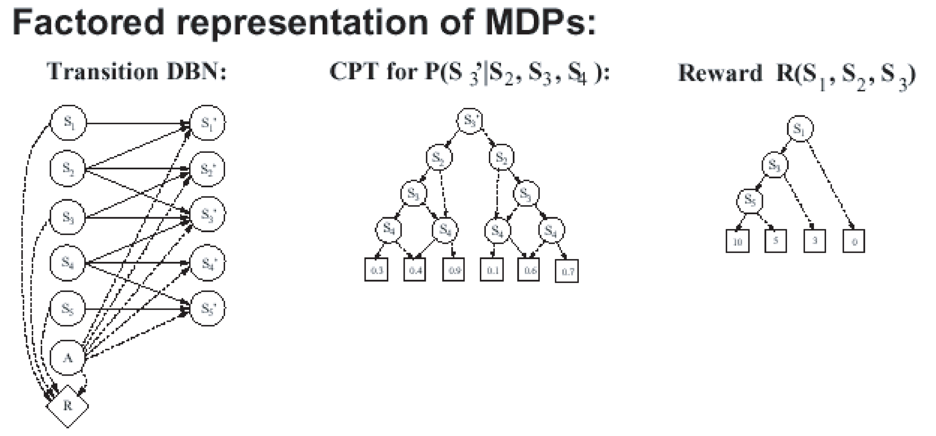
Approximate Linear Programming for First-order MDPs

Scott Sanner
University of Toronto
ssanner@cs.toronto.edu

Craig Boutilier
University of Toronto
cebly@cs.toronto.edu

1 Factored MDPs

- **Factored representation of MDPs:**



- **Bellman backup for factored MDPs:**

$$V^{t+1}(s_1,\ldots,s_n) = R(s_1,\ldots,s_n) + \gamma \max_a \sum_{s'_1 \ldots s'_n} \left[\prod_{i=1}^{n} P(s'_i|Parents(s'_i),a) \right] V^t(s'_1,\ldots,s'_n)$$

2 Approx. LP for Factored MDPs

- **Approximate $V(s_1,\ldots,s_n)$ with basis functions:**

$$V(s_1,\ldots,s_n) = w_1 B_1(s_x,\ldots,s_y) + \cdots + w_k B_k(s_z,\ldots,s_w)$$

- **Define backup operator:**

$$B^a(B_i)(s_x,\ldots,s_y) = \sum_{s'_x \ldots s'_y} \left[\prod_{i=1}^{n} P(s'_i|Par(s'_i),a) \right] B_i(s'_x,\ldots,s'_y)$$

- **Solve for approx. optimal value function using LP:**

Variables: $w_1,\ldots,w_k$

Minimize: $\sum_{s_1,\ldots,s_n} \sum_{i=1}^{k} w_i B_i(s_x,\ldots,s_y)$

Subject to: $0 \geq R(\cdots) + \gamma \sum_{i=1}^{k} w_i B^a(B_i)(\cdots) - \sum_{i=1}^{k} w_i B_i(\cdots) \; ; \; \forall a, s$

3 SitCalc and Stochastic Actions

- **Actions:** $upS(e)$, **Situations:** $s, do(upS(e), s)$, **Fluents:** $PAt(p,f,s)$
- **Successor-state axioms ($\Phi_F(\vec{x},a,s)$) for fluents F:**

$$\begin{aligned} PAt(p,f,do(a,s)) \equiv\ & (\exists e\ EAt(e,f,s) \wedge OnE(p,e,s) \wedge Dst(p,f) \wedge a = openS(e)) \vee \\ & PAt(p,f,s) \wedge \neg(\exists e\ EAt(e,f,s) \wedge \neg Dst(p,f) \wedge a = openS(e)) \end{aligned}$$

- **Regression:** $Regr(F(\vec{x}, do(a,s))) = \Phi_F(\vec{x},a,s)$

$$Regr(\neg\psi) = \neg Regr(\psi),\ Regr((\exists x)\psi) = (\exists x) Regr(\psi)$$
$$Regr(\psi_1 \wedge \psi_2) = Regr(\psi_1) \wedge Regr(\psi_2)$$

- **Stochastic actions decompose into deterministic actions:**

$$pCase(openS(e), open(e), s) = case[\neg old(e) : 0.9;\ old(e) : 0.7]$$
$$pCase(openF(e), open(e), s) = case[\neg old(e) : 0.1;\ old(e) : 0.3]$$

4 First-order MDPs (FOMDPs)

- **Represent *reward* and *value* functions using cases:**

$$rCase(s) = case[\forall p, f PAt(p,f,s) \supset Dst(p,f), 10 \; ; \; \neg``, 0]$$

- **Define operations $\{\oplus, \otimes, \ominus\}$ on cases:**

$$\begin{array}{|l|}\hline \psi_1 : v_1 \\ \hline \neg\psi_1 : v_2 \\ \hline \end{array} \oplus \begin{array}{|l|}\hline \psi_2 : v_3 \\ \hline \neg\psi_2 : v_4 \\ \hline \end{array} = \begin{array}{|ll|}\hline \psi_1 \wedge \psi_2 & : v_1 + v_3 \\ \hline \psi_1 \wedge \neg\psi_2 & : v_1 + v_4 \\ \hline \neg\psi_1 \wedge \psi_2 & : v_2 + v_3 \\ \hline \neg\psi_1 \wedge \neg\psi_2 & : v_2 + v_4 \\ \hline \end{array}$$

- **Define first-order decision-theoretic regression:**

$$FODTR(vCase(s), A(\vec{x})) = \gamma\, [\oplus_j \{pCase(n_j(\vec{x}), s) \otimes Regr(vCase(do(n_j(\vec{x}), s)))\}]$$

5 Symbolic Dynamic Programming for FOMDPs

- **Define a free-variable backup operator $B^{A(\vec{x})}$:**

$$B^{A(\vec{x})}(vCase(s)) = rCase(s) \oplus \gamma\ FODTR(vCase(s), A(\vec{x}))$$

- **Define a quantified backup operator B^A:**

$$B^A(vCase(s)) = rCase(s) \oplus \gamma\ \exists\vec{x}\ FODTR(vCase(s), A(\vec{x}))$$

- **Now can generalize Bellman equation for FOMDPs:**

$$vCase^{t+1}(s) = \max_A \gamma \cdot B^A(vCase^{t+1}(s))$$

6 Approximate LP for FOMDPs I

- **Represent $vCase(s)$ as sum of weighted basis functions:**

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

- **Redefine free-variable backup operator $B^{A(\vec{x})}$:**

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

- **Redefine quantified backup operator B^A where F are basis functions affected by action, N are not affected:**

$$B^A(\oplus_i\ w_i \cdot bCase_i(s)) = rCase(s) \oplus (\oplus_{i \in N}\ w_i\ bCase_i(s)) \oplus \exists\vec{x}\ (\oplus_{i \in F}\ w_i\ FODTR(bCase_i(s), A(\vec{x})))$$

Not all fluents affected by action, so retains additivity!

7 Backup Operator Example

- **Given reward and basis function case representation:**

$$\begin{aligned} rCase(s) &= case[\ \forall p, f\ PAt(p,f,s) \supset Dst(p,f) : 10 \; ; \neg `` : 0\ ] \\ vCase(s) &= w_1 \cdot case[\ \exists p, f\ PAt(p,f,s) \wedge \neg Dst(p,f) : 1 \; ; \neg `` : 0\ ] \oplus \\ & \quad w_2 \cdot case[\ \exists p, f, e\ Dst(p,f) \wedge OnE(p,f,s) \wedge EAt(e,f,s), 1 \; ; \neg ``, 0\ ] \end{aligned}$$

- **Apply $B^{down(x)}$ to obtain backup with free variable:**

$$\begin{aligned} B^{down(x)}(vCase(s)) = &\ case[\ \forall p, f\ PAt(p,f,s) \supset Dst(p,f) : 10 \; ; \neg `` : 0\ ] \\ & \oplus \gamma\ w_1 \cdot case[\ \exists p, f\ PAt(p,f,s) \wedge \neg Dst(p,f) : 1 \; ; \neg `` : 0\ ] \\ & \oplus \gamma\ w_2 \cdot case[\ \exists p, f, e\ Dst(p,f) \wedge OnE(p,f,s) \wedge \\ & \quad ((EAt(e,f,s) \wedge e \neq x) \vee (EAt(e, fa(f), s) \wedge e = x)) : 1 \; ; \neg `` : 0\ ] \end{aligned}$$

- **Quantify and maximize over all possible actions to obtain B^{down}:**

$$\begin{aligned} B^{down}(vCase(s)) = &\ case[\ \forall p, f\ PAt(p,f,s) \supset Dst(p,f) : 10 \; ; \neg `` : 0\ ] \\ & \oplus \gamma\ w_1 \cdot case[\ \exists p, f\ PAt(p,f,s) \wedge \neg Dst(p,f) : 1 \; ; \neg `` : 0\ ] \\ & \oplus \gamma\ w_2 \cdot case[\ \exists x, p, f, e\ Dst(p,f) \wedge OnE(p,f,s) \wedge \\ & \quad ((EAt(e,f,s) \wedge e \neq x) \vee (EAt(e, fa(f), s) \wedge e = x)) : 1 \; ; \\ & \quad \neg `` \wedge \exists x\ \forall p, f, e\ \neg Dst(p,f) \vee \neg OnE(p,f,s) \vee \\ & \quad ((\neg EAt(e,f,s) \vee e = x) \wedge \\ & \quad (\neg EAt(e, fa(f), s) \vee e \neq x)) : 0\ ] \end{aligned}$$

8 Approximate LP for FOMDPs II

- **Generalize approximate LP from propositional case:**

Variables: $w_i \; ; \; \forall i \leq k$

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

Subject to: $0 \geq B^A(\oplus_{i=1}^{k}\ w_i \cdot bCase_i(s)) \ominus (\oplus_{i=1}^{k}\ w_i \cdot bCase_i(s)) \; ; \; \forall A, s$

- **Objective ill-defined (infinite), need to redefine:**

$$\sum_s \sum_{i=1}^{k} w_i \cdot bCase_i(s) = \sum_{i=1}^{k} w_i \sum_s bCase_i(s) \sim \sum_{i=1}^{k} w_i \sum_{\langle \phi_j, t_j \rangle \in bCase_i} \frac{t_j}{|bCase_i|}$$

Preserves intent of original approx. LP formulation!

9 First-order Constraint Generation

- **Constraints are of the form:**

$$\begin{aligned} 0 &\geq case_1(s) \oplus \cdots \oplus case_j(s); \forall A, s \\ &\geq \max_s (case_1(s) \oplus \cdots \oplus case_j(s)) \; ; \forall A \end{aligned}$$

- **Infinite situations s so $\max_s$ appears to be impossible, but only finite number of constant-valued partitions of s!**
- **Thus, can solve LP efficiently using constraint generation:**
  1. Initialize LP with $\vec{w} = \vec{0}$ and empty constraint set
  2. For all $a \in A$, find maximally violated constraint c_a using *first-order* cost network max, add c_a to LP constraint set
  3. Solve LP, if solution $\vec{w}$ not within tolerance, goto step 2

10 Constraint Generation Example

Example of finding maximal violation for the following first-order LP constraint:

$$0 \geq \max_s \left(\begin{array}{|ll|}\hline \forall p, f\ Dst(p,f) \supset PAt(p,f,s) & : 10 \\ \hline \neg " & : 0 \\ \hline \end{array} \oplus \begin{array}{|ll|}\hline \exists p, f\ Dst(p,f) \wedge \neg PAt(p,f,s) : & w_1 \\ \hline \neg " & : -w_1 \\ \hline \end{array} \oplus \begin{array}{|ll|}\hline \exists p, e OnE(p,e,s) & : w_2 \\ \hline \neg " & : 0 \\ \hline \end{array} \right)$$

Assume last LP solution was $w_1 = 2$ and $w_2 = 1$. Evaluate weights:

$$0 \geq \max_s \left(\begin{array}{|ll|}\hline \{\neg Dst(p,f) \vee PAt(p,f,s)\} & : 10 \\ \hline \{Dst(c_1,c_2), \neg PAt(c_1,c_2,s)\} & : 0 \\ \hline \end{array} \oplus \begin{array}{|ll|}\hline \{Dst(c_3,c_4), \neg PAt(c_3,c_4,s)\} & : 2 \\ \hline \{\neg Dst(p,f) \vee PAt(p,f,s)\} & : -2 \\ \hline \end{array} \oplus \begin{array}{|ll|}\hline \{OnE(c_5,c_6,s)\} & : 1 \\ \hline \{\neg OnE(p,e,s)\} & : 0 \\ \hline \end{array} \right)$$

Given relation elimination order: PAt, Dst, OnE. Start by eliminating PAt: take cross-sum of case statements with PAt, resolve clauses in partition, and cross off any residual clauses with PAt.

$$0 \geq \max_s \left(\begin{array}{|ll|}\hline \{\cancel{\neg Dst(p,f) \vee PAt(p,f,s)}, Dst(c_3,c_4), \cancel{\neg PAt(c_3,c_4,s)}, \emptyset\} & : 12 \\ \hline \{\cancel{\neg Dst(p,f) \vee PAt(p,f,s)}\} & : 8 \\ \hline \{Dst(c_1,c_2), Dst(c_3,c_4), \cancel{\neg PAt(c_1,c_2,s)}, \cancel{\neg PAt(c_3,c_4,s)}\} & : 2 \\ \hline \{\cancel{\neg Dst(p,f) \vee PAt(p,f,s)}, Dst(c_1,c_2), \cancel{\neg PAt(c_1,c_2,s)}, \emptyset\} & : -2 \\ \hline \end{array} \oplus \begin{array}{|ll|}\hline \{OnE(c_5,c_6,s)\} & : 1 \\ \hline \{\neg OnE(p,e,s)\} & : 0 \\ \hline \end{array} \right)$$

Partitions with value 12 and -2 contain the empty clause (i.e. inconsistent), so remove them. Partition of value 8 dominates partition of value 2, so remove it. Yields simplified result:

$$0 \geq \max_s \left(\begin{array}{|l|}\hline \{\} : 8 \\ \hline \end{array} \oplus \begin{array}{|ll|}\hline \{OnE(c_5,c_6,s)\} & : 1 \\ \hline \{\neg OnE(p,e,s)\} & : 0 \\ \hline \end{array} \right)$$

Eliminating Dst and OnE will yield maximal consistent partition with value 9. This is a violation of the original constraint, so we generate the new linear constraint $0 \geq 10 + -w_1 + w_2$.

11 Experimental Results

- **Applied FOALP and other policies to elevator domain**
- **Eval accum., discounted reward @ step 50 for 5,10,15 floor domains and arrivals distributed according to $N(0.1, 0.35)$**
- **Compare to myopic/heuristic policies (avg 100 trials):**

Policy	5 Floors	10 Floors	15 Floors	Max Error
{ No Heuristics: Always Pickup }, { No Attended Conflict (A) }	116 ± 28	106 ± 27	105 ± 28	N/A
{ Prioritize VIP (V) }, { V,A }	115 ± 30	108 ± 30	107 ± 28	N/A
{ No Group Conflict (G) }, { A,G }	**125 ± 24**	**119 ± 21**	114 ± 20	N/A
{ V,G }, { V,A,G }	119 ± 30	114 ± 24	**115 ± 23**	N/A
Myopic 1-step Lookahead	118 ± 10	119 ± 9	120 ± 13	N/A
Myopic 2-step Lookahead	**123 ± 12**	**122 ± 5**	**120 ± 12**	N/A
FOALP { 1 & 2 Basis Functions }	133 ± 31	114 ± 32	112 ± 23	177
FOALP { 3 & 4 Basis Functions }	148 ± 26	129 ± 23	117 ± 23	159
FOALP { 5 Basis Functions }	147 ± 26	126 ± 17	120 ± 17	146
FOALP { 6 Basis Functions }	**154 ± 25**	**130 ± 19**	**125 ± 19**	92

12 Conclusions and Future Work

- **Conclusions:**
  - FOALP is an efficient approx. LP technique that exploits first-order structure *without grounding*
  - Implemented with highly optimized off-the-shelf software
  - Error bounds *apply equally to all domains*
  - Empirical results promising, but need more evaluation
- **Future work:**
  - Is uniform weighting the best approach?
  - Can we dynamically reweight based on Bellman error?