

Overview of Extensions II

Reinforcement Learning and Planning under Uncertainty

ANU COMP6460/4640, Sem 2, 2008

Scott Sanner

NICTA / RSISE

First.Last@nicta.com.au

Extensions to MDPs

**Continuous state,
Time (durative actions),
Multiagent**

Time-dependent MDPs

- State $S = (s, t)$ where s is discrete, $t \in [0, \infty)$
- Bellman equation for undiscounted case:

$$Q^{k+1}(s, t, a) = \sum_{s'} \int_{t'=t}^{\infty} \left(R(s, t, t', a) + \begin{cases} \text{Abs:} & P(t')P(s'|s, t, t', a)V^k(s', t') \\ \text{Rel:} & P(t' - t)P(s'|s, t, t', a)V^k(s', t') \end{cases} \right) dt'$$
$$V^{k+1}(s, t) = \max_{a \in A} Q^{k+1}(s, t, a)$$

- Relative transition model leads to convolution
 - Under certain restrictions, Q is piecewise linear in t
 - How to represent optimal value function?

If Piecewise-poly?

Continuous State & Action MDPs

- The state space is a vector of continuous and discrete variables $\vec{x} = (\vec{x}_c, \vec{x}_d)$
- Replace sums by integrals in Bellman eq:

$$Q^{t+1}(\vec{x}, a(\vec{z}))$$

$$= \gamma \left[\sum_{\vec{x}_d} \int_{\vec{x}_c} R(\vec{x}, a(\vec{z}), \vec{x}') + \left(\prod_{i=1}^m P(x'_i | \vec{x}, a(\vec{z})) \right) V^t(\vec{x}') d\vec{x}'_c \right]$$

$$V^{t+1}(\vec{x}) = \max_{a \in A} \max_{\vec{z}} \{ Q^{t+1}(\vec{x}, a(\vec{z})) \}$$

- Note: integrals don't always have closed-form
- Especially important to focus on restricted settings

Multiagent: Markov / Stochastic Games

- $A = A_1 \times \dots \times A_k$
- For $k=2$, zero-sum, a_1 max, a_2 min

$$V^t(s) = R(s) + \max_{a_1} \min_{a_2} \sum_{s'} T(s, a_1, a_2, s') V^{t-1}(s')$$

- If alternating turn where

$$T(s, a_1, a_2, s') = \sum_{s''} T(s, a_1, s'') T(s'', a_2, s')$$

Did not $\sum_{s''}$, for Tic-Tac-Toe, why?

then optimal policies are *deterministic*!

– can do value iteration with above Bellman eq.

RL for these Extensions

- We saw model-based Bellman backups for
 - Time-dependent MDPs
 - Continuous state and action MDPs
 - Multiagent MDPs
- What about (T,R) model-free RL solutions?
 - Recall: objective criterion is some expectation $E[\bullet]$
 - Model-based methods compute $E[\bullet]$ explicitly
 - But can always **sample** to compute $E[\bullet]$ instead
 - Full return (MC)
 - Bellman backup (TD)

Solving *Real* Problems

- Many real-world problems cannot be easily shoehorned into discrete MDP setting
- Often need to design setup for problem at hand
 - Formalize problem as accurately as possible!
- Exploit properties of domain to solve efficiently
 - *Continuous*: piecewise constant or linear functions?
 - *Factored*: use graphical models / decision diagrams
 - *Relational*: exploit logical abstraction...

First-order MDPs

**Exploit Logical Abstraction /
Symmetry in Relationally-
specified MDPs**

Planning Languages

- Common languages:

- STRIPS

- PDDL

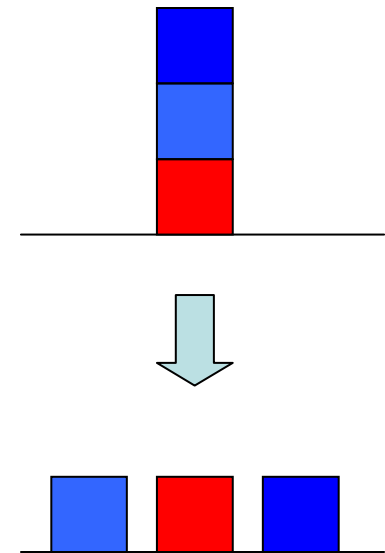
- more expressive than STRIPS

- for example, *universal* and *conditional* effects:

```
(:action put-all-blue-blocks-on-table
:parameters ( )
:precondition ( )
:effect (forall (?b)
         (when (Blue ?b)
              (not (OnTable ?b))))))
```

- General Game Playing (GGP)

- one or more agents

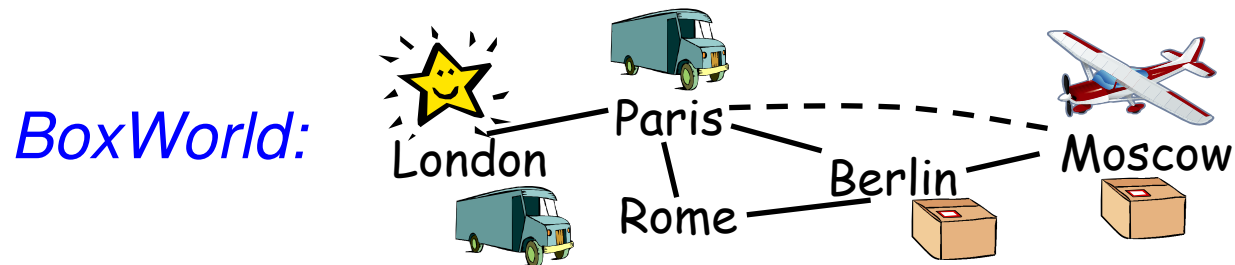


Benefits of Relational Languages




- STRIPS, PDDL, GGP are relational languages...
 - Refer to relational *fluents*:
 - e.g., *Bin(?b,?c)*, *OnTable(?b)*
 - specify relations between objects
 - change over time
 - Use first-order logic to specify...
 - action preconditions
 - action effects
 - goals / rewards
 - e.g., *(forall (?b ?c) ((Destination ?b ?c) \Rightarrow (Bin ?b ?c)))*
 - Are *domain-independent* and often compact!

How to Solve?

- Relational planning *problem*:



(*: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*))))

- Solve *ground problem* for *each domain instance*?
 - 3 trucks:  2 planes:  3 boxes: 
- Or solve lifted specification for *all domains* at once?

Full Specification: BoxWorld

- **Relational Fluents:** $BoxIn(Box, City), TruckIn(Truck, City), BoxOn(Box, Truck)$
- **Goal:** $[\exists Box : b. BoxIn(b, paris)]$
- **Actions:**
 - $load(Box : b, Truck : t)$:
 - * **Effects:**
 - $when [\exists City : c. BoxIn(b, c) \wedge TruckIn(t, c)] then [BoxOn(b, t)]$
 - $\forall City : c. when [BoxIn(b, c) \wedge TruckIn(t, c)] then [\neg BoxIn(b, c)]$
 - $unload(Box : b, Truck : t)$:
 - * **Effects:**
 - $\forall City : c. when [BoxOn(b, t) \wedge TruckIn(t, c)] then [BoxIn(b, c)]$
 - $when [\exists City : c. BoxOn(b, t) \wedge TruckIn(t, c)] then [\neg BoxOn(b, t)]$
 - $drive(Truck : t, City : c)$:
 - * **Effects:**
 - $when [\exists City : c_1. TruckIn(t, c_1)] then [TruckIn(t, c)]$
 - $\forall City : c_1. when [TruckIn(t, c_1)] then [\neg TruckIn(t, c_1)]$

Solving Ground BoxWorld

- Apply planner to BoxWorld grounded w.r.t. domain, e.g.,

- **Domain Object Instantiation:**

- $Box = \{box_1, box_2, box_3\}$, $Truck = \{truck_1, truck_2\}$, $City = \{paris, berlin, rome\}$

#states exponential
in #state-vars!

- **Ground Fluents (i.e., binary state variables):**

Exponential #state-vars in arity

- $BoxIn: \{BoxIn(box_1, paris), BoxIn(box_2, paris), BoxIn(box_3, paris), BoxIn(box_1, berlin), BoxIn(box_2, berlin), BoxIn(box_3, berlin), BoxIn(box_1, rome), BoxIn(box_2, rome), BoxIn(box_3, rome)\}$
 - $TruckIn: \{TruckIn(truck_1, paris), TruckIn(truck_1, berlin), TruckIn(truck_1, rome), TruckIn(truck_2, paris), TruckIn(truck_2, berlin), TruckIn(truck_2, rome)\}$
 - $BoxOn: \{BoxOn(box_1, truck_1), BoxOn(box_2, truck_1), BoxOn(box_3, truck_1), BoxOn(box_1, truck_2), BoxOn(box_2, truck_2), BoxOn(box_3, truck_2)\}$

- **Ground Actions:**

Exponential #actions in arity

- $load: \{load(box_1, truck_1), load(box_2, truck_1), load(box_3, truck_1), load(box_1, truck_2), load(box_2, truck_2)\}, load(box_3, truck_2)\}$
 - $unload: \{unload(box_1, truck_1), unload(box_2, truck_1), unload(box_3, truck_1), unload(box_1, truck_2), unload(box_2, truck_2)\}, unload(box_3, truck_2)\}$
 - $drive: \{drive(truck_1, paris), drive(truck_1, berlin), drive(truck_1, rome), drive(truck_2, paris), drive(truck_2, berlin), drive(truck_2, rome)\}$

Exponential in
#nested quantifiers

- **Goal:** $[BoxIn(box_1, paris) \vee BoxIn(box_2, paris) \vee BoxIn(box_3, paris)]$

A First-order Solution to BoxWorld

- Derive solution deductively at lifted PDDL level:

- if $(\exists b. \text{BoxIn}(b, \text{paris}))$
then do *noop*
- else if $(\exists b^*, t^*. \text{TruckIn}(t^*, \text{paris}) \wedge \text{BoxOn}(b^*, t^*))$
then do *unload*(b^*, t^*)
- else if $(\exists b, c, t^*. \text{BoxOn}(b, t^*) \wedge \text{TruckIn}(t, c))$
then do *drive*(t^*, paris)
- else if $(\exists b^*, c, t^*. \text{BoxIn}(b^*, c) \wedge \text{TruckIn}(t^*, c))$
then do *load*(b^*, t^*)
- else if $(\exists b, c_1^*, t^*, c_2. \text{BoxIn}(b, c_1^*) \wedge \text{TruckIn}(t^*, c_2))$
then do *drive*(t^*, c_1^*)
- else do *noop*

Optimal for *any*
domain instantiation!

- Great, but how do I obtain this solution?

See Sanner's thesis.

Mechanism Design:
Design Game to achieve
Social Objective Criterion

Changing the Task

- For most of the course, we assumed
 - Objective was to plan optimally given a model (or samples from it)
- Inverse RL did not fit this paradigm
 - Observed actions (π) from rational agent
 - Had to derive reward R consistent with actions
- What if we can **design** T , R in a game...
 - To achieve desired social behavior
 - Leads to economics field called Mechanism Design
 - Voting- and Auction-Theory important related fields

Mechanism Design: Playing God!

- Get to design the rules of a game T, R
 - To achieve some social objective
 - E.g., social welfare: sum of each agent's utility
 - Assume agents are rational utility maximizers
- Auctions:
 - Want the bidder with highest *true* valuation to win
 - But any bidder might lie in order to win
 - So want incentive compatibility
 - Truth-telling in your bid should be the optimal strategy
 - Thus highest bidder wins (maximizes social welfare)
 - Can do this through VCG-style 2nd-price mechanisms...

2nd-price Auctions

- **Ascending auction** (eBay)

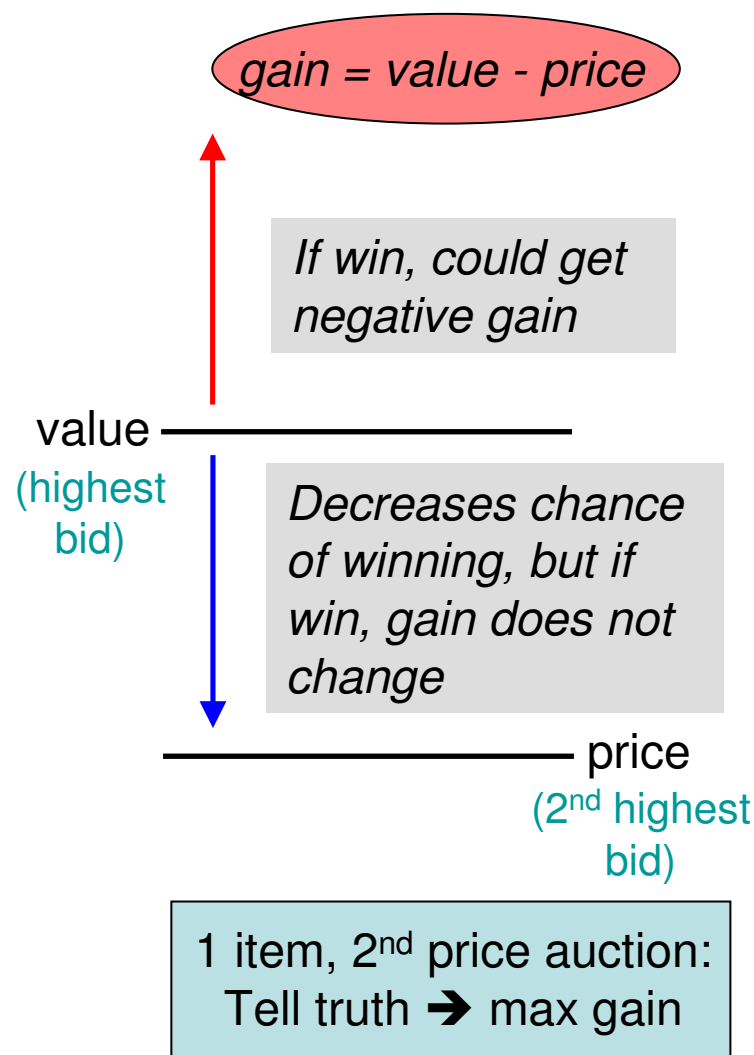
- Winner pays price of next highest bidder – 2nd price!
- Incentive compatible (IC)
 - Truthful bidding is optimal
 - Not IC if 1st-price auction!
 - Easy to see IC in diagram...

- **Sealed-bid combinatorial auction** (*multiple items*)

- More complex definition of 2nd-price to get IC...
- Vickrey-Clarke-Grove (VCG) mechanism

- **Sequential auctions**

- More complex: see <http://www.ftc.gov/be/vogt.pdf>



Summary

- We briefly covered some MDP extensions
 - Time-dependent MDPs
 - Continuous state and action spaces
 - Multiagent (Zero-sum) MDPs
 - First-order MDPS
 - To exploit relational structure
- But also remember...
 - There's more to decision-making than just planning
 - Mechanism design: construct problem so as to optimize social objective function of rational agents