CSEP 573

Markov Decision Processes: Heuristic Search & Real-Time Dynamic Programming

Slides adapted from Andrey Kolobov and Mausam

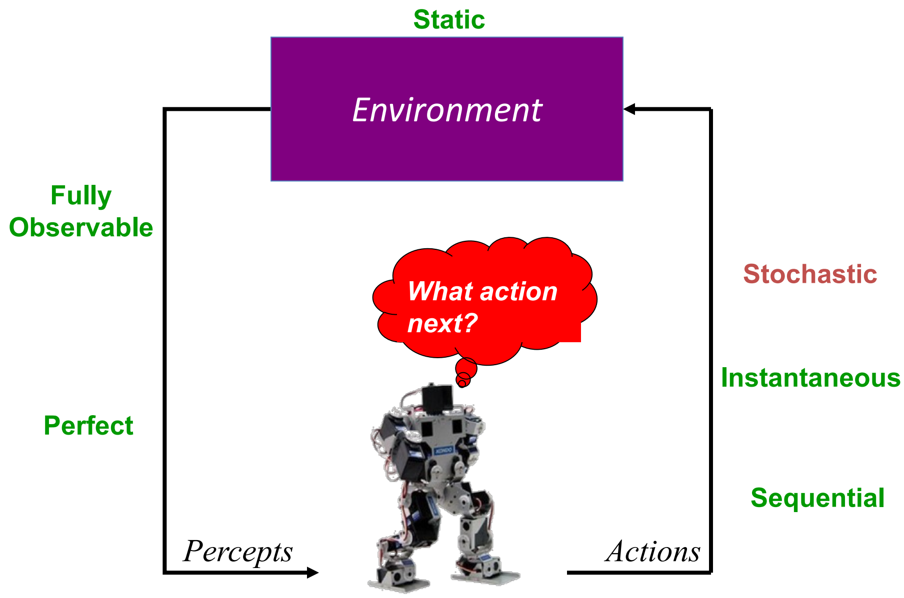
1

Midterm

Min	Mean & Median	Max
31	41	52

Standard Dev = 5

Probabilistic Planning

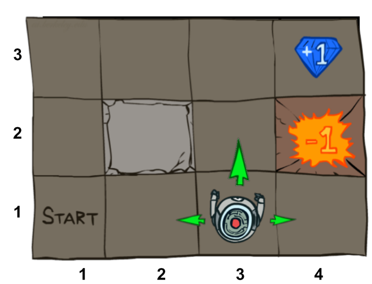


Markov Decision Processes

- An MDP is defined by:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition function $T(s, a, s')$
    - Probability that a from s leads to s', i.e., $P(s' | s, a)$
    - Also called the model or the dynamics
  - A reward function $R(s, a, s')$

…
$R(s_{32}, N, s_{33}) = -0.01$
…
$R(s_{32}, N, s_{42}) = -1.01$
…
$R(s_{33}, E, s_{43}) = 0.99$

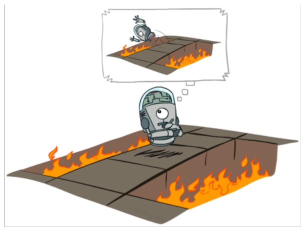
Cost of breathing



R & T are big tables!
For now, we give them to the agent

- We want to output the optimal policy, $\pi^*: S \rightarrow A$
  - One that maximizes expected value of the discounted sequence of rewards.

Offline *vs.* Online



Offline Solution
(Planning)
Think hard; compute policy; then act

Online Learning
(RL)
Act; learning as you go

The Bellman Equations

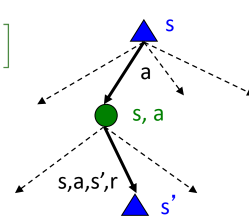
(1920-1984)

Definition of "optimal utility" via expectimax recurrence gives a simple one-step lookahead relationship amongst optimal utility values

$$V^*(s) = \max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$

These … values … use over and over



Value Iteration [Bellman 57]

iteration n

initialize V_0 arbitrarily for each state
$n \leftarrow 0$
repeat
    $n \leftarrow n + 1$
    foreach $s \in \mathcal{S}$ **do**
        compute $V_n(s)$ using Bellman backup at s
        compute $\text{residual}_n(s) = |V_n(s) - V_{n-1}(s)|$
    end
until $\max_{s \in \mathcal{S}} \text{residual}_n(s) < \epsilon$;
return greedy policy: $\pi^{V_n}(s) = \text{argmin}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s')\,[\mathcal{C}(s,a,s') + V_n(s')]$

Aka dynamic programming
Generating optimal policy assuming n time steps
In terms of optimal policy for n-1 steps
Converges to optimal policy for infinite horizon

35

(General) Asynchronous VI

initialize V arbitrarily for each state
while $Res^V > \epsilon$ **do**
    select a state s
    compute $V(s)$ using a Bellman backup at s
    update $Res^V(s)$
end
return greedy policy π^V

$Res^V(s) = |\ V(s) - \max \sum T(s,a,s')[R(s,a,s') + V(s')]\ |$
$Res^V = \max_s Res^V(s)$

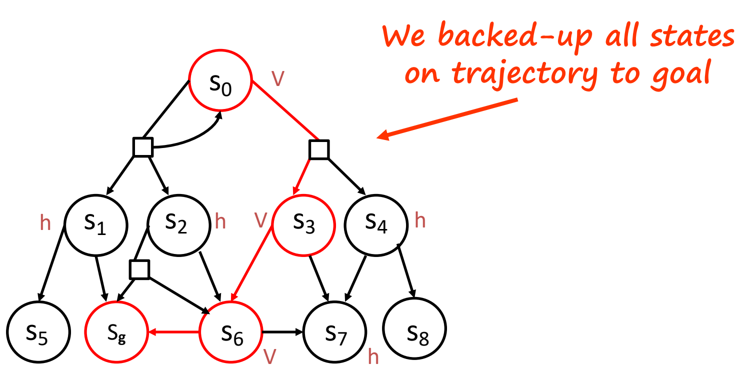
Real Time Dynamic Programming

[Barto et al 95]

- An ***instance*** of asynchronous value iteration
- RTDP: repeat trials forever
  - Converges in the limit #trials → ∞
  - (doesn't test residual)
- Trial
  - simulate greedy policy starting from start state;
  - perform Bellman backup on visited states
  - stop when you hit the goal (or after N actions)
- Original Motivation
  - Agent acting in the real world (**online**)
  - But also useful as a planning algorithm (**offline**)

95

Trial



RTDP

start at start state
repeat
choose greedy action, a
perform a Bellman backup
simulate greedy action
until hit the goal

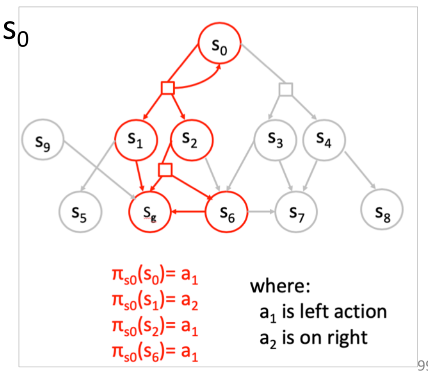
repeat forever

98

Creates a Greedy Partial Policy

- Define *greedy policy*: $\pi^V = \text{argmin}_a \, Q^V(s,a)$

- Define *greedy partial policy rooted at s_0*
  - Partial policy rooted at s_0
  - Greedy policy$_V$
  - denoted by π^{s0}



99

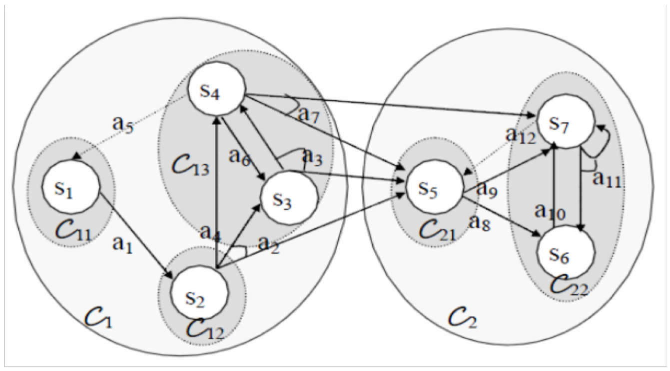
RTDP

- Pros
  - anytime
  - focuses attention on more probable successor states
  - can use admissible heuristic
- Cons
  - no termination condition (see LRTDP)
  - no emphasis on highly uncertain states (see BRTDP)

107

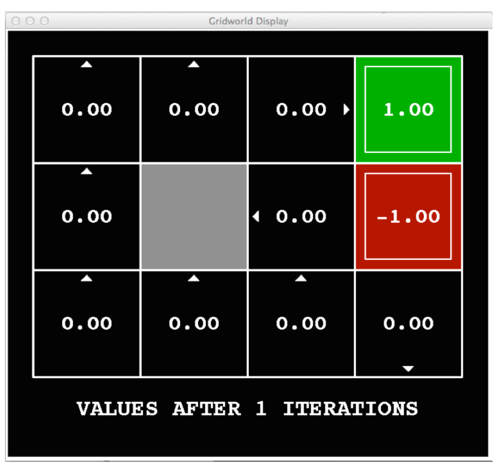
Topological Value Iteration

Another kind of asynchronous VI



129

Backup Order Matters
VI - k=1



Noise = 0.2
Discount = 0.9
Living reward = 0

k=2

Noise = 0.2
Discount = 0.9
Living reward = 0

k=3

Noise = 0.2
Discount = 0.9
Living reward = 0

TVI: When may it perform badly?

- Highly interconnected MDP
- TVI pays overhead of computing connected components
- Reaps no benefit

133

TVI: Convincing Experiments?

- Authors cherry-picked examples with many CCs
- And only 2 examples
- And one is artificial

- What should they have done?

134