Approximate Linear Programming for First-order MDPs

Scott Sanner
University of Toronto

Craig Boutilier
University of Toronto

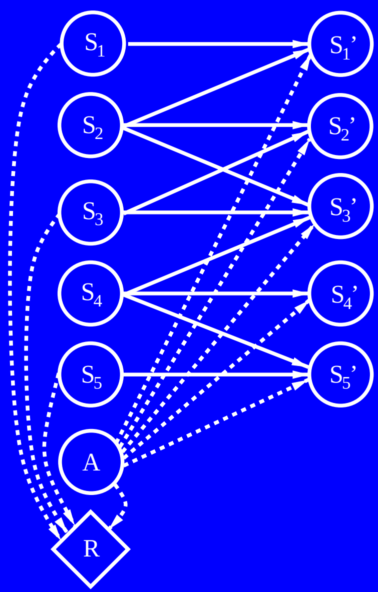
Outline

1. **Background for factored MDPs (ALP and constraint gen)**
2. **Background for first-order MDPs (FOMDPs)**
3. **Approximate linear programming (ALP) for FOMDPs**
   - Backup operators
   - First-order factored max (FOMax)
   - First-order constraint generation (FOCG)
4. **Experimental results**
5. **Conclusions and future work**

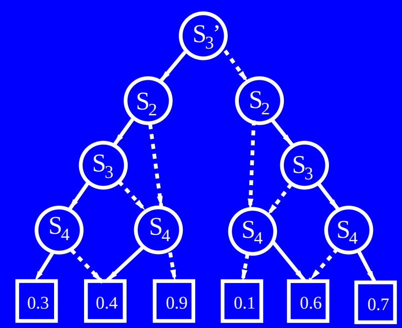
Factored MDPs

- **Factored representation of MDPs:**

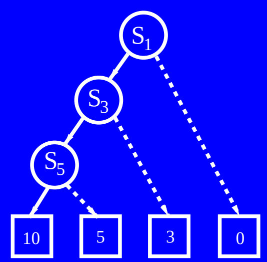
Transition DBN:



CPT for $P(S_3'|S_2, S_3, S_4)$:



Reward $R(S_1, S_2, S_3)$



- **Bellman backup for factored MDPs:**

$$V^{t+1}(s_1, \ldots, s_n) = R(s_1, \ldots, s_n) +$$

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

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

Constraint Generation

- **Constraints are of the form:**

$$\begin{aligned} 0 &\geq F(s_x, \ldots, s_y) + \cdots + F(s_z, \ldots, s_w) \;; \forall a, s \\ &\geq \max_{s_1 \ldots s_n} (F(s_x, \ldots, s_y) + \cdots + F(s_z, \ldots, s_w)) \;; \forall a \end{aligned}$$

- **Can find max efficiently in cost network!**

- **So use this to iteratively solve LP:**

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 cost network max, and add c_a to LP constraint set
3. Solve LP, if solution $\vec{w}$ not within tolerance, goto step 2

Situation Calculus

- **Deterministic actions:** $upS(e), downS(e), openS(e)$
- **Situations:** $S_0,\ do(upS(e), S_0),\ do(openS(e), do(upS(e), S_0))$
- **Fluents:** $OnE(p, e, s), PAt(p, f, s), EAt(e, f, s)$, **but not** $Dst(p, f)$
- **Successor-state axioms (**$\Phi_F(\vec{x}, a, s)$**) for fluents** F**:**

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

- **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 in SitCalc

- **Stochastic actions decompose into deterministic Nature's choice actions (usually success/failure):**

$$prob(openS(e), open(e), s) = 0.9$$

$$prob(openF(e), open(e), s) = 0.1$$

- **Use case notation to specify probability distribution:**

$$pCase(n_j(\vec{x}), A(\vec{x}), s) = case[\phi_1^j(\vec{x}, s), p_1^j; \cdots; \phi_n^j(\vec{x}, s), p_n^j]$$

- **Restate more complex version of above example:**

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

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

First-order MDPs (FOMDPs)

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

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

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

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

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

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

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:**

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

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 $B^{A(\vec{x})}$:**

$$\begin{aligned} B^{A(\vec{x})}(\oplus_i \, w_i \cdot bCase_i(s)) \quad &= \\ & rCase(s) \oplus (\oplus_i \; w_i \; FODTR(bCase_i(s), A(\vec{x}))) \end{aligned}$$

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

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

Not all fluents affected by action, so retains additivity!

Backup Operator Example

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

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

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

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

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

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

Approximate LP for FOMDPs II

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

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

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

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:**

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

$$\sim \quad \sum_{i=1}^{k} w_i \sum_{\langle \phi_j , t_j \rangle \in bCase_i} \frac{t_j}{|bCase_i|}$$

Preserves intent of original approx. LP formulation!

Constraint Generation II

- **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!**
- **Suggests a generalization of propositional cost network max and constraint generation.**

First-order Factored Max Algorithm

1. Convert the FOL formulae in each case partition to a set of CNF clauses.
2. For each relation $R \in R_1 \ldots R_n$ (under given ordering):
   (a) Remove all *case* statements in c containing R and store $\oplus$ in tmp.
   (b) Do the following for each partition in tmp:
      - Resolve all clauses on relation R, afterward remove remaining clauses containing R (ordered resolution).
      - If a resolvent of $\emptyset$ exists in this partition, remove this partition from tmp and continue.
      - Remove dominated partitions whose clauses are a superset of another partition with greater value.

   (c) Insert tmp back into c.
3. Return max of partitions remaining in c.

FO Constraint Gen. Algorithm

1. Initialize the weights: $w_i = 0 \; ; \; \forall i \leq k$
2. Initialize the LP constraint set: $C = \emptyset$
3. Initialize $C_{new} = \emptyset$
4. For each constraint inequality:
   (a) Calculate $\varphi = \arg\max_s \oplus_i \, case_i(s)$ using FOMAX.
   (b) If $eval(\varphi) \geq tol$, let c encode $0 \geq \oplus_i \, case_i(\varphi)$.
   (c) $C_{new} = C_{new} \cup \{c\}$
5. If $C_{new} = \emptyset$, terminate with w_i as the solution to this LP.
6. $C = C \cup C_{new}$
7. Re-solve the LP with updated constraints C, goto step 3.

FOALP Error Bounds

- **Based on Schuurmans and Patrascu (2001), we can also compute error bounds that *apply equally to all domains*:**

$$\max_s \; vCase^*(s) \ominus vCase_{\pi_{greedy}(\oplus_i \, w_i \cdot bCase_i)}(s)$$

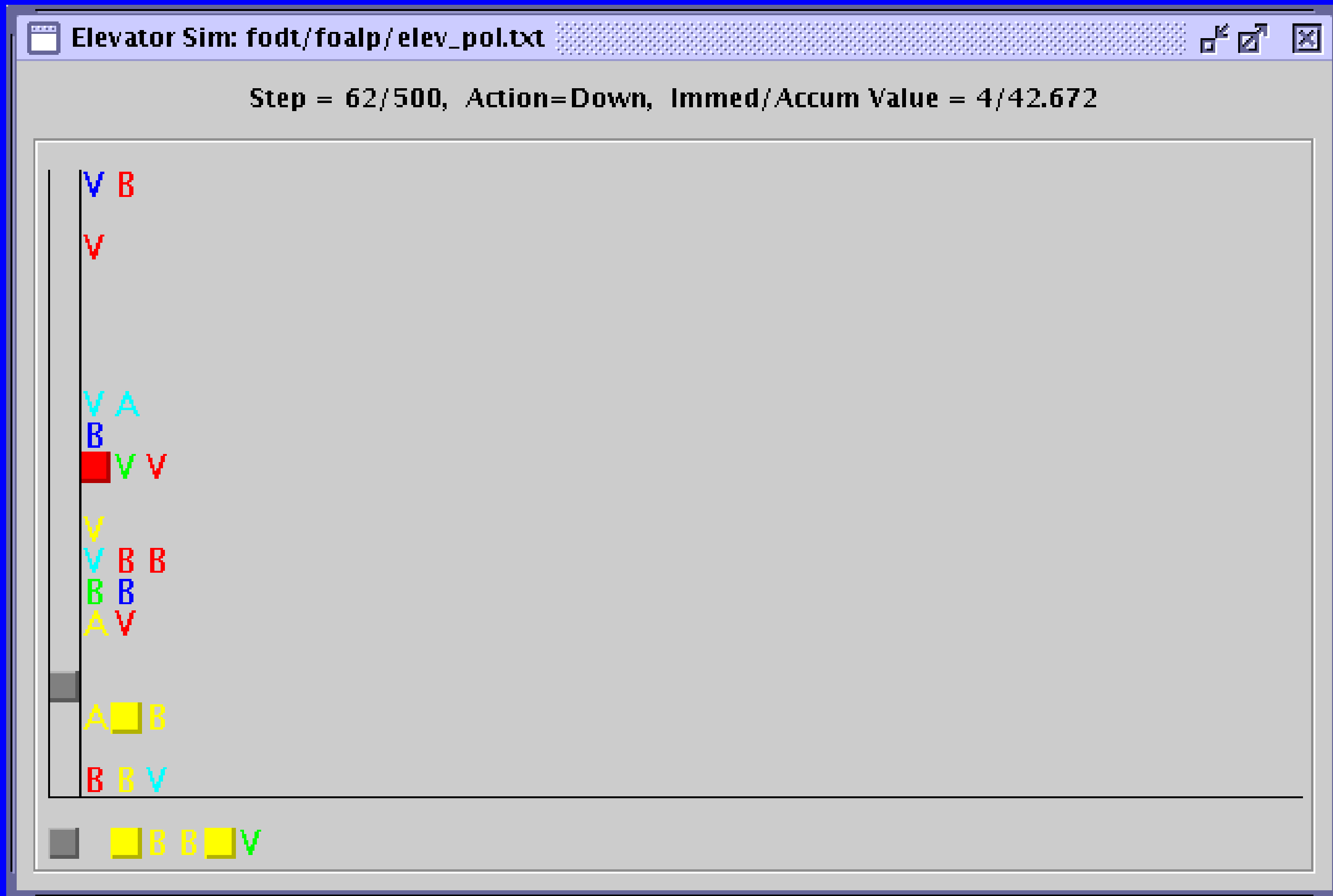
$$\leq \frac{\gamma}{1-\gamma} \max_s \min_A \Bigg[(\oplus_i \; w_i \cdot bCase_i(s)) \ominus B^A \, (\oplus_i \; w_i \cdot bCase_i(s)) \Bigg]$$

$$\leq \frac{\gamma}{1-\gamma} \min_A \max_s \Bigg[(\oplus_i \; w_i \cdot bCase_i(s)) \ominus B^A \, (\oplus_i \; w_i \cdot bCase_i(s)) \Bigg]$$

- **Final inequality can be efficiently computed via FOMax!**

Experimental Results I

- **Applied FOALP with FOCG to Elevator domain:**



- **Augmented with VIPs (V), Attended (A), Groups (Color)**

Experimental Results II

- **Elevator domain used additive reward criteria:**

$$
\begin{aligned}
&+2 : \forall p, f \; PAt(p, f, s) \supset Dst(p, f) \\
&+2 : \forall p, f \; VIP(p) \wedge PAt(p, f, s) \supset Dst(p, f) \\
&+4 : \forall p, e \; OnE(p, e, s) \wedge Attended(p) \supset \exists p_2 OnE(p_2, e, s) \\
&+2 : \forall p, f \; Dst(p, f) \wedge \neg PAt(p, f, s) \supset \exists e \; On(p, e, s) \\
&+2 : \forall p, f \; VIP(p) \wedge Dst(p, f) \wedge \neg PAt(p, f, s) \\
&\qquad \supset \exists e \; OnE(p, e, s) \\
&+8 : \forall p_1, p_2, g_1, g_2, e \; OnE(p_1, e, s) \wedge OnE(p_2, e, s) \\
&\qquad \wedge p_1 \neq p_2 \wedge Group(p_1, g_1) \wedge Group(p_2, g_2) \supset g_1 = g_2
\end{aligned}
$$

- **Also made basis functions for each of these formulae.**
- **Ran FOALP using 1-6 basis functions in given order.**

Experimental Results III

- **Implementation based on Vampire/CPLEX (5m - 2h)**
- **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

Related Work

- **SDP and ReBel require *difficult FOL simplification***
- **Both ALP for Rel MDP (Guestrin *et al*, 2003) and Approx. Policy Iteration (Fern *et al*, 2003) require *domain sampling***
- **Approx. Policy Iteration (Fern *et al*, 2003) and Gretton and Thiebaux (2004) use *inductive methods requiring substantial simulation***
- **Guestrin *et al* (2003) provide *PAC-bounds under assumption that prob. of domain falls off exponentially with size*; . . . in contrast, FOALP bounds *apply equally to all domains***

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 comparison
- **Future work:**
  - Is uniform weighting the best approach?
  - Can we dynamically reweight based on Bellman error?