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

```pddl
(: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)))
)
```

Manuela Veloso 5 15-780 Spring 2013
Markov Decision Processes

• Finite set of states, \( s_1, \ldots, s_n \)
• Finite set of actions, \( a_1, \ldots, a_m \)
• Probabilistic state,action transitions:
  \[ p_{ij}^k = \text{prob (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, \( r_1, \ldots, r_n \)
• 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

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 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 - $\gamma$
- 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 $\gamma$

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 $\epsilon$. 
3-State Example

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

<table>
<thead>
<tr>
<th>Iteration</th>
<th>SUN</th>
<th>WIND</th>
<th>HAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>5.0</td>
<td>-1.0</td>
<td>-10.0</td>
</tr>
<tr>
<td>3</td>
<td>5.0</td>
<td>-1.25</td>
<td>-10.75</td>
</tr>
<tr>
<td>4</td>
<td>4.9375</td>
<td>-1.4375</td>
<td>-11.0</td>
</tr>
<tr>
<td>5</td>
<td>4.875</td>
<td>-1.515625</td>
<td>-11.109375</td>
</tr>
<tr>
<td>6</td>
<td>4.8398437</td>
<td>-1.5585937</td>
<td>-11.15625</td>
</tr>
<tr>
<td>7</td>
<td>4.8203125</td>
<td>-1.5791016</td>
<td>-11.17811</td>
</tr>
<tr>
<td>8</td>
<td>4.8103027</td>
<td>-1.5895996</td>
<td>-11.189453</td>
</tr>
<tr>
<td>9</td>
<td>4.805176</td>
<td>-1.5947876</td>
<td>-11.194763</td>
</tr>
<tr>
<td>10</td>
<td>4.802597</td>
<td>-1.5973969</td>
<td>-11.197388</td>
</tr>
<tr>
<td>11</td>
<td>4.8013</td>
<td>-1.5986977</td>
<td>-11.198696</td>
</tr>
<tr>
<td>12</td>
<td>4.8006506</td>
<td>-1.599349</td>
<td>-11.199348</td>
</tr>
<tr>
<td>13</td>
<td>4.8003254</td>
<td>-1.5996745</td>
<td>-11.199675</td>
</tr>
<tr>
<td>14</td>
<td>4.800163</td>
<td>-1.5998373</td>
<td>-11.199837</td>
</tr>
<tr>
<td>15</td>
<td>4.8000813</td>
<td>-1.5999185</td>
<td>-11.199919</td>
</tr>
</tbody>
</table>
### 3-State Example: Values $\gamma = 0.9$

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

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

<table>
<thead>
<tr>
<th>Iteration</th>
<th>SUN</th>
<th>WIND</th>
<th>HAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0</td>
<td>-8</td>
</tr>
<tr>
<td>2</td>
<td>4.4</td>
<td>-0.4</td>
<td>-8.8</td>
</tr>
<tr>
<td>3</td>
<td>4.4</td>
<td>-0.44000003</td>
<td>-8.92</td>
</tr>
<tr>
<td>4</td>
<td>4.396</td>
<td>-0.452</td>
<td>-8.936</td>
</tr>
<tr>
<td>5</td>
<td>4.3944</td>
<td>-0.454</td>
<td>-8.9388</td>
</tr>
<tr>
<td>6</td>
<td>4.39404</td>
<td>-0.45443997</td>
<td>-8.93928</td>
</tr>
<tr>
<td>7</td>
<td>4.39396</td>
<td>-0.45452395</td>
<td>-8.939372</td>
</tr>
<tr>
<td>8</td>
<td>4.393944</td>
<td>-0.4545412</td>
<td>-8.939389</td>
</tr>
<tr>
<td>9</td>
<td>4.3939404</td>
<td>-0.45454454</td>
<td>-8.939393</td>
</tr>
<tr>
<td>10</td>
<td>4.3939395</td>
<td>-0.45454526</td>
<td>-8.939394</td>
</tr>
<tr>
<td>11</td>
<td>4.3939395</td>
<td>-0.45454547</td>
<td>-8.939394</td>
</tr>
<tr>
<td>12</td>
<td>4.3939395</td>
<td>-0.45454547</td>
<td>-8.939394</td>
</tr>
</tbody>
</table>
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:
  \[
  V^{n+1}(s_i) = \max_k \{ r_i + \gamma \sum_{j=1}^{N} p_{ij} V^n(s_j) \}
  \]
- 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:
  $$\pi_{k+1}(s_i) = \arg \max_a \{ r_i + \gamma \sum_j p_{ij} V_{\pi_k}(s_j) \}$$
- Keep computing
- Stop when $\pi_{k+1} = \pi_k$.

Nondeterministic Example
Nondeterministic Example

\( \pi^*(s) = D, \) for any \( s = S_1, S_2, S_3, \) and \( S_4, \gamma = 0.9. \)

\[
V^*(S_2) = r(S_2, D) + 0.9 (1.0 \ V^*(S_2))
\]
\[
V^*(S_2) = 100 + 0.9 \ V^*(S_2)
\]
\[
V^*(S_2) = 1000.
\]
\[
V^*(S_1) = r(S_1, D) + 0.9 (1.0 \ V^*(S_2))
\]
\[
V^*(S_1) = 0 + 0.9 \times 1000
\]
\[
V^*(S_1) = 900.
\]
\[
V^*(S_3) = r(S_3, D) + 0.9 (0.9 \ V^*(S_2) + 0.1 \ V^*(S_3))
\]
\[
V^*(S_3) = 0 + 0.9 (0.9 \times 1000 + 0.1 \ V^*(S_3))
\]
\[
V^*(S_3) = 81000/91.
\]
\[
V^*(S_4) = r(S_4, D) + 0.9 (0.9 \ V^*(S_2) + 0.1 \ V^*(S_4))
\]
\[
V^*(S_4) = 40 + 0.9 (0.9 \times 1000 + 0.1 \ V^*(S_4))
\]
\[
V^*(S_4) = 85000/91.
\]

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**

<table>
<thead>
<tr>
<th></th>
<th>Passive</th>
<th>Controlled</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fully Observable</strong></td>
<td>Markov Models</td>
<td>MDP</td>
</tr>
<tr>
<td><strong>Hidden State</strong></td>
<td>HMM</td>
<td>POMDP</td>
</tr>
<tr>
<td><strong>Time Dependent</strong></td>
<td>Semi-Markov</td>
<td>SMDP</td>
</tr>
</tbody>
</table>
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)