Practical Linear-value Approximation Techniques for First-order MDPs

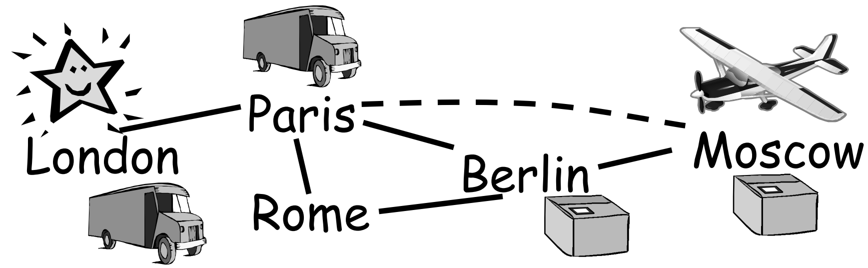
Scott Sanner & Craig Boutilier
University of Toronto

UAI 2006

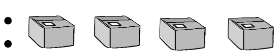
Why Solve First-order MDPs?

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

Box World:




```
(: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)))
```

- **Can solve a *ground MDP* for *each* domain instantiation:**
  - 3 trucks:  2 planes:  4 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 $\rightarrow$ 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 F:**
  - **Describe how action affects fluent** (like det. FO-DBN)
  - **Ex:** BIn(b,c,do(a,s)) ≡
    (1) Bin(b,c,s) AND a ≠ loadS(b,t)
    OR (2) for some t: a = unloadS(b,t) AND TIn(t,c,s)
- **Regression Operator:** Regr(φ) = φ′
  - **Takes a formula φ describing a *post-action* state**
  - **Uses SSAs to build φ′ 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…

rCase(s) =

$\forall b,c.\ Dest(b,c) \Rightarrow BIn(b,c,s)$	1
$\neg\ \forall b,c.\ Dest(b,c) \Rightarrow BIn(b,c,s)$	0

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

$\exists x.A(x)$	10
$\neg\exists x.A(x)$	20

$\oplus$

$\exists y.A(y)\wedge B(y)$	3
$\neg\exists y.A(y)\wedge B(y)$	4

=

$\exists x.A(x)\ \wedge\ \exists y.A(y)\wedge B(y)$	13
$\exists x.A(x)\ \wedge\ \neg\exists y.A(y)\wedge B(y)$	14
$\cancel{\neg\exists x.A(x)\ \wedge\ \exists y.A(y)\wedge B(y)}$	$\cancel{23}$
$\neg\exists x.A(x)\ \wedge\ \neg\exists y.A(y)\wedge B(y)$	24

  - 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(loadS(b,t) | load(b,t)) =

snow(s)	.1
$\neg$ snow(s)	.5

P(loadF(b,t) | load(b,t)) = 1 $\ominus$ P(loadS(b,t) | load(b,t))

- **First-order decision-theoretic regression (FODTR):**
  - **Given value fun** *vCase(s)* **and user action, produces first-order description of "Q-fun" (modulo reward)**

"Q-Fun" = FODTR[vCase(s), load(b,t)] =
Regr[vCase(after loadS…)] $\otimes$ P(loadS… | load…)
$\oplus$ Regr[vCase(after loadF…)] $\otimes$ P(loadF… | load…)

FOMDP Backup Operators

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

1) $B^{A(x)}[vCase(s)] = rCase(s) \oplus \gamma \cdot FODTR[vCase(s)]$

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

$\varphi(b,t)$	.9
$\neg\varphi(b,t)$	0

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

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

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

$\exists b,t.\ \varphi(b,t)$	.9
$\exists b,t.\ \neg\varphi(b,t)$	0

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

3) $B^{A}_{max}[vCase(s)] = \max(\ B^{A}[vCase(s)]\ )$

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

$\exists b,t.\ \varphi(b,t)$	.9
$\neg(\exists b,t.\ \varphi(b,t))$ $\wedge \exists b,t.\ \neg\varphi(b,t)$	0

Think of as $Q(A,s)$, *no* free vars and *no* overlap!

First-order Approx. Linear Prog. (FOALP)

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

$$vCase(s) = w_1 \cdot \begin{array}{|l|c|}\hline \exists b,c\ BIn(b,c,s) & 1 \\ \hline \neg\exists b,c\ BIn(b,c,s) & 0 \\ \hline \end{array} \oplus \ldots \oplus w_k \cdot \begin{array}{|l|c|}\hline \exists t,c\ TIn(t,c,s) & 1 \\ \hline \neg\exists t,c\ TIn(t,c,s) & 0 \\ \hline \end{array}$$

- **Reduces MDP solution to finding good weights… generalize <u>approx. LP</u> used by (van Roy, GKP, SP):**

Vars: $w_i;\ i \le k$

Minimize: $\sum_s \sum_{i=1..k} w_i \cdot bCase_i(s)$

Subject to: $0 \ge B^a_{max}[\oplus_{i=1..k} w_i \cdot bCase_i(s)] \ominus \oplus_{i=1..k} w_i \cdot bCase_i(s); \quad \forall a \in A, s$

- **FOALP issues resolved in (SB, 2005):**
  - ∞ **sum in objective:** We give principled approximation
  - ∞ **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:**
  - $\pi Case(s) = \max(\cup_{i=1..m} B^{A_i}[vCase(s)])$
  - **Each case partition should retain mapping to** $\boldsymbol{A_i}$
- **Now separate partitions in A_i-specific policies:**
  - $\pi Case_{A_i}(s) = \{ \text{ part} \in \pi Case(s) \text{ s.t. part} \rightarrow A_i \}$
  - **Specifies states where policy would apply** $\boldsymbol{A_i}$
- **FOAPI: Direct generalization of GKP (exact objective!)**
  - **Start w/** $w_i^0=0$, $\pi Case^0(s)$; **iterate LP soln until** $\pi^{j+1} = \pi^j$**:**

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

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

Subject to: $\phi^{(j+1)} \geq | \pi Case^j_a(s) \oplus B^a_{max}(\oplus_{i=1..k} w_i^{(j+1)} \cdot bCase_i(s))$
$\ominus \oplus_{i=1..k} w_i^{(j+1)} \cdot bCase_i(s)|$; $\forall a \in A, s$

- **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: $\exists b.Bin(b,P,s)$
  - Regr w/ unload: $\exists b.Bin(b,P,s) \vee (\exists b^*,t^*.TIn(t^*,P,s) \wedge On(b^*,t^*,s))$
- **Render basis *disjoint* from parents, will use later**
- **Iteratively solve FOMDP**
  - Retain all basis functions with wgt. > threshold τ
  - Generate new basis fns from retained set

Problems w/ Universal Reward

- **Universal rewards are difficult for FOMDPs, e.g.**
  - **Given reward:**

rCase(s)=

$\forall b,c.\ Dest(b,c) \Rightarrow BIn(b,c,s)$	1
$\neg$”	0

  - **Exact n-stage-to-go value function has form:**

vCase^n(s)=

$\forall b,c.\ Dest(b,c) \Rightarrow BIn(b,c,s)$	1
1 box not at dest	γ
…	…
n-1 boxes not at dest	γ^{n-1}

  - **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:**

  When reward in simple implicative form, solve for single goal with distinguished constants.

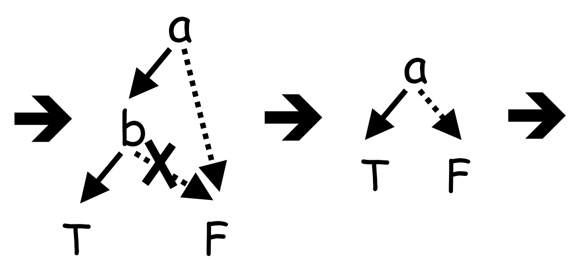
  - E.g., given: $\forall b,c.\ Dest(b,c) \Rightarrow BIn(b,c,s)$
  - Solve FOMDP for: $BIn(b^*,c^*,s)$
  - Given solution, gen. Q-funs $Q(A,s)_{\langle b^*,c^*\rangle}(s)$ for $\forall a \in A$

- **At run-time: Given concrete domain, e.g.**
  - Instantiation: $\{Dest(b_1,c_1), Dest(b_2,c_2), Dest(b_3,c_3)\}$
  - Let overall $Q(A,s) = Q(A,s)_{\langle b1,c1\rangle}(s) + Q(A,s)_{\langle b2,c2\rangle}(s) + Q(A,s)_{\langle b3,c3\rangle}(s)$ 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 B of basis functions: must examine $2^{|B|}$ case partitions in constraint generation
  - But for any pairwise disjoint set B' of basis functions, need examine only $|B'|$ 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: $(\exists x\ A(x)) \wedge (\exists x\ A(x)) \wedge (\exists x\ A(x) \wedge B(x))$

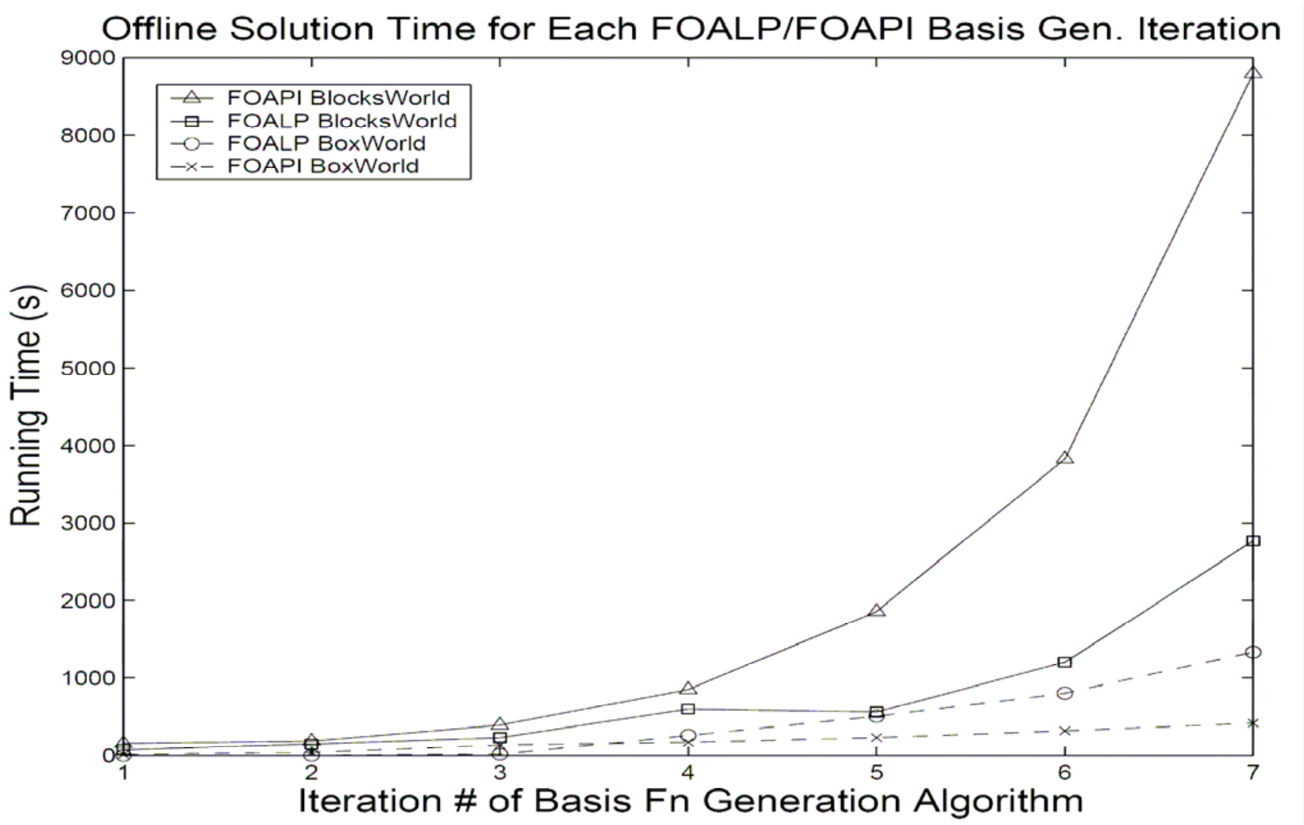
Prop Var	FOL Mapping	Impl.
a	$\exists x\ A(x) \wedge B(x)$	$a \Rightarrow b$
b	$\exists x\ A(x)$	$\neg b \Rightarrow \neg a$



➔ $\boxed{\exists x\ A(x) \wedge B(x)}$

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):**

Problem	Prob. Planning System					FO–	
	G2	P	J1	J2	J3	ALP	API
bx c10 b5	438	184	419	376	425	433	433
bx c10 b10	376	0	317	0	346	366	366
bx c10 b15	0	–	129	0	279	0	0
bw b5	495	494	494	495	494	494	490
bw b11	479	466	480	480	481	480	0
bw b15	468	397	469	468	0	470	0
bw b18	352	–	462	0	0	464	0
bw b21	286	–	456	455	459	456	0

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)
  - $\rightarrow$ 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)
  - $\rightarrow$ sampled domain instantiations do not ensure generalization across all possible worlds
  - $\rightarrow$ 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
  - 2^{nd} place in ICAPS 2006 IPPC by # problems solved

- **Future work:**
  - Goal decomposition for complex $\forall$ rewards
    - $(\forall b,c.\ Dest(b,c) \Rightarrow BIn(b,c,s)) \vee \exists b.Bin(b,Paris,s)$
  - Online search to "patch-up" decomposition error
    - E.g., additive decomposition is inadequate to solve some difficult problems in BlocksWorld
  - More expressive rewards
    - $\Sigma_b\ (\forall c.\ Dest(b,c) \Rightarrow BIn(b,c,s))$