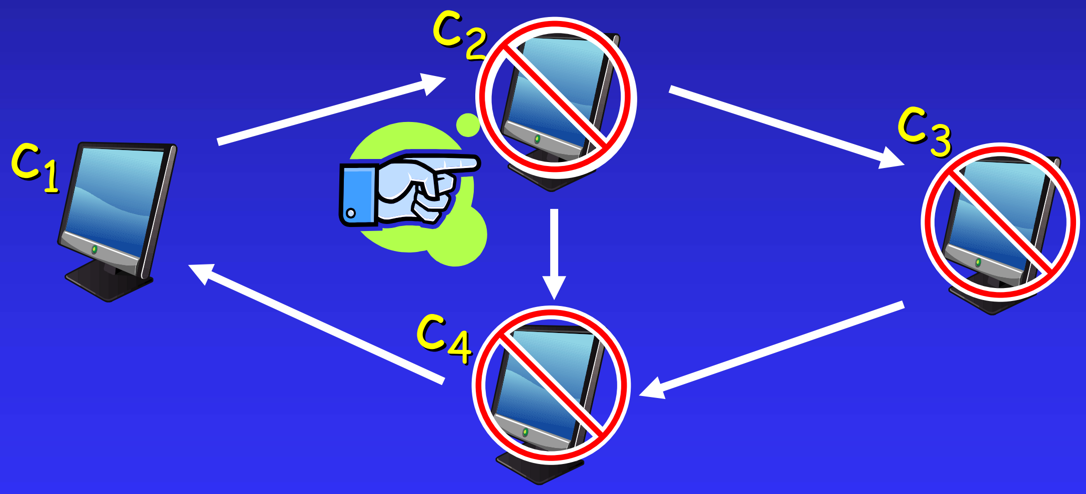
Approximate Solution Techniques for *Factored* FOMDPs

Scott Sanner & Craig Boutilier

University of Toronto

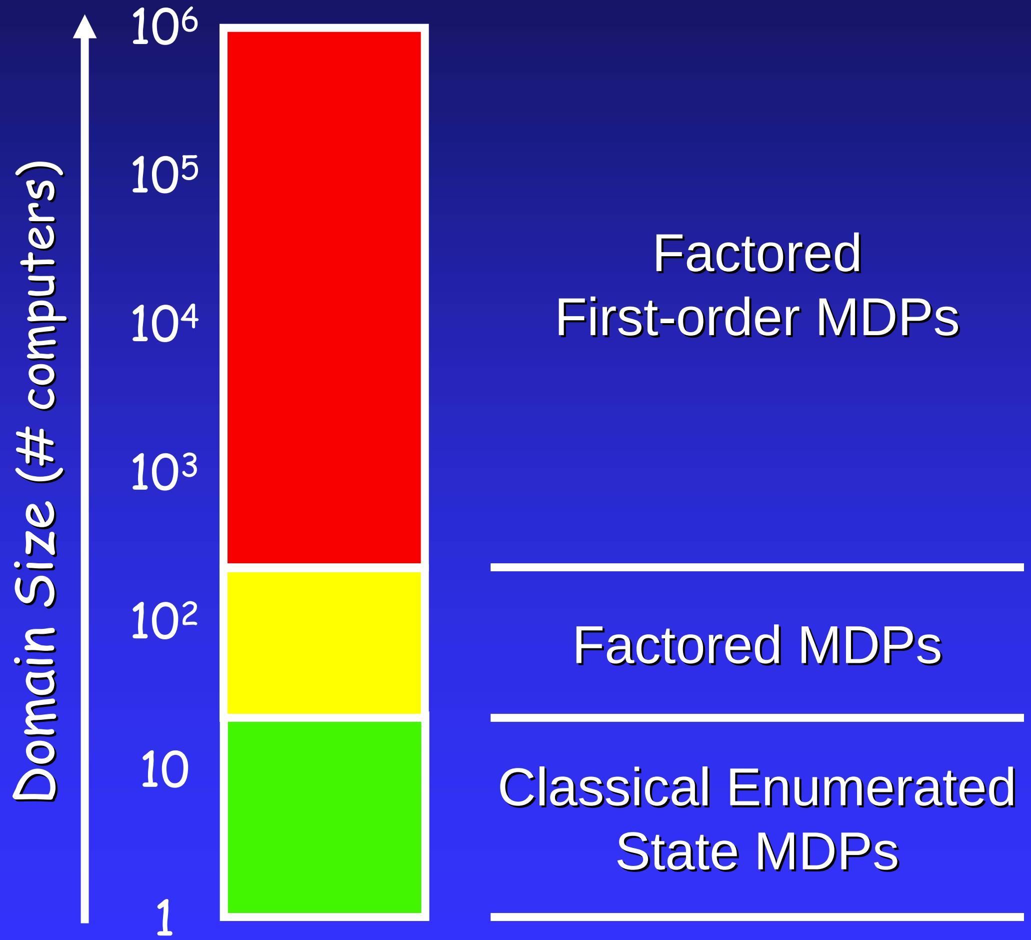
Motivating Example: SysAdmin MDP

- Have n computers $C = \{c_1, \ldots, c_n\}$ in a network
- State: each computer c_i is either "up" or "down"
- Transition: computer is "up" proportional to its state and # upstream connections that are "up"



- Action: manually reboot one computer
- Reward: +1 for every "up" computer

How to Solve SysAdmin?



- First-order MDPs cannot represent SysAdmin domain independently

  ⇒ need factored first-order MDP!
- State-of-the-art (factored) MDP solutions can scale only to ~140 computers

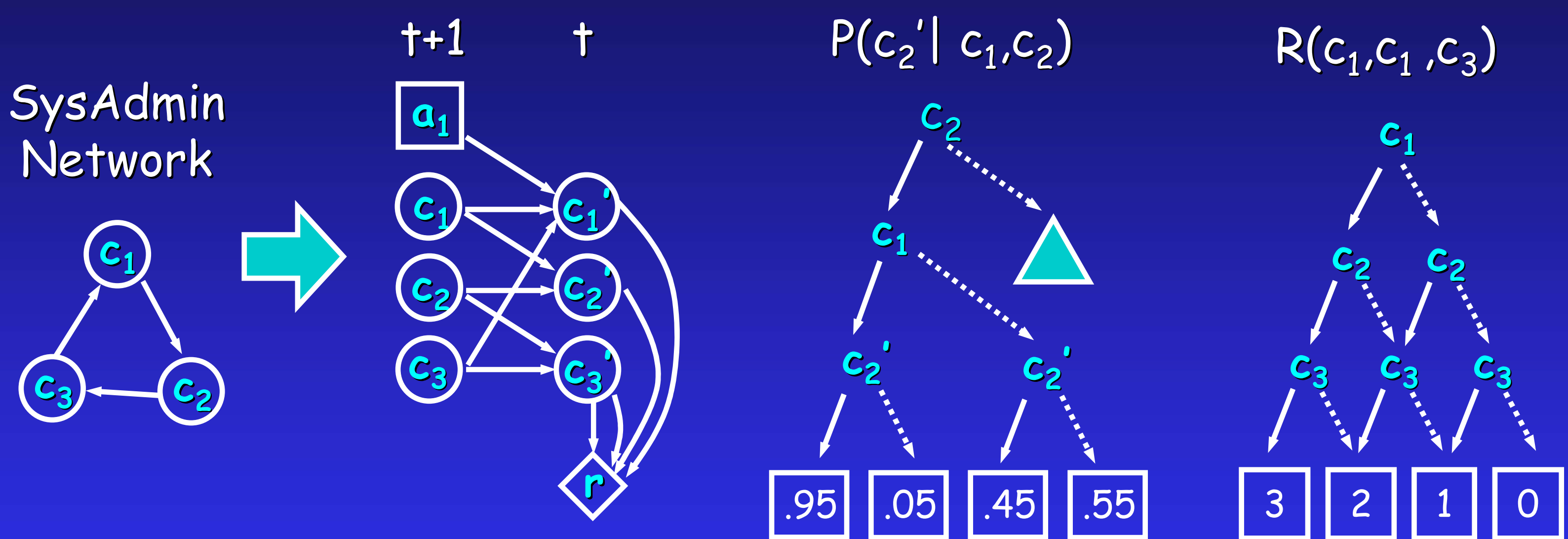
Background: Factored MDPs

Classical MDP Review

- **Representation:**
  - $\langle S, A, T, R \rangle$
    - S: finite set of states
    - A: finite set of actions
    - $T: S \times A \times S \rightarrow [0,1]$ transition function
    - $R: S \times A \rightarrow \mathbb{R}$ reward function
  - Policy $\pi: S \rightarrow A$
  - Value function: $V(s) = E_\pi [\sum_{t=0} \gamma^t r^t | s]$
- **Some Solution Methods:**
  - Value Iteration
  - Policy Iteration
  - (Approximate) Linear Programming

Factored MDPs and Value Iteration

- **Can use DBNs and decision diagrams:**



- **SPUDD** (HSHB, 1999)**: Value Iteration w/ DDs**

$$V^{t+1}(\vec{x}) = \max_{a \in A} \left\{ R(\vec{x}, a) + \gamma \sum_{\vec{x}'} \left[\prod_{i=1}^{n} P(x_i' | Par(x_i'), a) V^t(\vec{x}) \right] \right\}$$

Backup Operator

- **Define backup operator $B^a[.]$:**

$$B^a[V(\vec{x})] = \gamma \sum_{\vec{x}'} \left[\prod_{i=1}^{n} P(x_i' | Par(x_i'), a) V(\vec{x}') \right]$$

- **E.g., rewrite value iteration:**

$$V^{t+1}(\vec{x}) = \max_{a \in A} \left\{ R(\vec{x}, a) + B^a[V^t(\vec{x})] \right\}$$

- **$B^a[.]$ is a *linear* operator:**

$$B^a[V_1(\vec{x}) + V_2(\vec{x})] = B^a[V_1(\vec{x})] + B^a[V_2(\vec{x})]$$

Approximate LP for Factored MDPs

- **Linear-value function approximation:**

$$V(\vec{x}) = \sum_{j=1}^{k} w_j b_j(\vec{x})$$

- **Solve for weights using LP** (GKP,01; SP,01)**:**

Variables: $w_1, \ldots, w_k$

Minimize: $\sum_{\vec{x}} V(\vec{x})$

Exploit linearity!

Subject to: $0 \geq R(\vec{x}, a) + B^a[V(\vec{x})] - V(\vec{x})\ ;\ \forall a, \vec{x}$

Constraint Generation

- **Simple, general constraint format:**

$$0 \geq \{F_1(\vec{x}_1) + \ldots + F_m(\vec{x}_m)\} \;\; ; \; \forall a, \vec{x}$$

$$\geq \max_{\vec{x}} \{F_1(\vec{x}_1) + \ldots + F_m(\vec{x}_m)\} \;\; ; \; \forall a$$

- **Efficiently find max in cost network**
  - i.e., variable elimination

- **Iteratively solve LP with constraint generation**

1) Start with $w_i = 0$
2) Find max violated constraint for each a
3) If violations: add to LP and re-solve, goto (2)

Why Factored FOMDPs?

- **Back to SysAdmin…** $c_1 \longrightarrow \cdots\ c_i\ \cdots \longrightarrow c_n$
- **As a factored MDP**
  - Instantiate MDP for n = #computers
  - Specific solutions aside…
    - MDP repr. is $\Omega(n) \Rightarrow$ *any* solution is $\Omega(n)$
- **As a factored first-order MDP**
  - "Lift" MDP specification
  - "Lift" (approximate) solution
    - Solution O(sub-linear(n)) in struct. cases!

Contribution: Factored FOMDP Representation

First-order MDPs (FOMDPs)

- $\langle S,A,T,R \rangle$ for FOMDPs defined in terms of *cases*
  - E.g., possible *reward* in SysAdmin …

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

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

$$\begin{array}{|c|c|} \hline \varphi & 10 \\ \hline \neg\varphi & 20 \\ \hline \end{array} \oplus \begin{array}{|c|c|} \hline \phi & 3 \\ \hline \neg\phi & 4 \\ \hline \end{array} = \begin{array}{|c|c|} \hline \varphi \wedge \phi & 13 \\ \hline \varphi \wedge \neg\phi & 14 \\ \hline \neg\varphi \wedge \phi & 23 \\ \hline \neg\varphi \wedge \neg\phi & 24 \\ \hline \end{array}$$

Factored FOMDPs: Additive Reward

- **SysAdmin reward scales with domain size:**

$$rCase(s) = \begin{array}{|l|c|}\hline Run(c_1,s) & 1 \\ \hline \neg\, Run(c_1,s) & 0 \\ \hline \end{array} \oplus \cdots \oplus \begin{array}{|l|c|}\hline Run(c_n,s) & 1 \\ \hline \neg\, Run(c_n,s) & 0 \\ \hline \end{array}$$

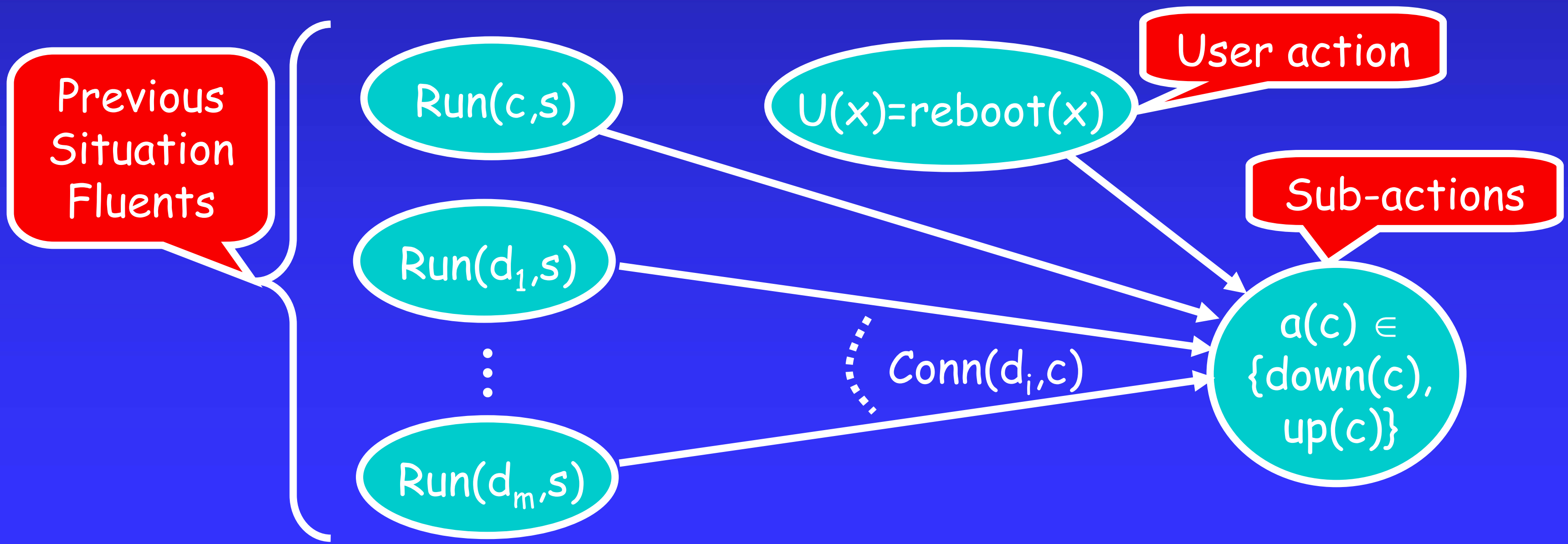
- **Beyond expressive power of current FOMDP**
- **Need language extension for $\sum$ aggregator:**

$$rCase(s) = \sum_{c \in C} \begin{array}{|l|c|}\hline Run(c,s) & 1 \\ \hline \neg\, Run(c,s) & 0 \\ \hline \end{array}$$

  - Semantics is just the expanded $\oplus$

Factored FOMDPs: Sub-actions

- **Usual FOMDP Frame Assumption:**
  - Def: *Anything not changed by action remains same*
  - Not true for factored FOMDPs (e.g., SysAdmin)
- **Remedy: Make "local" frame assumption (i.e., DBN)**
  - Specify first-order DBN for deterministic *sub-actions*



Factored FOMDP: Transition

- SysAdmin *sub-action* transition probability:

$$P(a(c) = up(c)|U(x) = reboot(x) \wedge x = c, s) = \boxed{\top \ : \ 1}$$

$$P(a(c) = up(c)|U(x) = reboot(x) \wedge x \neq c, s) =$$

$$\boxed{\begin{array}{ll} Run(c,s) & : \ 0.95 \\ \hline \neg Run(c,s) & : \ 0.05 \end{array}} \otimes \frac{1 + \sum_d \left(\boxed{\begin{array}{ll} Conn(d,c) \wedge Run(d,s) & : \ 1 \\ \hline \neg Conn(d,c) \vee \neg Run(d,s) & : \ 0 \end{array}} \right)}{1 + \sum_d \left(\boxed{\begin{array}{ll} Conn(d,c) & : \ 1 \\ \hline \neg Conn(d,c) & : \ 0 \end{array}} \right)}$$

- Need $\prod$ aggregator for *joint action* distribution:

$$P(a = a(c_1) \circ \cdots \circ a(c_n)|U(\vec{x})) = \prod_{c \in C} P(a(c)|U(\vec{x}), s)$$

SitCalc Extensions and Irrelevance

- **Upgrade SSAs to handle factored actions:**

$$Run(c, do(a,s)) \equiv a \sqsupseteq up(c) \vee Run(c,s) \wedge \neg a \sqsupseteq down(c)$$

- **Example regression in SysAdmin:**

$$Regr[Run(c_i,s)\ ;\ a = \cdots \circ up(c_i) \circ \cdots] \equiv \top$$

- **Observation: some sub-actions irrelevant in Regr[.]**
  - Def: *all sub-action outcomes B irrelevant to $\phi(s)$ iff*

$$\forall b \in B.\ Regr[\phi(s), b] \equiv \phi(s)$$

  - Need independence of sub-actions (not restrictive)
  - Allows irr. sub-actions to be dropped from Regr[.]

Recap: Factored FOMDPs

- **Reward:**
  - Can be expressed with $\sum$ aggregator
- **Transition Distribution:**
  - Decompose joint action into sub-actions
    - SSAs expressed in terms of sub-actions
    - Local distribution for each sub-action
  - Joint distribution uses $\prod$ aggregator
- **So far, just syntax…**
  - Real problem is exploiting syntactic structure
  - Key ideas: *irrelevance* and *operator linearity*

Contribution: Factored FOMDP Solutions

Factored FOMDP Backup Operator

- Define backup operator:

$$B^{U(\vec{x})}[vCase(s)] = \gamma \bigoplus_{a \in A} \left[P(a|U(\vec{x})) \otimes Regr[vCase(s), a] \right]$$

- Ex: SysAdmin $vCase^0(s) = rCase(s) = \sum_c \boxed{\begin{array}{l|l} Run(c,s) & : 1 \\ \hline \neg Run(c,s) & : 0 \end{array}}$

$$B^{reboot(x)}[vCase^0(s)] = \gamma \sum_c \Bigg[\bigoplus_{a_1 \in A(c_1), \ldots, a_n \in A(c_n)}$$

$$\left(\prod_{i=1}^{n} P(a_i|U) \right) \otimes \boxed{\sum_c} \boxed{\begin{array}{l|l} Regr[Run(c,s), a_1 \circ \cdots \circ a_n] & : 1 \\ \hline Regr[\neg Run(c,s), a_1 \circ \cdots \circ a_n] & : 0 \end{array}} \Bigg]$$

- Result after simplification:

$$B^{reboot(x)}[vCase(s)] = \gamma \sum_c P(up(c)|reboot(x))$$

SDP for Factored FOMDPs

- Complete symbolic dynamic programming step:

Recall Classical MDP: $V^1(s) = R(s) + \gamma \max_a B^a[V^0(s)]$

Upgrade to Factored FOMDP:

$$vCase^1(s) = rCase(s) \oplus \gamma \max \exists x. B^{reboot(x)}[vCase^0(s)]$$

$$= \sum_c \left(\boxed{\begin{array}{l|l} Up(c,s) & : \ 1 \\ \hline \neg Up(c,s) & : \ 0 \end{array}} \right) \oplus \gamma \max \exists x. \sum_c P(up(c)|reboot(x))$$

- Caveat: What to do with **max ∃x**?
- Workaround is to derive a policy (see GKP) to axiomatize optimal **x**
  - Need to add policy axioms on every SDP step
  - Exact representation blows up ☹

Linear-value Approximation

- **Approximate value w/ basis fn *classes*:**

$$vCase(s) = w_1 \cdot \sum_c \begin{array}{|c|c|}\hline \phi(c) & 1 \\ \hline \neg\phi(c) & 0 \\ \hline\end{array} \oplus w_2 \cdot \sum_c \begin{array}{|c|c|}\hline \varphi(c) & 1 \\ \hline \neg\varphi(c) & 0 \\ \hline\end{array}$$

- **Reduces solution to finding good weights**
  - Weight projection $\Rightarrow$ policy axioms don't accumulate
  - Only need to do consistency checking!
- **Where do basis functions come from?**
  - Use variant of ideas proposed in (GreThi, UAI-04)
- **How to find weights?**
  - We provide factored FOALP algorithm

Approximate Linear Programming

- **Factored First-order ALP:**

$$\text{Variables: } w_i \; ; \;\; \forall i \le k$$

$$\text{Minimize: } \sum_s vCase(s)$$

$$\text{Subject to: } 0 \ge rCase(s) \oplus B^{U(\vec{x})}[vCase(s)] \ominus vCase(s) \; ; \; \forall\, U(\vec{x}), s$$

- **Constraint generation solution:**
  - Constraints are always of the general form:

$$0 \ge \max_s \exists \vec{x} \left[\sum_c case_1(c, \vec{x}, s) \oplus \ldots \oplus \sum_c case_p(c, \vec{x}, s) \right]$$

  - Make a *domain size* assumption
  - Extend var. elimination techniques from FOPI (Poole, 2003; Braz, Amir, & Roth, 2005, 2006) to relation elimination

[Ch. 6] Some FOPI Techniques for Cases

- **Known Eliminations** (Poole, 2003; Braz, Amir, & Roth, 2005, 2006)
  - **(Partial) Inversion Elimination:**

    $\max \sum_c [case_1(c) \oplus case_2(c)]$ *where* $Rel(case_1) \cap Rel(case_2) = \varnothing$
  - **Counting Elimination:**

    $\max \sum_c \sum_{d \neq c} case(c,d)$
- **New Eliminations** (SanBout, 2007)
  - **Existential elimination (max elim. only):**

    $\max \exists x \sum_c [case_1(c,x) \oplus \cdots \oplus case_p(c,x)]$
  - **Linear elimination:**

    $\max \sum_{ci} case(c_i, c_{i+1})$

Existential Elimination

- **Need to compute:** max $\exists x \; \Sigma_c$ [case (c, x)]
  - where case(c, x) = $\begin{array}{|l|r|}\hline x = c & 10 \\ \hline x \neq c \wedge \ldots & 9 \\ \hline x \neq c \wedge \ldots & 0 \\ \hline \end{array}$
- **Introduce:**
  - $\sum_c eCase(c, s) = \sum_c \begin{array}{|lr|}\hline b(c) \supset b(next(c)) \; : & 0 \\ \hline b(c) \wedge \neg b(next(c)) : & -\infty \\ \hline \end{array}$
  - $b(c_1)$, $b(c_2)$, b(c3), …, $b(c_{n-1})$, $b(c_n)$
  - $\perp$ $\perp$ **T** **T** **T** **T**
- **Replace:** $(x = c) \equiv \neg b(c) \wedge b(next(c))$
- **Final constraint:**

$$0 \geq \max_s \sum_c \left[case_1(c, s) \oplus .. \oplus case_p(c, s) \oplus eCase(c, s) \right]$$

Linear Elimination $c_1 \longrightarrow \cdots \; c_i \; \cdots \longrightarrow c_n$

- Need to compute: $r(n) = \max_{c2 \ldots cn} \Sigma_{i=1 \ldots n}\, case(c_i, c_{i+1})$
  - where $case(c_i, c_{i+1}, s)$ =

c_i	c_{i+1}	
$\perp$	$\perp$	1
$\perp$	$\top$	-5
$\top$	$\perp$	-5
$\top$	$\top$	0

  - $r(2) = \max\ c_2$

c_1	c_2	
$\perp$	$\perp$	1
$\perp$	$\top$	-5
$\top$	$\perp$	-5
$\top$	$\top$	0

+

c_2	c_3	
$\perp$	$\perp$	1
$\perp$	$\top$	-5
$\top$	$\perp$	-5
$\top$	$\top$	0

=

c_1	c_3	
$\perp$	$\perp$	2
$\perp$	$\top$	-4
$\top$	$\perp$	-4
$\top$	$\top$	0

  - $r(4) = \max\ c_2, c_3, c_4$

c_1	c_3	
$\perp$	$\perp$	2
$\perp$	$\top$	-4
$\top$	$\perp$	-4
$\top$	$\top$	0

+

c_3	c_5	
$\perp$	$\perp$	2
$\perp$	$\top$	-4
$\top$	$\perp$	-4
$\top$	$\top$	0

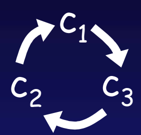
=

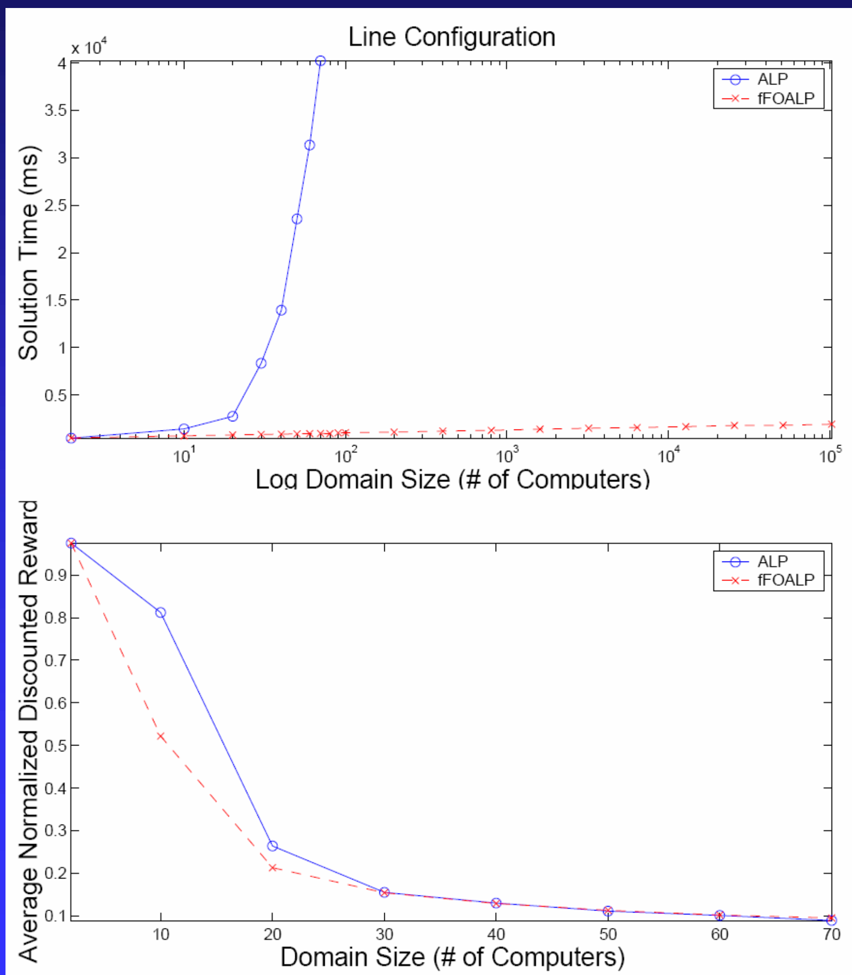
c_1	c_5	
$\perp$	$\perp$	4
$\perp$	$\top$	-2
$\top$	$\perp$	-2
$\top$	$\top$	0

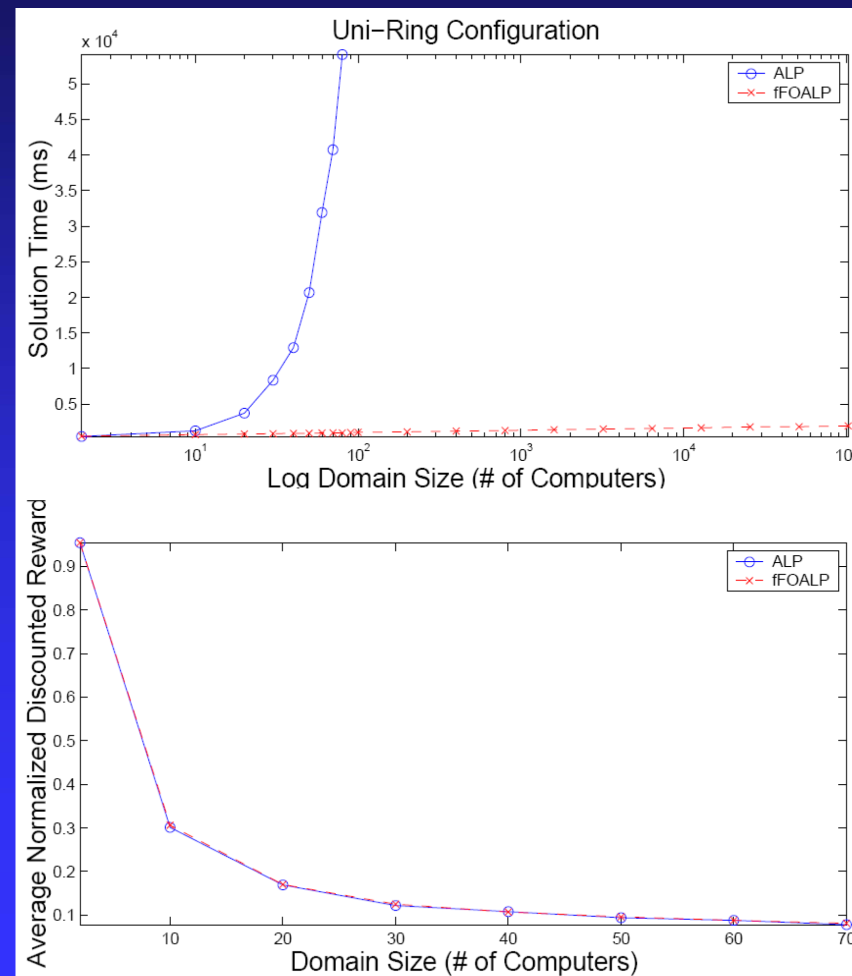
- Computation of $r(n)$ takes $O(\log(n))$!

Some Results for fFOALP vs. ALP

- Line Configuration $c_1 \rightarrow c_2 \rightarrow c_3$
- Unidirectional Ring







Conclusions

- Introduced factored FOMDP, "lifted" fFOALP solution
  - Exploited:
    - backup linearity
    - first-order irrelevance
    - FOPI for constraint generation
  - Solutions $O(\text{sub-linear}(n))$ in structured cases!
- What about *general* factored FOMDPs?
  - Key to efficient solution is FOPI techniques
  - FOPI specific to constraint structure induced by
    - basis function structure X
    - FO-DBN dynamics
  - Need to catalog "efficient" structures
  - Identify new structures exploitable by FOPI!