

Markov Decision Processes

Reinforcement Learning and Planning under Uncertainty

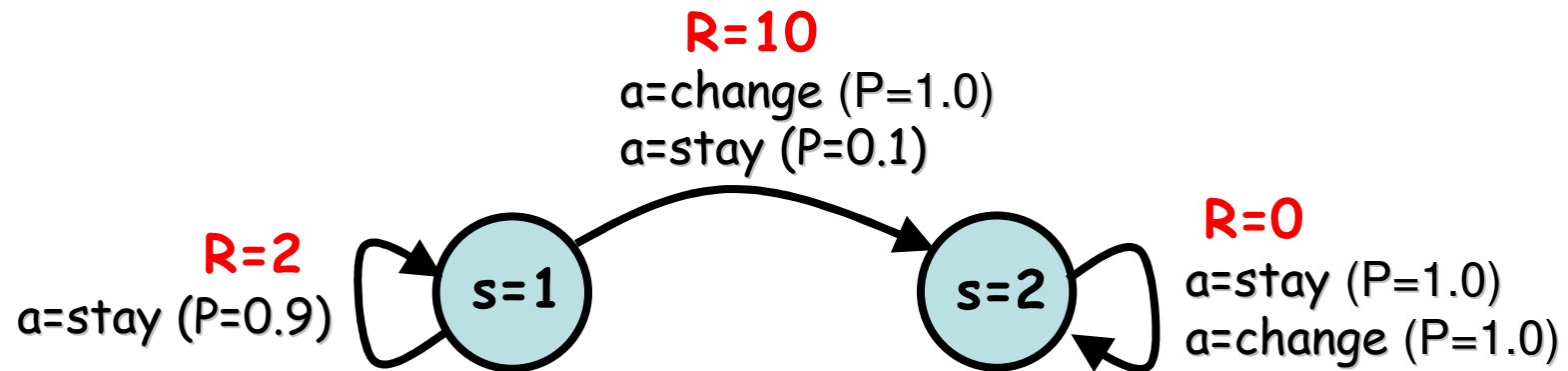
ANU COMP6460/4640, Sem 2, 2008

Scott Sanner

NICTA / RSISE

First.Last@nicta.com.au

MDPs $\langle S, A, T, R, \gamma \rangle$

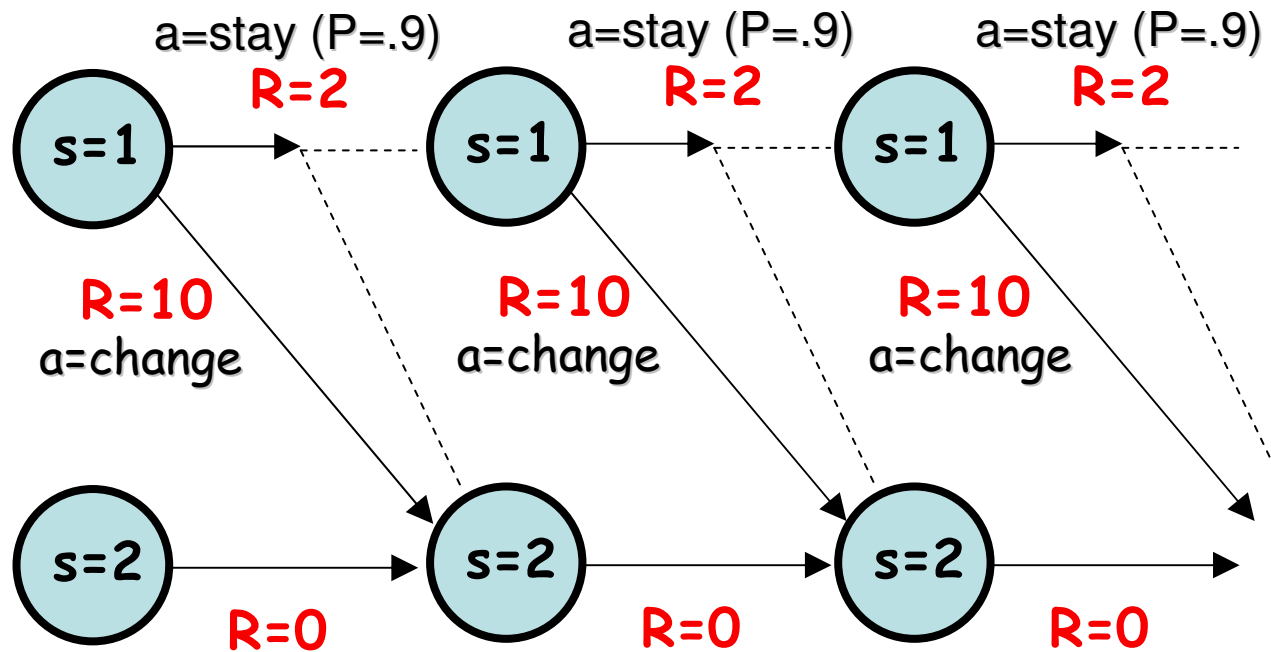


- $S = \{1,2\}; A = \{stay, change\}$
- Reward
 - $R(s=1, a=stay) = 2$
 - ...
- Transitions
 - $T(s=1, a=stay, s'=1) = P(s'=1 | s=1, a=stay) = .9$
 - ...
- Discount γ

How to act
in an MDP?

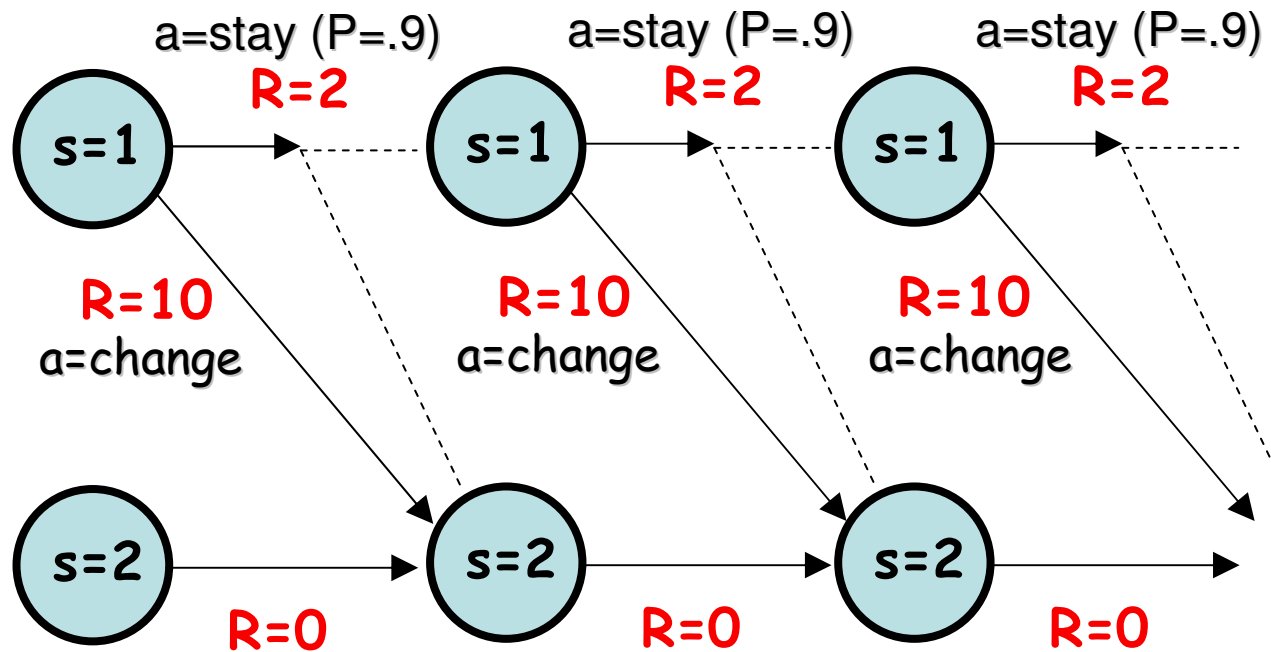
Define policy
 $\pi: S \rightarrow A$

What's the best Policy?



- Immediate vs. long-term gain?

What's the best Policy?



- Must define reward criterion to optimize!
 - Discount factor γ important ($\gamma=.9$ vs. $\gamma=.1$)

MDP Policy, Value, & Solution

- Define value of a policy π :

$$V_{\pi}(s) = E_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t \cdot r_t \mid s = s_0 \right]$$

- Tells how much value you expect to get by following π starting from state s
- MDP Optimal Solution:
 - Find optimal policy π^* that maximizes value
 - Surprisingly: $\exists \pi^* . \forall s, \pi . V_{\pi^*}(s) \geq V_{\pi}(s)$
 - Furthermore:
 - \exists deterministic policy π that performs at least as well as any stochastic policy... so only consider deterministic policies!

Value Function \rightarrow Policy

- Given arbitrary value V (optimal or not)...
 - A *greedy policy* π_V takes action in each state that maximizes expected value w.r.t. V :

$$\pi_V(s) = \arg \max_a \left\{ R(s, a) + \gamma \sum_{s'} T(s, a, s') V(s') \right\}$$

- If can act so as to obtain V after doing action a in state s , π_V guarantees $V(s)$ in expectation

If V not optimal, but a *lower bound* on V^* ,
 π_V guarantees at least that much value!

Value Iteration: from finite to ∞ decisions

- Given optimal *t-1-stage-to-go* value function
- How to act optimally with *t* decisions?
 - Take action *a* then act so as to achieve V^{t-1} thereafter

$$Q^t(s, a) := R(s, a) + \gamma \cdot \sum_{s' \in S} T(s, a, s') \cdot V^{t-1}(s')$$

- What is expected value of best action *a* at decision stage *t*?

$$V^t(s) := \max_{a \in A} \{Q^t(s, a)\}$$

- At ∞ horizon, get same value ($=V^*$)

$$\lim_{t \rightarrow \infty} \max_s |V^t(s) - V^{t-1}(s)| = 0$$

- π^* says act same at each decision stage for ∞ horizon!

Make sure you can derive these equations from first principles!

Bellman Fixed Point

- Define *Bellman backup* operator B :

$$\underbrace{(B V)}_{V^{t-1}}(s) = \max_a \left\{ R(s, a) + \gamma \sum_{s'} T(s, a, s') \underbrace{V(s')}_{V^t} \right\}$$

- \exists an optimal value function V^* and an optimal deterministic greedy policy $\pi^* = \pi_{V^*}$ satisfying:

$$\forall s. V^*(s) = (B V^*)(s)$$

Bellman Error and Properties

- Define *Bellman error* BE :

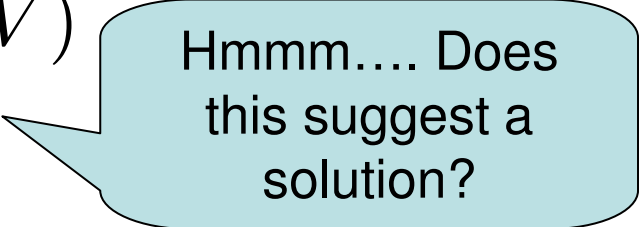
$$(BE V) = \max_s |(BV)(s) - V(s)|$$

- Clearly:

$$(BE V^*) = 0$$

- Can prove B is a contraction operator for BE :

$$(BE (BV)) \leq \gamma (BE V)$$



Hmmm.... Does this suggest a solution?

Value Iteration: in search of fixed-point

- Start with arbitrary value function V^0
- Iteratively apply Bellman backup

$$V^t(s) = (B V^{t-1})(s)$$

- Bellman error decreases on each iteration
 - Terminate when

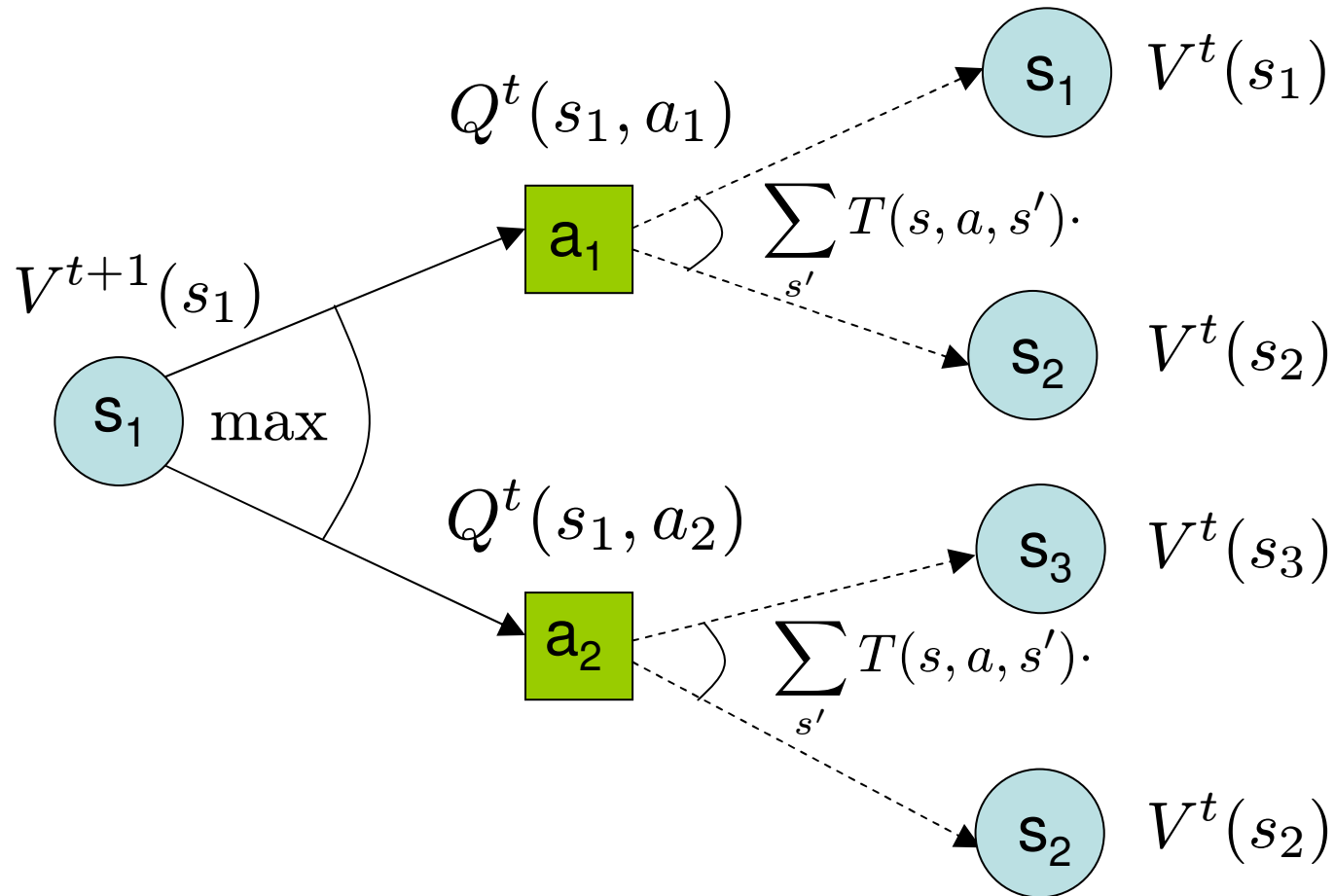
$$\max_s |V^t(s) - V^{t-1}(s)| < \frac{\epsilon(1 - \gamma)}{2\gamma}$$

- Guarantees ϵ -optimal value function
 - i.e., V^t within ϵ of V^* for all states

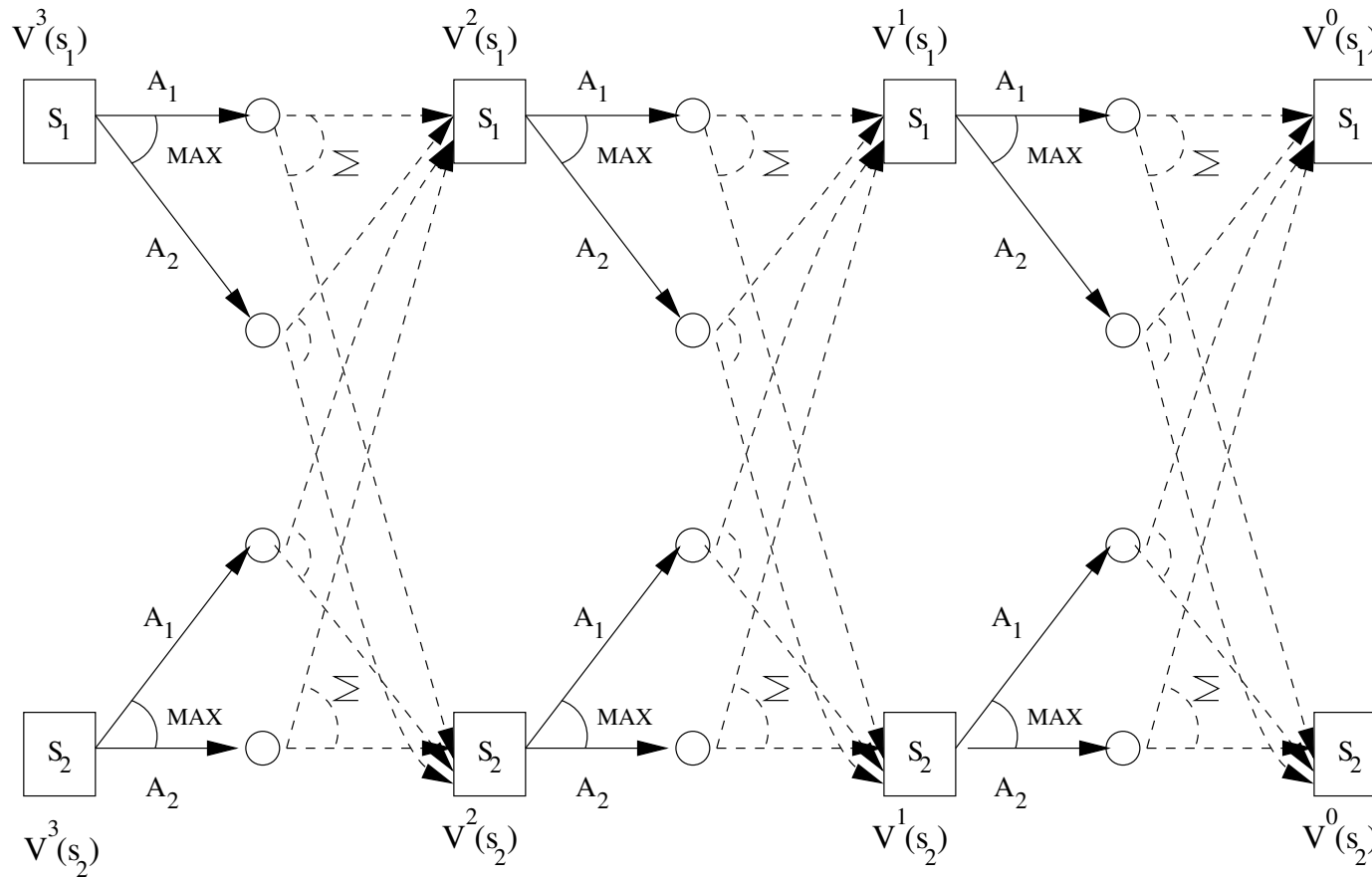
Precompute maximum number of steps for ϵ ?

Single Dynamic Programming Step

- Graphical view:



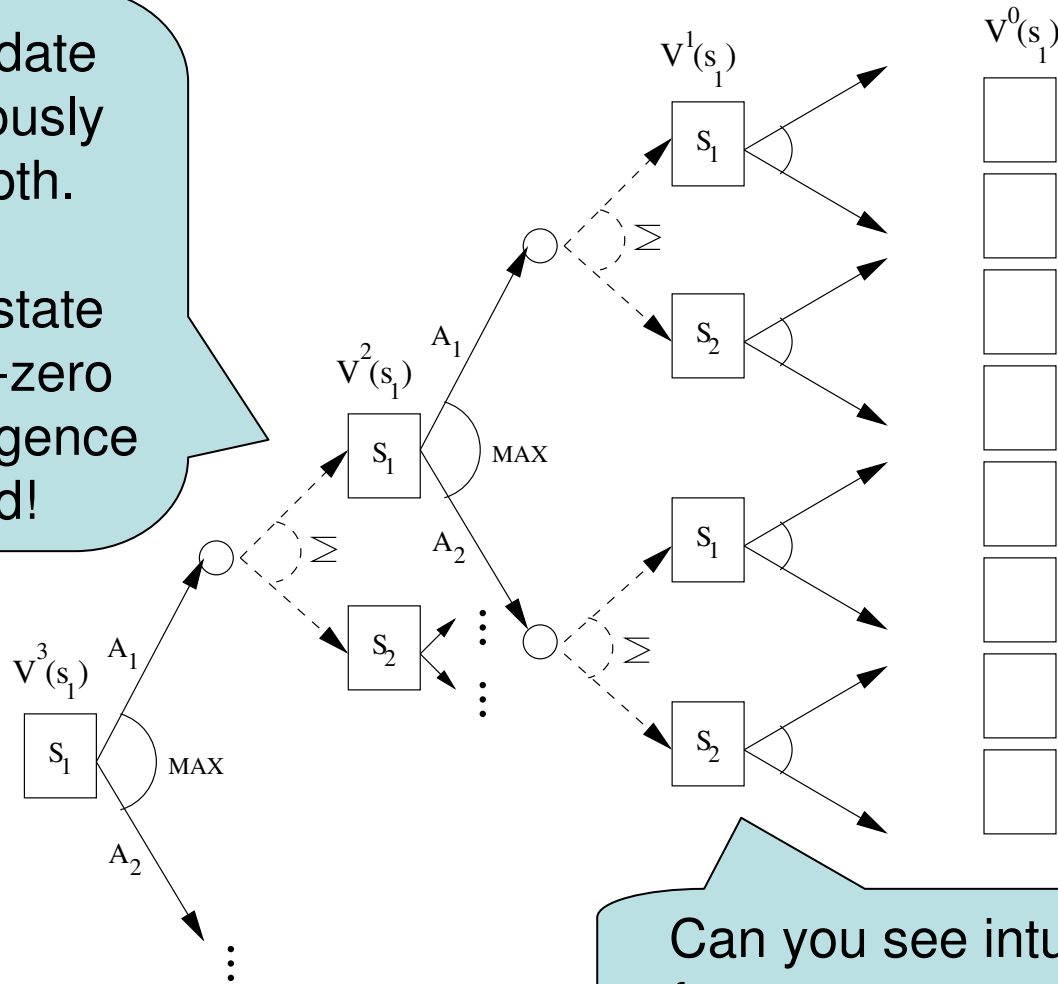
Synchronous DP Updates (Value Iteration)



Asynchronous DP Updates (Asynchronous Value Iteration)

Don't need to update values synchronously with uniform depth.

As long as each state updated with non-zero probability, convergence still guaranteed!



Can you see intuition for error contraction?

Real-time DP (RTDP)

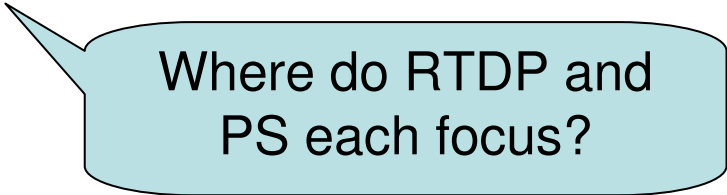
- Async. DP guaranteed to converge over *relevant states*
 - *relevant states*: states reachable from initial states under π^*
 - may converge without visiting all states!

Algorithm 1: RTDP (for stochastic shortest path problems)

```
begin
  // Initialize  $\hat{V}_h$  with admissible value function
   $\hat{V}_h := V_h$ ;
  while not converged and not out of time do
    Draw  $s$  from initial state distribution at random;
    while  $s \notin \{\text{terminal states}\}$  do
       $\hat{V}_h(s) = \text{BELLMANUPDATE}(\hat{V}_h, s)$ ;
       $a = \text{GREEDYACTION}(\hat{V}_h, s)$ ;
       $s := s' \sim T(s, a, s')$ ;
    return  $\hat{V}_h$ ;
end
```

Prioritized Sweeping (PS)

- Simple asynchronous DP idea
 - Focus backups on high error states
 - Can use in conjunction with other focused methods, e.g., RTDP
- Every time state visited:
 - Record Bellman error of state
 - Push state onto queue with priority = Bellman error
- In between simulations / experience, repeat:
 - Withdraw maximal priority state from queue
 - Perform Bellman backup on state
 - Record Bellman error of predecessor states
 - Push predecessor states onto queue with priority = Bellman error



Where do RTDP and PS each focus?

Which approach is better?

- Synchronous DP Updates
 - Good when you need a policy for every state
 - OR transitions are dense
- Asynchronous DP Updates
 - Know best states to update
 - e.g., reachable states, e.g. RTDP
 - e.g., high error states, e.g. PS
 - Know how to order updates
 - e.g., from goal back to initial state if DAG

Policy Evaluation

- Given π , how to derive V_π ?
- *Matrix inversion*
 - Set up linear equality (no max!) for each state

$$\forall s. V_\pi(s) = \left\{ R(s, \pi(s)) + \gamma \sum_{s'} T(s, \pi(s), s') V_\pi(s') \right\}$$

- Can solve linear system in vector form as follows

$$V_\pi = R_\pi (I - \gamma T_\pi)^{-1}$$

Guaranteed invertible.

- *Successive approximation*
 - Essentially value iteration with fixed policy
 - Initialize V_π^0 arbitrarily

$$V_\pi^t(s) := \left\{ R(s, \pi(s)) + \gamma \sum_{s'} T(s, \pi(s), s') V_\pi^{t-1}(s') \right\}$$

- Guaranteed to converge to V_π

Policy Iteration

1. *Initialization:* Pick an arbitrary initial decision policy $\pi_0 \in \Pi$ and set $i = 0$.
2. *Policy Evaluation:* Solve for V_{π_i} (previous slide).
3. *Policy Improvement:* Find a new policy π_{i+1} that is a greedy policy w.r.t. V_{π_i}

(i.e., $\pi_{i+1} \in \arg \max_{\pi \in \Pi} \{R_{\pi} + \gamma T_{\pi} V_{\pi_i}\}$ with ties resolved via a total precedence order over actions).
4. *Termination Check:* If $\pi_{i+1} \neq \pi_i$ then increment i and go to step 2 else return π_{i+1} .

Modified Policy Iteration

- *Value iteration*
 - Each iteration seen as doing 1-step of policy evaluation for current greedy policy
 - Bootstrap with value estimate of previous policy
- *Policy iteration*
 - Each iteration is full evaluation of V_π for current policy π
 - Then do greedy policy update
- *Modified policy iteration*
 - Like policy iteration, but V_{π_i} need only be closer to V^* than $V_{\pi_{i-1}}$
 - Fixed number of steps of successive approximation for V_{π_i} suffices when bootstrapped with $V_{\pi_{i-1}}$
 - Typically faster than VI & PI in practice

Linear Programming

- One side of fixed point equality

$$\begin{aligned} V^*(s) &\geq \max_{a \in A} \left(R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V^*(s') \right); \forall s \in S \\ &\geq R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V^*(s'); \forall a \in A, s \in S \end{aligned}$$

- To satisfy other side, need minimal V
 - Can formulate solution as linear program

Variables: V^*

Minimize: $\|V^*\|_1$

Subject to: $0 \geq R_a + \gamma T_a V^* - V^*, \forall a \in A$

Conclusion

- Basic introduction to Markov decision processes
 - Derivation of Bellman equations
 - Solution via various algorithms
- Should be familiar with following
 - Value Iteration
 - Synchronous DP
 - Asynchronous DP (RTDP, PS)
 - (Modified) Policy Iteration
 - Policy evaluation
 - Linear Programming
- Required reading
 - Sutton & Barto, Chapter 4
 - Sanner, Thesis Chapter 2