neg a$</td>
</tr>
</tbody>
</table>

$\exists x \ A(x) \land B(x)$
Empirical Results: Runtime

- Offline solution times for BoxWorld & BlocksWorld:
  - Without optimizations, cannot get past iteration 2 (> 36000 sec.)
  - BoxWorld: Policies simple, fewer constraints for FOAPI
  - BlocksWorld: Policies complex (lots of equality)
Empirical Results: Performance

- Evaluated cumulative reward on ICAPS 2004 Prob. Planning Comp. BoxWorld (bx) and BlocksWorld (bw):

<table>
<thead>
<tr>
<th>Problem</th>
<th>Prob. Planning System</th>
<th>FO-Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>G2</td>
<td>P</td>
</tr>
<tr>
<td>bx c10 b5</td>
<td>438</td>
<td>184</td>
</tr>
<tr>
<td>bx c10 b10</td>
<td>376</td>
<td>0</td>
</tr>
<tr>
<td>bx c10 b15</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>bw b5</td>
<td>495</td>
<td>494</td>
</tr>
<tr>
<td>bw b11</td>
<td>479</td>
<td>466</td>
</tr>
<tr>
<td>bw b15</td>
<td>468</td>
<td>397</td>
</tr>
<tr>
<td>bw b18</td>
<td>352</td>
<td>–</td>
</tr>
<tr>
<td>bw b21</td>
<td>286</td>
<td>–</td>
</tr>
</tbody>
</table>

G2: temp. logic w/ control knowledge; P: RTDP-based
J1: human-coded policy; J2: inductive FO policy iter.; J3: deterministic FF-replanner
Related Work

- **Direct value iteration:**
  - ReBel algorithm for RMDPs (KvOdR, 2004)
  - FOVIA algorithm for fluent calculus (KS, 2005)
  - First-order decision diagrams (JKW, 2006)
  - → all disallow $\forall$ quant., e.g., universal cond. effects

- **Sampling and/or inductive techniques:**
  - Approx. linear programming for RMDPs (GKGK, 2003)
  - Inductive policy selection using FO regression (GT, 2004)
  - Approximate policy iteration (FYG, 2004)
  - → sampled domain instantiations do not ensure generalization across all possible worlds
  - → nonetheless, these methods have worked well empirically
Conclusions and Future Work

- **Conclusions:**
  - Developed *domain-independent* linear-value approximation techniques / optimization for FOMDPs
  - Encouraging empirical results on ICAPS 2004 IPPC
  - 2nd place in ICAPS 2006 IPPC by # problems solved

- **Future work:**
  - Goal decomposition for complex ∀ rewards
    - (∀b,c. Dest(b,c) ⇒ BIn(b,c,s)) ∨ ∃b.Bin(b,Paris,s)
  - Online search to “patch-up” decomposition error
    - E.g., additive decomposition is inadequate to solve some difficult problems in BlocksWorld
  - More expressive rewards
    - Σ_b (∀c. Dest(b,c) ⇒ BIn(b,c,s))