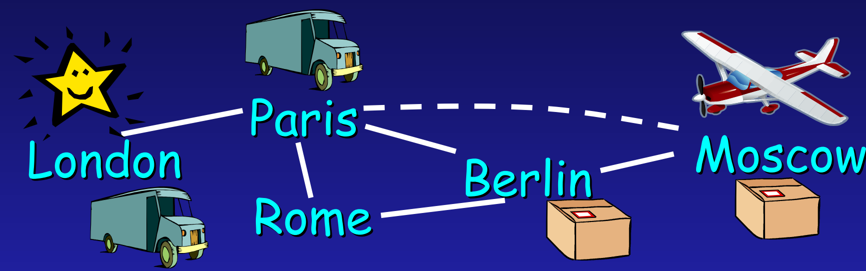
First-order Decision-theoretic Planning in Structured Relational Environments

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

Why First-order DT Planning?

- **Relational planning problem 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  (prob .9 (and (On ?b ?t) (not (BIn ?b ?c))))
```

- **Can solve *ground MDP* for *each* domain instance:**
  - 3 trucks: 2 planes: 3 boxes:
- **Or solve *first-order MDP* for *all* domains at once!**
  - Lift PPDDL problem to first-order MDP (FOMDP)
  - Solution makes value distinctions for *all* domains!

Talk Outline

1) **FOMDP Introduction**
   - Original definition, solution (BoutReiPr, IJCAI-01)
2) **Exploiting structure**
   - First-order decision diagrams
3) **Linear value approximation**
4) **Practical issues & results**
5) **Related work and conclusions**

FOMDP Introduction

Markov Decision Processes

- $\langle S,A,T,R \rangle$
  - S: finite set of states
  - A: finite set of actions
  - $T: S \times A \times S \rightarrow [0,1]$ transition function
  - $R: S \rightarrow \mathbb{R}$ reward function
- Policy $\pi: S \rightarrow A$
- Value function: $V(s) = E_\pi [\sum_{t=0} \gamma^t r^t | s]$
- $B^a[V(s)] = R(s) + \sum_{s' \in S} T(s,a,s')V(s')$
  - $Q^V(s,a) = B^a[V(s)]$
- $V^*(s) = \max_{a \in A} B^a[V^*(s)]$

Building Blocks of FOMDPs

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

$$rCase(s) = \begin{array}{|c|c|}\hline \forall b,c.\ Dest(b,c) \Rightarrow BIn(b,c,s) & 1 \\ \hline \neg \text{ “} & 0 \\ \hline \end{array}$$

- **Operators:** Define unary, binary case operations
  - E.g., can take “cross-sum” ⊕ (or ⊗, ⊖) of cases…

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

  - Can remove inconsistent elements, simplify

FOMDP Foundation: SitCalc

- **Deterministic Actions:** loadS(b,t), unloadS(b,t), …
- **Fluents (situated):** BIn(b,c,s), TIn(t,c,s), On(b,t,s)
- **Successor-state axioms (SSAs):**
  - Describe how actions affect fluents
  - Ex: BIn(b,c, after action a in situation s) ≡
(1) for some t: TIn(t,c,s) AND On(b,t,s)
AND a = unloadS(b,t)
OR (2) Bin(b,c,s) AND a ≠ loadS(b,t)
- **Regression Operator:** Regr[φ] = φ'
  - Takes a formula φ describing a *post-action* state
  - Uses SSAs to build φ' describing *pre-action* state

Stochastic Actions & FODTR

- **Stochastic actions using deterministic SitCalc:**
  - User's stochastic action: $A(x) = load(b,t)$
  - Nature's choice: $n(x) \in \{loadS(b,t), loadF(b,t)\}$
  - Probability distribution over Nature's choice:

$P(loadS(b,t) \mid load(b,t)) =$

snow(s)	.1
¬ snow(s)	.6

$P(loadF(b,t) \mid load(b,t)) =$

snow(s)	.9
¬ snow(s)	.4

- **First-order decision-theoretic regression**
  - FODTR = *expectation* of regression:

$$FODTR[vCase(s),A(x)] = \mathbf{E}_{P(n(x)|A(x))}\,[\,Regr[vCase(s),n(x)]\,]$$

Q-functions and Backups

- **FODTR almost gives us a Q-function**

$$\mathrm{FODTR}[vCase(s), unload(b,t)] = \begin{array}{|l|c|} \hline On(b,t,s) & 5 \\ \hline \neg On(b,t,s) & 0 \\ \hline \end{array}$$

  - FODTR specific to action variables
  - Also need to add reward, discount

- **Specify a backup operator for this**

$$B^{unload}[vCase(s)] = rCase(s) \oplus \gamma \begin{array}{|l|c|} \hline \exists b,t.\ On(b,t,s) & 5 \\ \hline \exists b,t.\ \neg On(b,t,s) & 0 \\ \hline \end{array}$$

  - Yields a first-order Q-function

Symbolic Dynamic Programming

- **What value if 0-stages-to-go?**
  - Obviously $V^0(s) = rCase(s)$
- **What value if 1-stage-to-go?**
  - We know value for each action

$V^1(s) =$

φ_1	9
else φ_3	3
else φ_4	1
else φ_2	0

φ_1	9
φ_2	0

$= B^{unload}[rCase(s)]$

φ_3	3
φ_4	1

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

  - Now just need max for every state
- **Value iteration:** (BoutReiPr, IJCAI-01)
  - Obtain V^{n+1} from V^n until $(V^{n+1} \ominus V^n) < \varepsilon$

Exploiting Structure

Exploiting Redundancy

- Many case operations in solutions

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

$\oplus$

$\exists y\ A(y)$	1
$\neg\ \exists y\ A(y)$	2

=

$\exists x\ A(x)$ ^ $\exists y\ A(y)$	11
~~$\exists x\ A(x)$ ^ $\neg\ \exists y\ A(y)$~~	~~12~~
~~$\neg\ \exists x\ A(x)$ ^ $\exists y\ A(y)$~~	~~21~~
$\neg\ \exists x\ A(x)$ ^ $\neg\ \exists y\ A(y)$	22

- Still have redundant formulae!
- Extract propositional structure

Prop Var	FOL Mapping
a	$\exists x\ A(x)$
b	$\exists x\ B(x)$

$\rightarrow$

a	10
$\neg$a	20

$\oplus$

a	1
$\neg$a	2

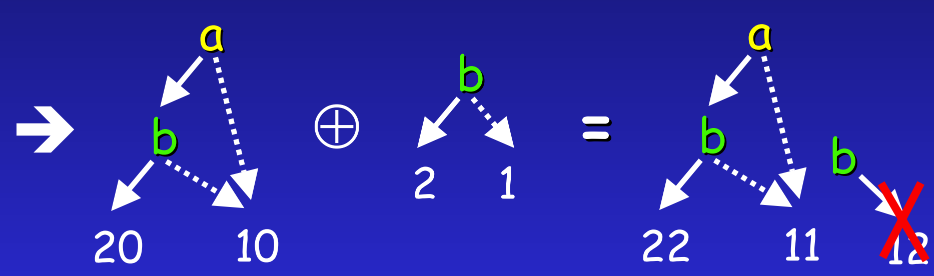
=

a	11
$\neg$a	22

Exploiting CSI

- **First-order ADDs**
  - Exploit (FO) context-specific independence

Prop Var	FOL Mapping
a	$\exists x\ A(x)$
b	$\exists x\ A(x) \wedge B(x)$

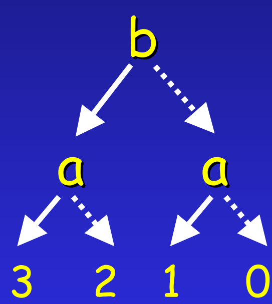


- **Some decision paths are unreachable**
  - Track implications
  - Avoid inconsistent paths
- **Use FOADDs to replace case & operations**

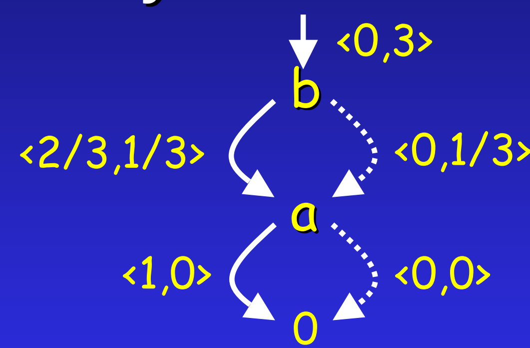
Exploiting Affine Structure

- **Replace ADDs with affine ADDs** (SanMc,IJCAI-05)
  - Best case: exp→linear reduction; never worse!

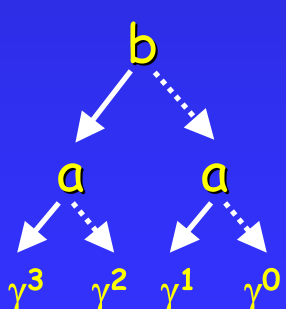
- **Example 1:** Additive reward/utility functions



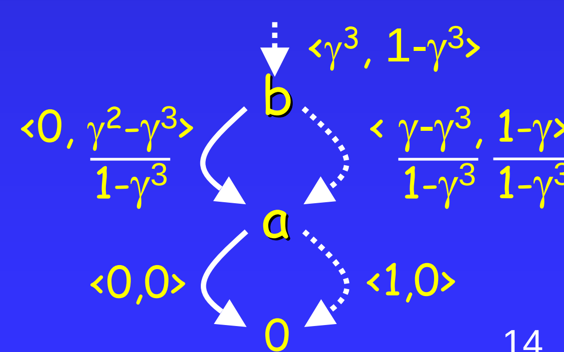
  - $R(a,b) = R(a) + R(b)$
    $= a + 2b$



- **Example 2:** Multiplicative value functions



  - $V(a,b) = V(a) \cdot V(b)$
    $= \gamma^{(a+2b)}; \gamma<1$



Linear Value Approximation

First-order Basis Functions

- **Approximate value with basis functions:**

$V(s) = w_1 \cdot$

$\exists b,c\ BIn(b,c,s)$	1
$\neg \exists b,c\ BIn(b,c,s)$	0

$\oplus\ w_2 \cdot$

$\exists t,c\ TIn(t,c,s)$	1
$\neg \exists t,c\ TIn(t,c,s)$	0

- **Reduces solution to finding good weights**
  - Weight projection $\Rightarrow$ no need for simplification
  - Only need to do consistency checking!

- **How to find weights?**
  - Formulate as optimization of LP

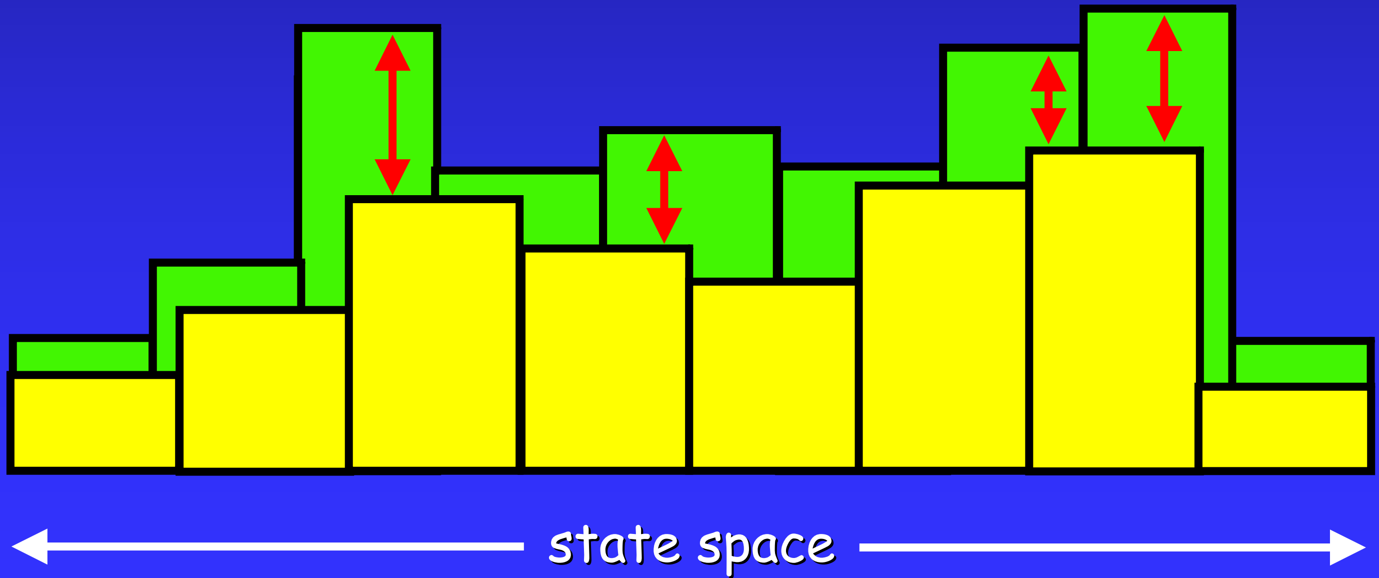
Approximate Linear Programming

- (SanBout, UAI-05) **FOALP: Generalize approximate LP solution of van Roy, GKP, SP**
  - Define: $V(s) = \oplus_{i=1..k} w_i \cdot bCase_i(s)$

    Vars: $w_i;\ i \leq k$

    Minimize: $\sum_{i=1..k} c_i \cdot w_i$

    Subject to: $V(s) \geq B^a[V(s)]\ ;\ \forall a \in A, s$



First-order Constraints

Technically

∞ constraints

- **Example constraint:**

$$0 \geq w_1 \cdot \begin{array}{|c|c|} \hline \exists b,c\ BIn(b,c,s) & 1 \\ \hline \neg \exists b,c\ BIn(b,c,s) & 0 \\ \hline \end{array} \oplus w_2 \cdot \begin{array}{|c|c|} \hline \exists t,c\ TIn(t,c,s) & 1 \\ \hline \neg \exists t,c\ TIn(t,c,s) & 0 \\ \hline \end{array} \ ; \forall s$$

- **Only finite *distinct* constraints**

$\exists b,c\ BIn(b,c,s) \wedge \exists t,c\ TIn(t,c,s)$	$0 \geq w_1 + w_2$
$\exists b,c\ BIn(b,c,s) \wedge \neg \exists t,c\ TIn(t,c,s)$	$0 \geq w_1$
$\neg \exists b,c\ BIn(b,c,s) \wedge \exists t,c\ TIn(t,c,s)$	$0 \geq w_2$
$\neg \exists b,c\ BIn(b,c,s) \wedge \neg \exists t,c\ TIn(t,c,s)$	$0 \geq 0$

$\cancel{w_1=1,\ w_2=1}$

$\cancel{w_1=-1,\ w_2=1}$

$w_1=-1,\ w_2=-1$ ☺

- **Solve via constraint generation**
  - Efficiently find max violated constraints
  - Generalize variable elimination to first-order

Policy Construction

- **Derive greedy policy π from $V(s)$:**

unload(b,t) $\leftarrow$	$\exists b,t\ \varphi_1$	9
load(b,t) $\leftarrow$	$\exists b,t\ \varphi_3$	3
load(b,t) $\leftarrow$	$\exists b,t\ \varphi_4$	1
unload(b,t) $\leftarrow$	$\exists b,t\ \varphi_2$	0

$\exists b,t\ \varphi_1$	9
$\exists b,t\ \varphi_2$	0

$= B^{unload}[V(s)]$

$\exists b,t\ \varphi_3$	3
$\exists b,t\ \varphi_4$	1

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

- **Now build $\pi Case_a(s)$ for $a \in \{unload, load\}$:**

$\pi Case_{unload}(s) =$

$\exists b,t\ \varphi_1$	9
$\neg \exists b,t\ \varphi_1 \wedge \neg \exists b,t\ \varphi_3 \wedge \neg \exists b,t\ \varphi_4 \wedge \exists b,t\ \varphi_2$	0

$\pi Case_{load}(s) =$

$\neg \exists b,t\ \varphi_1 \wedge \exists b,t\ \varphi_3$	3
$\neg \exists b,t\ \varphi_1 \wedge \neg \exists b,t\ \varphi_3 \wedge \exists b,t\ \varphi_4$	1

Approximate Policy Iteration

- **Basic Algorithm**

  1) Define $V^{(j)}(s) = \oplus_{i=1..k}\, w_i^{(j)} \cdot bCase_i(s)$,
     Initialize $j=0$, $\pi=\{any\ policy\}$
  2) Given policy π^j, find Bellman-error minimizing $w_i^{(j)}$ for $V^{(j)}(s)$
  3) Derive greedy policy π^{j+1} from $V^{(j)}(s)$
  4) If $\pi^{j+1} \neq \pi^j$ then let $j=j+1$ and go to step 2

- **LP for Bellman-error minimizing $w_i^{(j)}$:**

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

  Minimize: ϕ

  Subject to: $\phi \geq |\ \pi Case^{(j)}_a(s) \oplus V^{(j)}(s) \ominus B^a\,[V^{(j)}(s)]\ |;\ \forall a \in A, s$

- **Use $\pi Case_a(s)$ to enforce B^π** (GKPV, JAIR-02)

Practical Issues & Results

Generating Basis Functions

- **Where do basis functions come from?**
  - Major question for automation
  - Systematically build from FOL components?
  - Candidate space too large!
- **Idea (Gretton & Thiebaux, UAI-04) :**
  - Regressions from goal make good candidates
  - Guaranteed to have some value
  - Building blocks of value iteration
- **Iteratively solve FOMDP**
  - Retain basis functions with weight > threshold
  - Generate new basis functions from retained set

Problems w/ Universal Reward

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

rCase (s)=

∀b,c. Dest(b,c) ⇒ BIn(b,c,s)	1
¬"	0

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

vCase^n(s)=

∀b,c. Dest(b,c) ⇒ 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!
  - No compact representation (using piecewise-constant case statement)

Additive Goal Decomposition

- **Off-line solution for universal rewards:**
  - Given goal $\forall b,c.\ Dest(b,c) \Rightarrow BIn(b,c,s)$
  - Solve FOMDP for goal $BIn(b^*,c^*,s)$ to get $V(b^*,c^*,s)$
- **At run-time:**
  - Given concrete domain: $\{Dest(b_1,c_1), Dest(b_2,c_2)\}$
  - "Score" actions additively w.r.t. each goal

$$B^{unload}[V(b_1,c_1,s)] \qquad B^{unload}[V(b_2,c_2,s)]$$

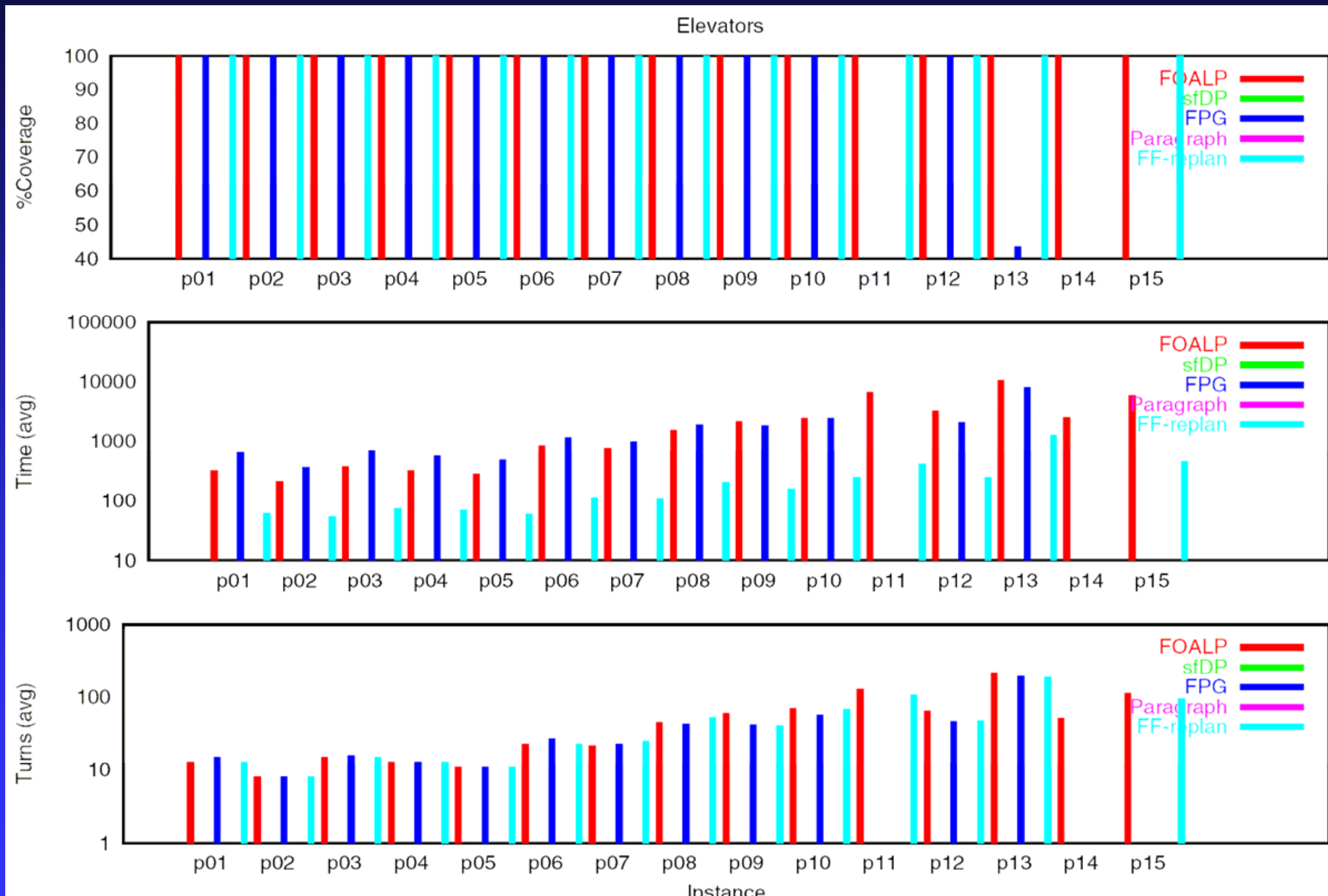
$$= \begin{array}{|c|c|}\hline \exists b,t\ \varphi_1 & 3 \\ \hline \exists b,t\ \varphi_2 & 1 \\ \hline \end{array} \oplus = \begin{array}{|c|c|}\hline \exists b,t\ \varphi_3 & 3 \\ \hline \exists b,t\ \varphi_4 & 1 \\ \hline \end{array}$$

$$\{unload(b_1,t_1)\rightarrow 3,\ unload(b_2,t_2)\rightarrow 1\} + \{unload(b_2,t_2)\rightarrow 3,\ unload(b_2,t_1)\rightarrow 1\} = \{unload(b_1,t_1)\rightarrow 3,\ \boxed{unload(b_2,t_2)\rightarrow 4,}\ unload(b_2,t_1)\rightarrow 1\}$$

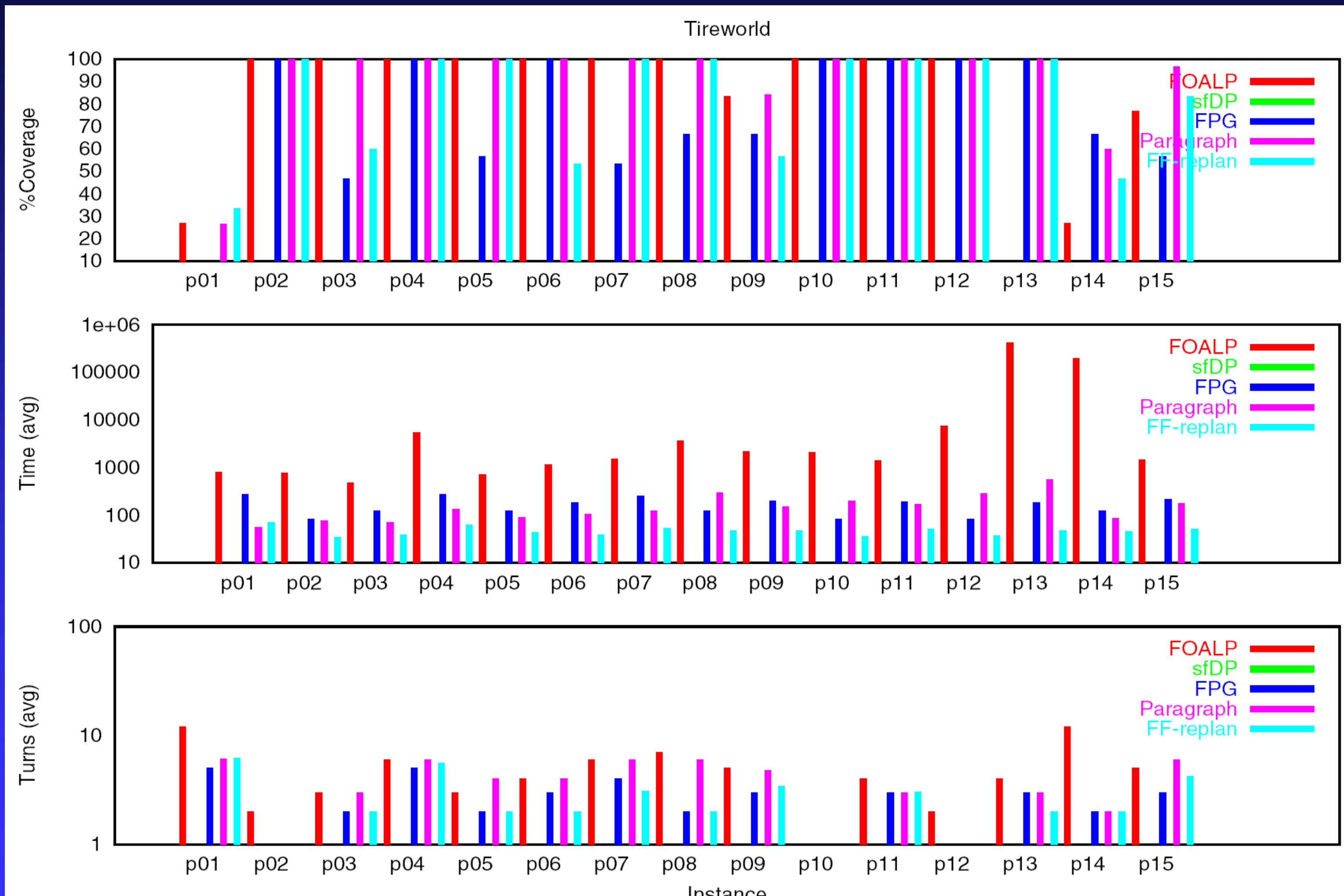
Optimizations

- **Enforce disjointness in basis functions:**
  - Can reduce search of $2^{|B|}$ case partitions to $|B|$
- **Exploiting implicit max in constraint generation**
  - Don't always need to enforce disjointness
  - Max-sum does this automatically
- **FOADDs for formulae simplification**
- **Huge cache of proved/unproved theorems**
  - Store FOL formulae in canonical format
- **Structural optimization in CNF transformation**
  - Introduce propositional literals to exploit DPLL in Vampire
- **Join-order optimizations in policy matcher**

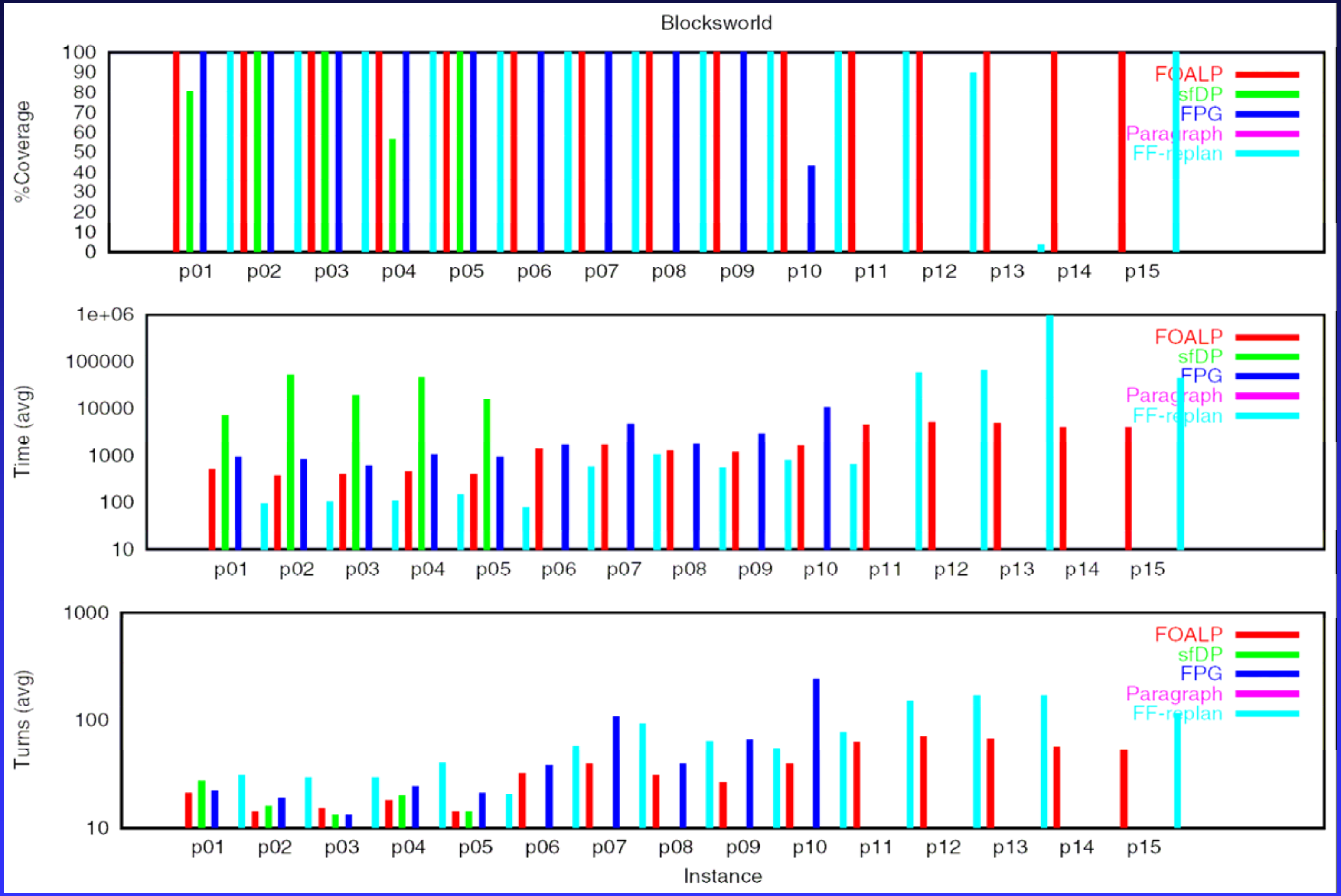
Results – ICAPS-06, Elevators



Results – ICAPS-06, Tire World



Results – ICAPS-06, Blocks World



Related Work

- **Direct “first-order” value iteration:**
  - ReBel algorithm for RMDPs (KvOdR, 2004)
  - FOVIA algorithm for fluent calculus (KS, 2005)
  - First-order decision diagrams (JKW, 2007)
  - $\rightarrow$ 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
  - $\rightarrow$ must restrict to “small” domain instances

Conclusions and Future Work

- **Conclusions:**
  - Managing structure in FOMDP solutions
  - Approximation techniques for FOMDPs
  - Range of practical implementation issues
  - Only *completely* first-order planner to date
    - $\Rightarrow$ 2nd place in ICAPS 2006 IPPC by # problems solved
- **Current & future work:**
  - Sum aggregator: $\Sigma_c \, \exists c \; BIn(b,c,s) : 1$; factored actions
  - Program constraints
  - Handling real-valued quantities, arithmetic
  - Exploiting topological structure
  - Integration with RL? First-order POMDPs?

Bibliography

(BRP, 2001) C. Boutilier, R. Reiter, and B. Price. (2001). *Symbolic Dynamic Programming for First-order MDPs*. IJCAI-01.

(SP, 2001) Dale Schuurmans and Relu Patrascu. (2001) Direct value approximation for factored MDPs. NIPS-2001.

(SM, 2005) S. Sanner and D. McAllester. (2005). *Affine Algebraic Decision Diagrams (AADDs) and their Application to Structured Probabilistic Inference*. IJCAI-05.

(SB, 2005) S. Sanner and C. Boutilier. (2005). *Approximate Linear Programming for First-order MDPs*. UAI-05.

(GKGK, 2003) C. Guestrin, D. Koller, C. Gearhart, and N. Kanodia. (2003). Generalizing Plans to New Environments in Relational MDPs. IJCAI-03.

(GT, 2004) C. Gretton and S. Thiebaux. (2004). Exploiting first-order regression in inductive policy selection. UAI-04.

(GKPV, 2002) C. Guestrin, D. Koller, R. Parr, and S. Venktaraman. Efficient solution methods for factored MDPs. JAIR, 2002.

Extra: Language Extensions

Sum & Product Aggregators

- **Often, reward scales with domain size:**

$$rCase(s) = \Sigma_c \begin{array}{|l|c|} \hline \exists c\ BIn(b,c,s) & 1 \\ \hline \neg\, \exists c\ BIn(b,c,s) & 0 \\ \hline \end{array}$$

- **Beyond expressive power of current FOMDP**
- **Need language extension for sum/product aggregators**
  - Functional as opposed to truth semantics
  - Like a quantifier (but indefinite ⊕)
- **Can extend symbolic dynamic programming, approximate solutions**
  - But tricky

Factored Actions

- **What if action has indefinite number of independent outcomes?**

$P(lost(b) \mid a) =$

large(b)	.0001
medium(b)	.0005
small(b)	.001

- **Then we get an indefinitely large joint distribution:**

$P(lost(b_1) \circ \ldots \circ lost(b_n) \mid a) = \prod_b$

large(b)	.0001
medium(b)	.0005
small(b)	.001

- **Have to exploit (FO) independence in solutions**
  - Then most of product will marginalize