Markov Decision Processes

Manuela M. Veloso

Carnegie Mellon University
Computer Science Department

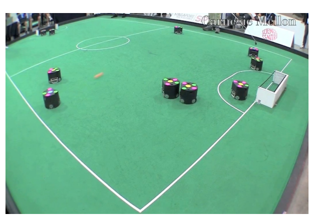
15-780 Graduate AI – Spring 2013

Readings:

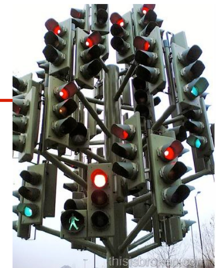
- Russell & Norvig: chapter 17, 17.1-3.

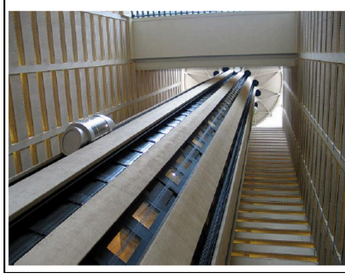
Planning under Uncertainty

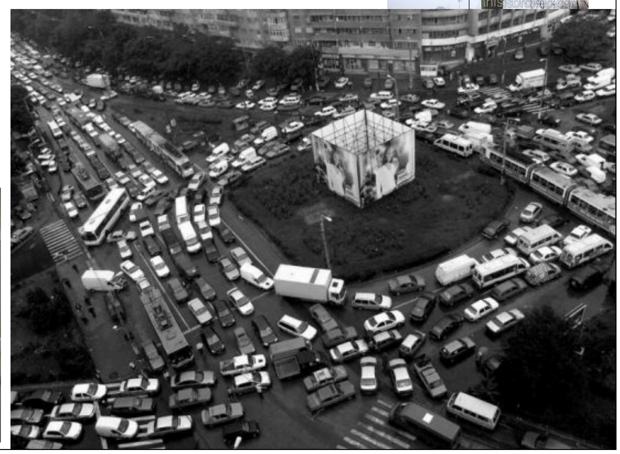
- Motivation: Uncertainty everywhere – discuss; in particular robotics, cyber and physical world



Planning under Uncertainty!



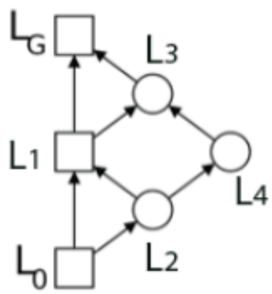




Exploding Blocks World

```
(define (domain exploding-blocks-world-pre)
(:action put-down-block-on-table
         :parameters (?b - block)
         :precondition
         (and (holding ?b)
              (not (destroyed-table)) )
:effect
         (and (not (holding ?b))
         (on-top-of-table ?b)
         ((probabilistic .3   (and (detonated ?b)
                                   (destroyed-table)))) )
```

The Triangle TireWorld



- At every move, flat tire 0.5 probability
- Spare tires at some locations only
- L2, L3, L4 have spare tires
- L1 does not

PDDL Representation

```
(:action move-car
  :parameters (?from - location ?to - location)
  :precondition (and (car-at ?from) (road ?from ?to) (not (flat-tire)))
  :effect (and (car-at ?to) (not (car-at ?from))
               (probabilistic 0.5 (flat-tire))
          )
)

 (:action changetire
   :parameters (?loc - location)
   :precondition (and (spare-in ?loc) (car-at ?loc) (flat-tire))
   :effect (and (not (spare-in ?loc)) (not (flat-tire)))
 )
```

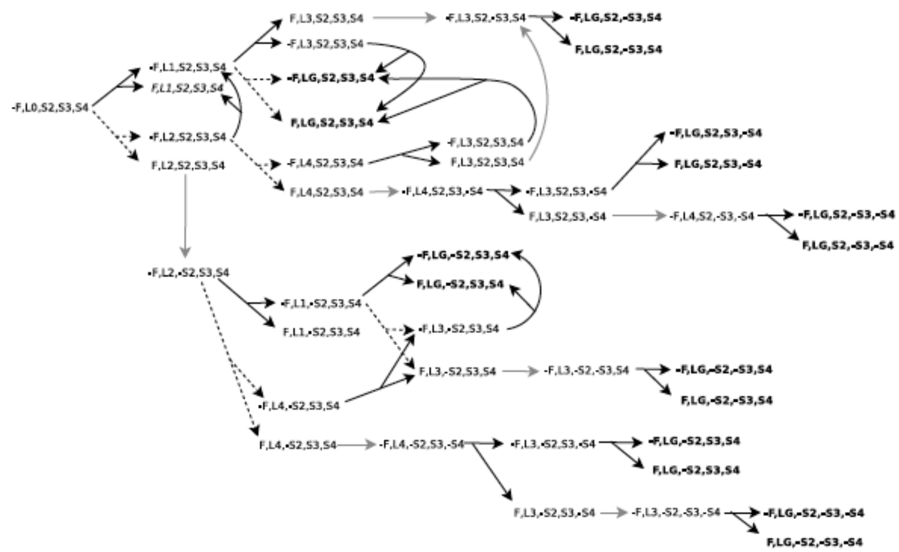


Figure 2: MDP representation of the triangle tireworld of size 1. Black arrows represent the move action, which has 2 resulting states each one with probability 0.5. Gray arrows represent the change-tire action. States in bold are goal states

Markov Decision Processes

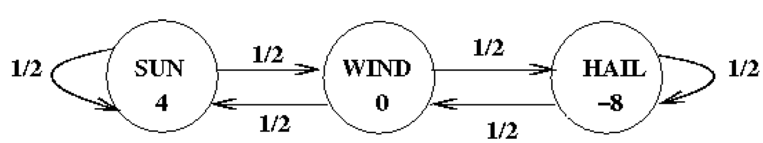
- Finite set of states, $s_1,..., s_n$
- Finite set of actions, $a_1,..., a_m$
- Probabilistic state,action transitions:

  $p_{ij}^k = \text{prob}\,(\text{next} = s_j \mid \text{current} = s_i \text{ and take action } a_k)$
- Markov assumption: State transition function only dependent on current state, not on the "history" of how the state was reached.
- Reward for each state, $r_1,..., r_n$
- Process:
  - Start in state s_i
  - Receive immediate reward r_i
  - Choose action $a_k \in A$
  - Change to state s_j with probability p_{ij}^k .
  - Discount future rewards

Markov Systems with Rewards

- Finite set of n states, s_i
- Probabilistic state matrix, P, p_{ij}
- "Goal achievement" - Reward for each state, r_i
- Discount factor - γ
- Process/observation:
  - Assume start state s_i
  - Receive immediate reward r_i
  - Move, or observe a move, randomly to a new state according to the probability transition matrix
  - Future rewards (of next state) are discounted by γ

Example – Markov System with Reward



- States
- Rewards in states
- Probabilistic transitions between states
- Markov: transitions only depend on current state

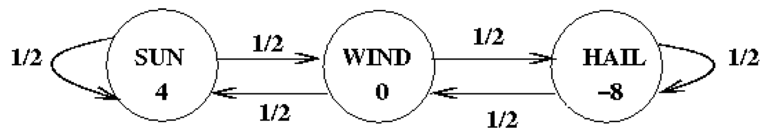
Solving a Markov System with Rewards

- $V^*(s_i)$ - expected discounted sum of future rewards starting in state s_i
- $V^*(s_i) = r_i + \gamma[p_{i1}V^*(s_1) + p_{i2}V^*(s_2) + \ldots p_{in}V^*(s_n)]$

Value Iteration to Solve a Markov System with Rewards

- $V^1(s_i)$ - expected discounted sum of future rewards starting in state s_i *for one step*.
- $V^2(s_i)$ - expected discounted sum of future rewards starting in state s_i *for two steps*.
- …
- $V^k(s_i)$ - expected discounted sum of future rewards starting in state s_i *for* k *steps*.
- As $k \to \infty$ $V^k(s_i) \to V^*(s_i)$
- Stop when difference of $k + 1$ and k values is smaller than some $\in$.

3-State Example



3-State Example: Values $\gamma = 0.5$

Iteration	SUN	WIND	HAIL
0	0	0	0
1	4	0	-8
2	5.0	-1.0	-10.0
3	5.0	-1.25	-10.75
4	4.9375	-1.4375	-11.0
5	4.875	-1.515625	-11.109375
6	4.8398437	-1.5585937	-11.15625
7	4.8203125	-1.5791016	-11.178711
8	4.8103027	-1.5895996	-11.189453
9	4.805176	-1.5947876	-11.194763
10	4.802597	-1.5973969	-11.197388
11	4.8013	-1.5986977	-11.198696
12	4.8006506	-1.599349	-11.199348
13	4.8003254	-1.5996745	-11.199675
14	4.800163	-1.5998373	-11.199837
15	4.8000813	-1.5999185	-11.199919

3-State Example: Values $\gamma = 0.9$

Iteration	SUN	WIND	HAIL
0	0	0	0
1	4	0	-8
2	5.8	-1.8	-11.6
3	5.8	-2.6100001	-14.030001
4	5.4355	-3.7035	-15.488001
5	4.7794	-4.5236254	-16.636175
6	4.1150985	-5.335549	-17.521912
7	3.4507973	-6.0330653	-18.285858
8	2.8379793	-6.6757774	-18.943516
9	2.272991	-7.247492	-19.528683
...	...	...	...
50	-2.8152928	-12.345073	-24.633476
51	-2.8221645	-12.351946	-24.640347
52	-2.8283496	-12.3581295	-24.646532
...	...	...	...
86	-2.882461	-12.412242	-24.700644
87	-2.882616	-12.412397	-24.700798
88	-2.8827558	-12.412536	-24.70094

3-State Example: Values $\gamma = 0.2$

Iteration	SUN	WIND	HAIL
0	0	0	0
1	4	0	-8
2	4.4	-0.4	-8.8
3	4.4	-0.44000003	-8.92
4	4.396	-0.452	-8.936
5	4.3944	-0.454	-8.9388
6	4.39404	-0.45443997	-8.93928
7	4.39396	-0.45452395	-8.939372
8	4.393944	-0.4545412	-8.939389
9	4.3939404	-0.45454454	-8.939393
10	4.3939395	-0.45454526	-8.939394
11	4.3939395	-0.45454547	-8.939394
12	4.3939395	-0.45454547	-8.939394

Solving an MDP

- Find an action to apply to each state.
- A policy is a mapping from states to actions.
- Optimal policy - for every state, there is no other action that gets a higher sum of discounted future rewards.
- For every MDP there exists an optimal policy.
- Solving an MDP is finding an optimal policy.
- A specific policy converts an MDP into a plain Markov system with rewards.

Value Iteration

- $V^*(s_i)$ - expected discounted future rewards, if we start from state $s_{i,}$ and we follow the optimal policy.
- Compute V^* with value iteration:
  - $V^k(s_i)$ = maximum possible future sum of rewards starting from state s_i for k steps.
- Bellman's Equation:

$$V^{n+1}(s_i) = \max_k \{r_i + \gamma \sum_{j=1}^{N} p_{ij}^k V^n(s_j)\}$$

- Dynamic programming

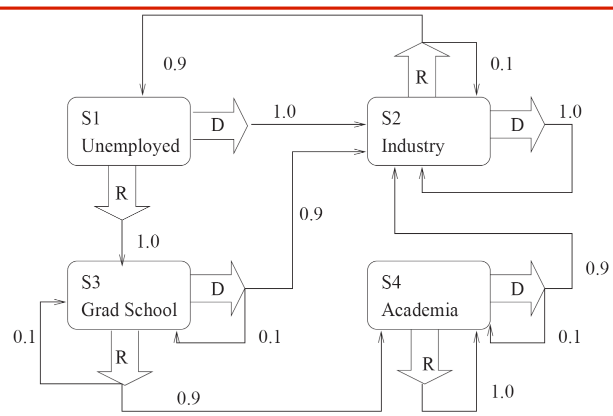
Policy Iteration

- Start with some policy $\pi_0(s_i)$.
- Such policy transforms the MDP into a plain Markov system with rewards.
- Compute the values of the states according to the current policy.
- Update policy:

$$\pi_{k+1}(s_i) = \arg\max_a \{r_i + \gamma \sum_j p_{ij}^a V^{\pi_k}(s_j)\}$$

- Keep computing
- Stop when $\pi_{k+1} = \pi_k$.

Nondeterministic Example



REWARD

S1	S2	S3	S4
0	100	0	10

Nondeterministic Example

$\pi^*(s)$ = D, for any s= S1, S2, S3, and S4, $\gamma = 0.9$.

```
V*(S2) = r(S2,D) + 0.9 (1.0 V*(S2))
V*(S2) = 100 + 0.9 V*(S2)
V*(S2) = 1000.

V*(S1) = r(S1,D) + 0.9 (1.0 V*(S2))
V*(S1) = 0 + 0.9 x 1000
V*(S1) = 900.

V*(S3) = r(S3,D) + 0.9 (0.9 V*(S2) + 0.1 V*(S3))
V*(S3) = 0 + 0.9 (0.9 x 1000 + 0.1 V*(S3))
V*(S3) = 81000/91.

V*(S4) = r(S4,D) + 0.9 (0.9 V*(S2) + 0.1 V*(S4))
V*(S4) = 40 + 0.9 (0.9 x 1000 + 0.1 V*(S4))
V*(S4) = 85000/91.
```

Summary: Markov Models

- Plan is a ***Policy***
  - ***Stationary***: Best action is fixed
  - ***Non-stationary***: Best action depends on time
- States can be *discrete*, ***continuous***, or ***hybrid***

	Passive	**Controlled**
Fully Observable	**Markov Models**	**MDP**
Hidden State	**HMM**	**POMDP**
Time Dependent	**Semi-Markov**	**SMDP**

Tradeoffs

- **MDPs**
  + Tractable to solve
  + Relatively easy to specify
  – Assumes perfect knowledge of state
- **POMDPs**
  + Treats all sources of uncertainty uniformly
  + Allows for taking actions that gain information
  – Difficult to specify all the conditional probabilities
  – *Hugely* intractable to solve optimally
- **SMDPs**
  + General distributions for action durations
  – Few good solution algorithms

Summary

- Planning under uncertainty
- Markov Models with Reward
- Value Iteration
- Markov Decision Process
- Value Iteration
- Policy Iteration
- POMDPs (later)
- Reinforcement Learning (later)