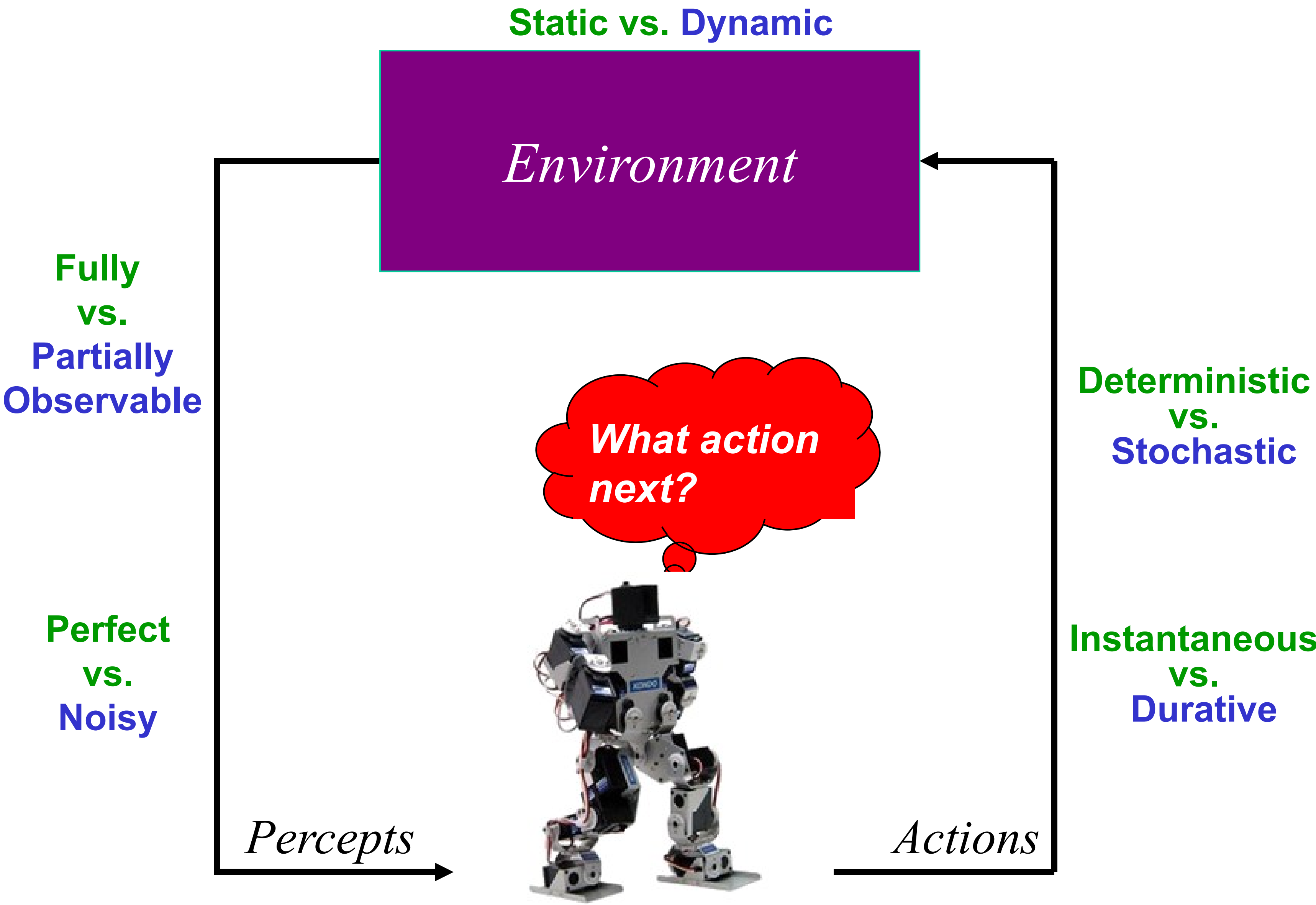
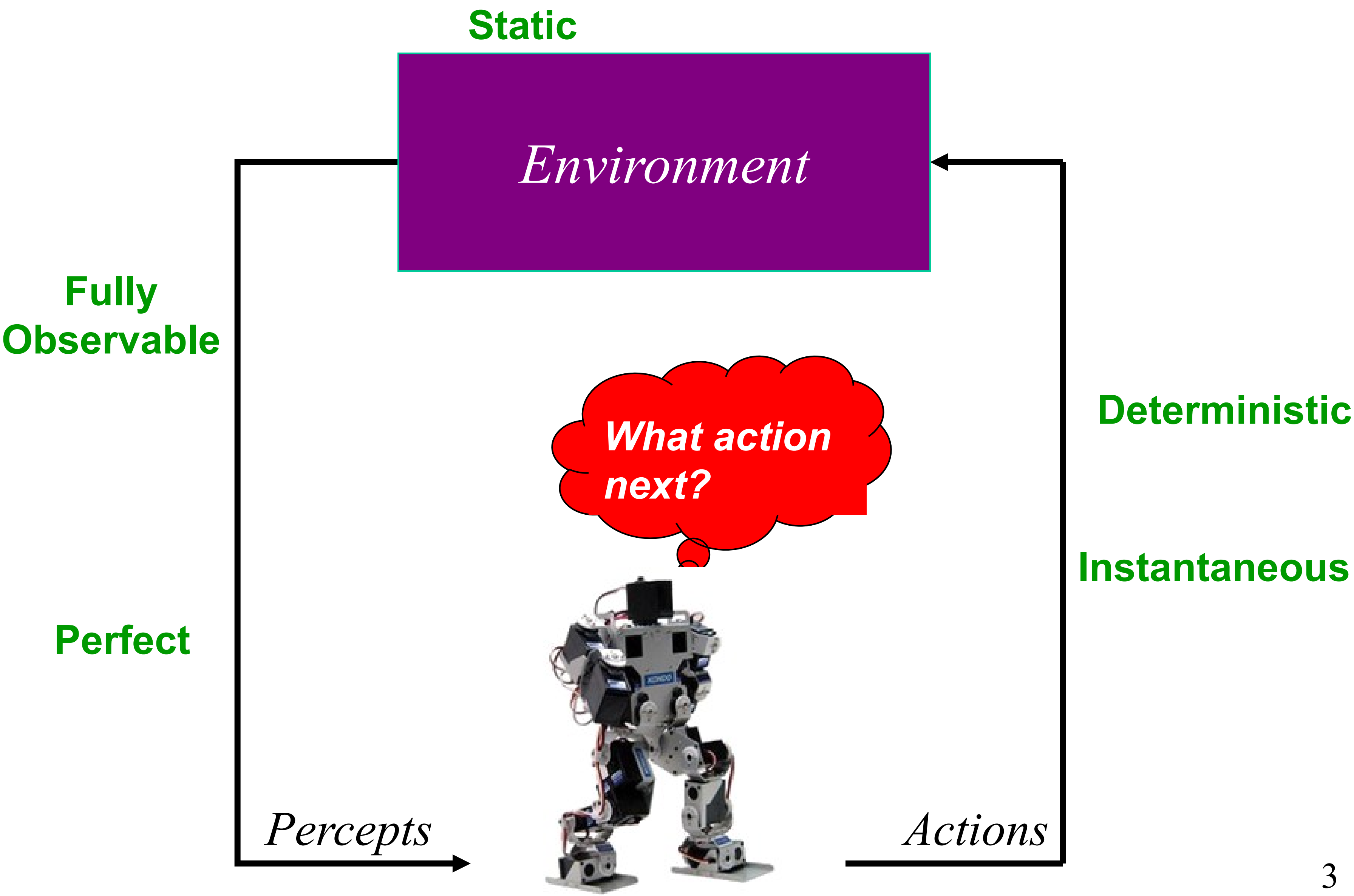
Markov Decision Processes
Chapter 17

Mausam

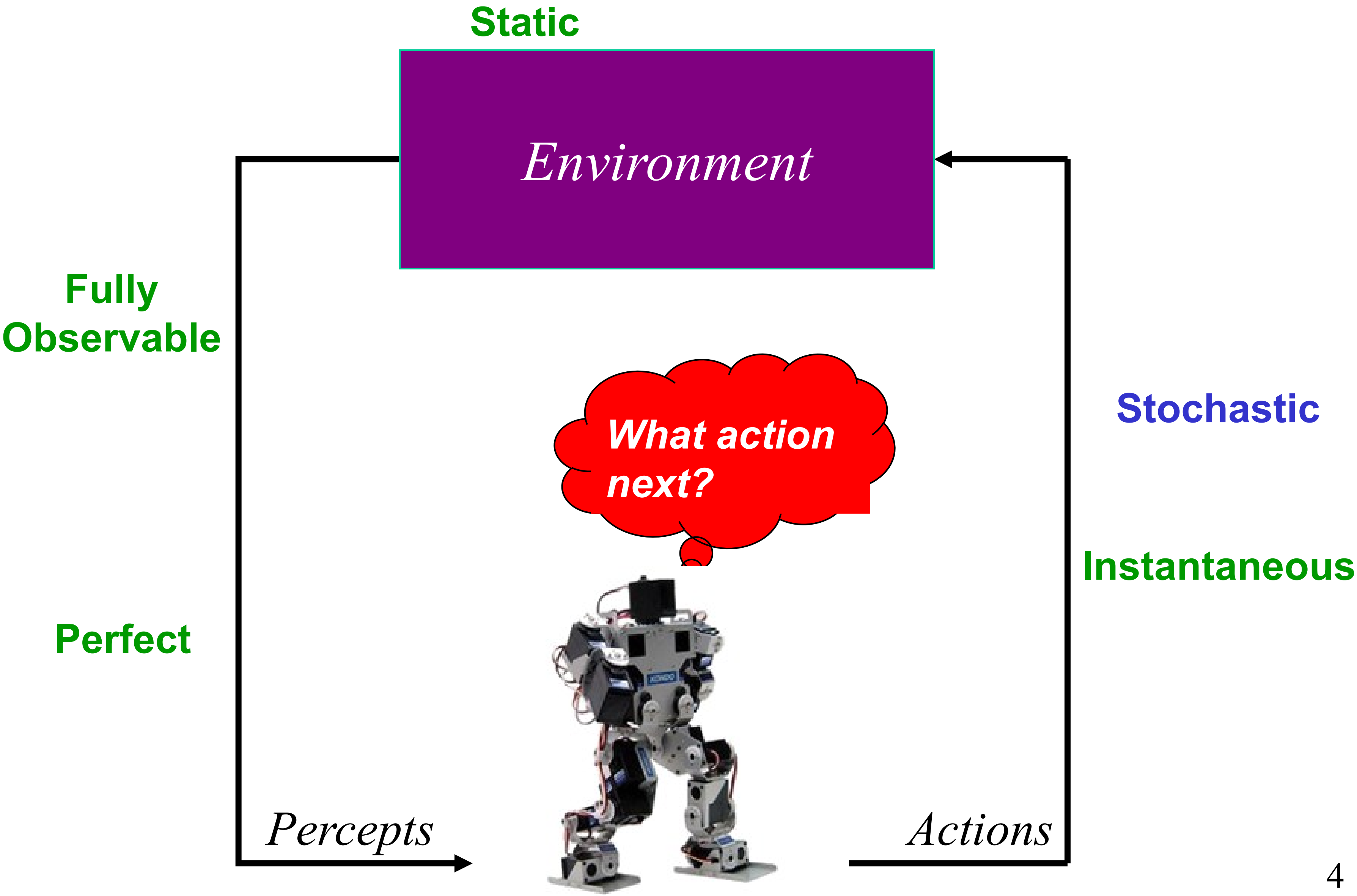
Planning Agent



Classical Planning



Stochastic Planning: MDPs



MDP vs. Decision Theory

- Decision theory - episodic
- MDP -- sequential

Markov Decision Process (MDP)

- $\mathcal{S}$: A set of states
- $\mathcal{A}$: A set of actions
- $\mathcal{T}$(s,a,s'): transition model
- $\mathcal{C}$(s,a,s'): cost model
- $\mathcal{G}$: set of goals
- s_0: start state
- γ: discount factor
- $\mathcal{R}$(s,a,s'): reward model

factored

Factored MDP

absorbing/
non-absorbing

Objective of an MDP

- Find a policy $\pi: \mathcal{S} \rightarrow \mathcal{A}$

- which optimizes
  - minimizes $\begin{pmatrix} \text{discounted} \\ \text{or} \\ \text{undiscount.} \end{pmatrix}$ expected cost to reach a goal
  - maximizes expected reward
  - maximizes expected (reward-cost)

- given a ____ horizon
  - finite
  - infinite
  - indefinite

- assuming full observability

Role of Discount Factor (γ)

- Keep the total reward/total cost finite
  - useful for infinite horizon problems

- Intuition (economics):
  - Money today is worth more than money tomorrow.

- Total reward: $r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots$
- Total cost: $c_1 + \gamma c_2 + \gamma^2 c_3 + \ldots$

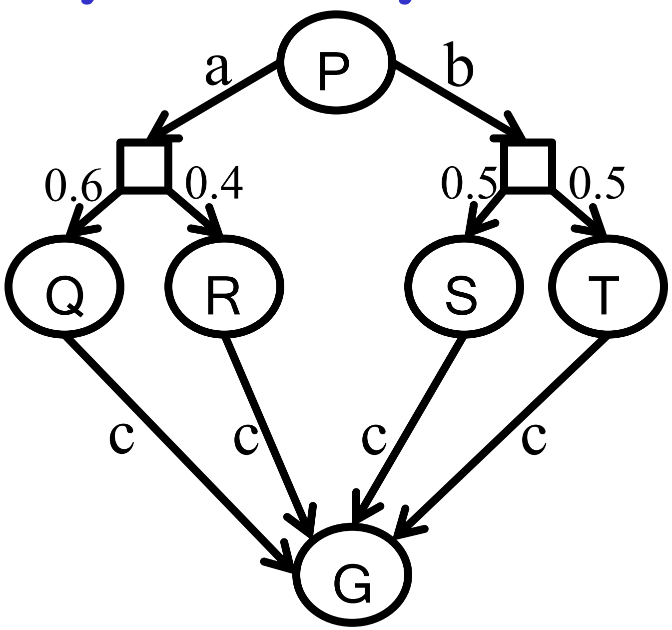
Examples of MDPs

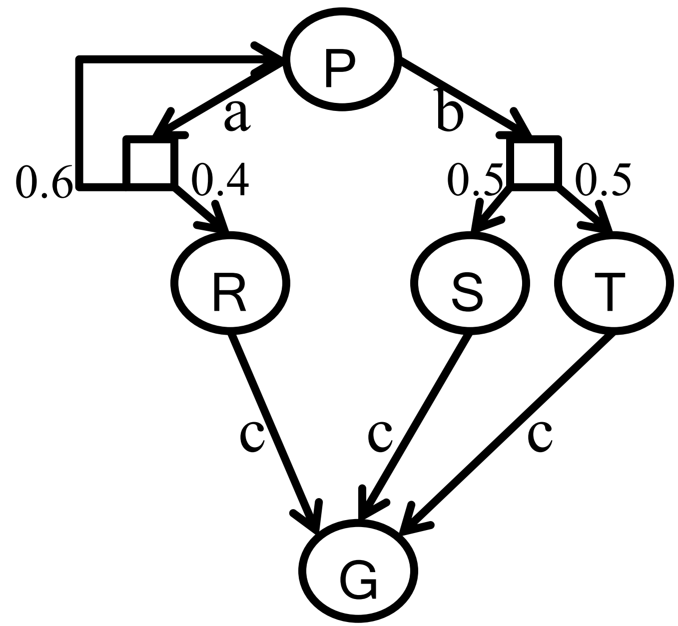
- Goal-directed, Indefinite Horizon, Cost Minimization MDP
  - $<\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, \mathcal{G}, s_0>$
  - Most often studied in planning, graph theory communities
- Infinite Horizon, Discounted Reward Maximization MDP
  - $<\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma>$
  - Most often studied in machine learning, economics, operations research communities

most popular

- Oversubscription Planning: Non absorbing goals, Reward Max. MDP
  - $<\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{G}, \mathcal{R}, s_0>$
  - Relatively recent model

Acyclic vs. Cyclic MDPs





C(a) = 5, C(b) = 10, C(c) =1

Expectimin works

- V(Q/R/S/T) = 1
- V(P) = 6 – action a

Expectimin doesn't work

- infinite loop
- V(R/S/T) = 1
- Q(P,b) = 11
- Q(P,a) = ????
- suppose I decide to take a in P
- Q(P,a) = 5+ 0.4*1 + 0.6Q(P,a)
- ➔ = 13.5

Brute force Algorithm

- Go over all policies π
  - How many? $|A|^{|S|}$ ← finite

- Evaluate each policy ← how to evaluate?
  - $V^{\pi}(s) \leftarrow$ expected cost of reaching goal from s

- Choose the best
  - We know that best exists (SSP optimality principle)
  - $V^{\pi*}(s) \leq V^{\pi}(s)$

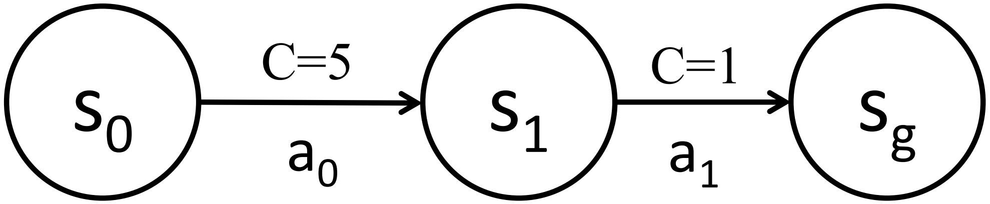
Policy Evaluation

- Given a policy π: compute V^π
  - V^π : cost of reaching goal while following π

Deterministic MDPs

- Policy Graph for π

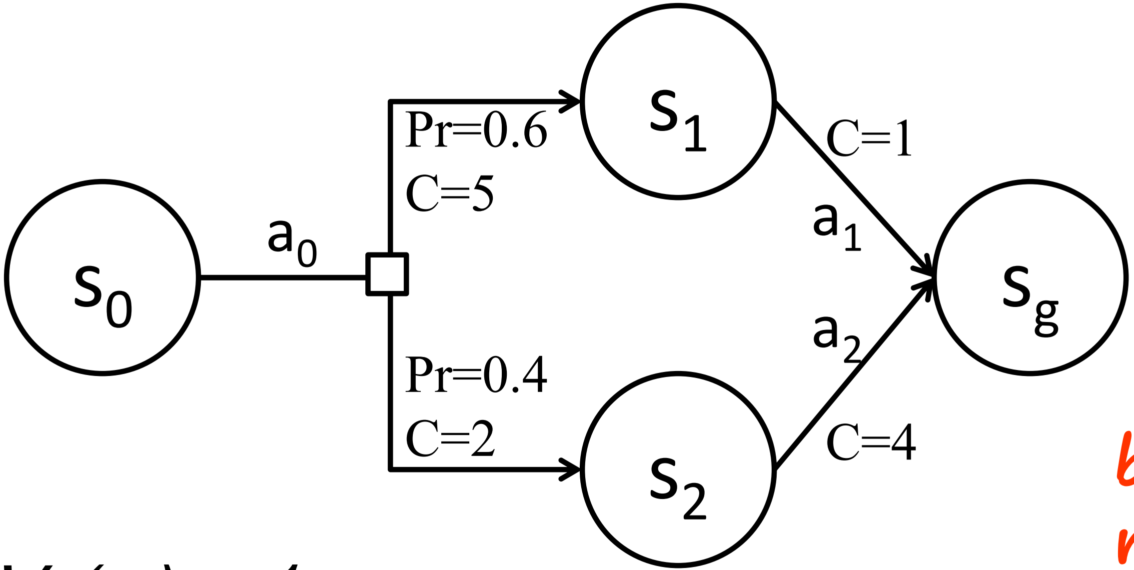
  $\pi(s_0) = a_0; \pi(s_1) = a_1$



- $V^\pi(s_1) = 1$
- $V^\pi(s_0) = 6$

Acyclic MDPs

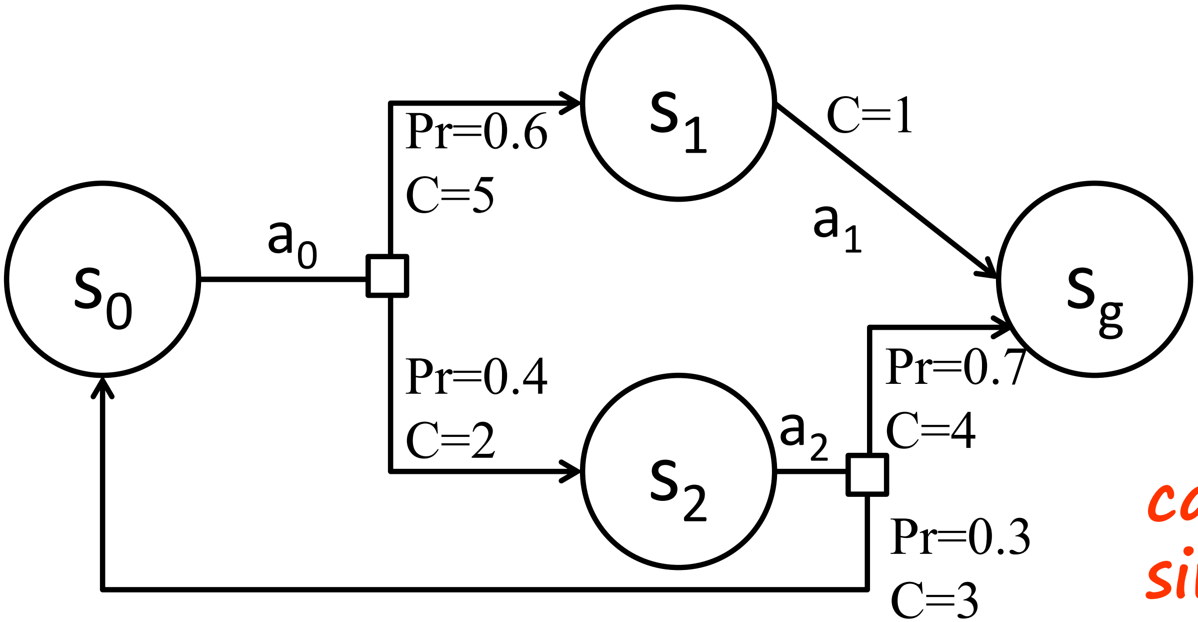
- Policy Graph for π



- $V^{\pi}(s_1) = 1$
- $V^{\pi}(s_2) = 4$
- $V^{\pi}(s_0) = 0.6(5+1) + 0.4(2+4) = 6$

backward pass in reverse topological order

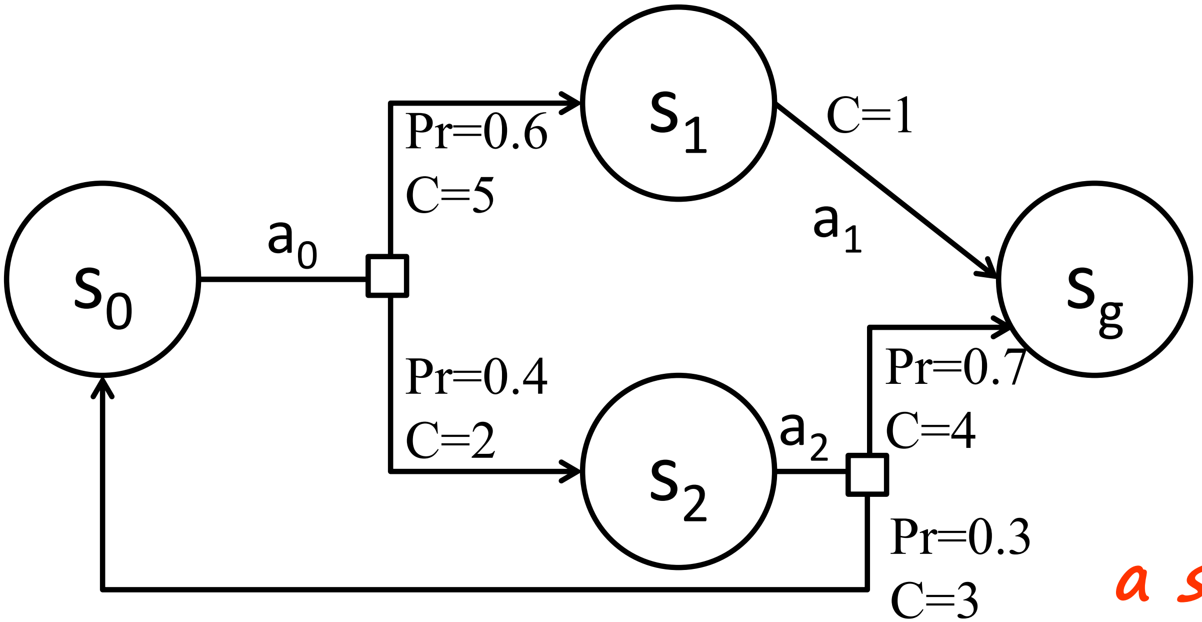
General MDPs can be cyclic!



cannot do a simple single pass

- $V^{\pi}(s_1) = 1$
- $V^{\pi}(s_2) =$?? (depends on $V^{\pi}(s_0)$)
- $V^{\pi}(s_0) =$?? (depends on $V^{\pi}(s_2)$)

General SSPs can be cyclic!



a simple system of linear equations

- $V^{\pi}(g) = 0$
- $V^{\pi}(s_1) = 1 + V^{\pi}(s_g) = 1$
- $V^{\pi}(s_2) = 0.7(4 + V^{\pi}(s_g)) + 0.3(3 + V^{\pi}(s_0))$
- $V^{\pi}(s_0) = 0.6(5 + V^{\pi}(s_1)) + 0.4(2 + V^{\pi}(s_2))$

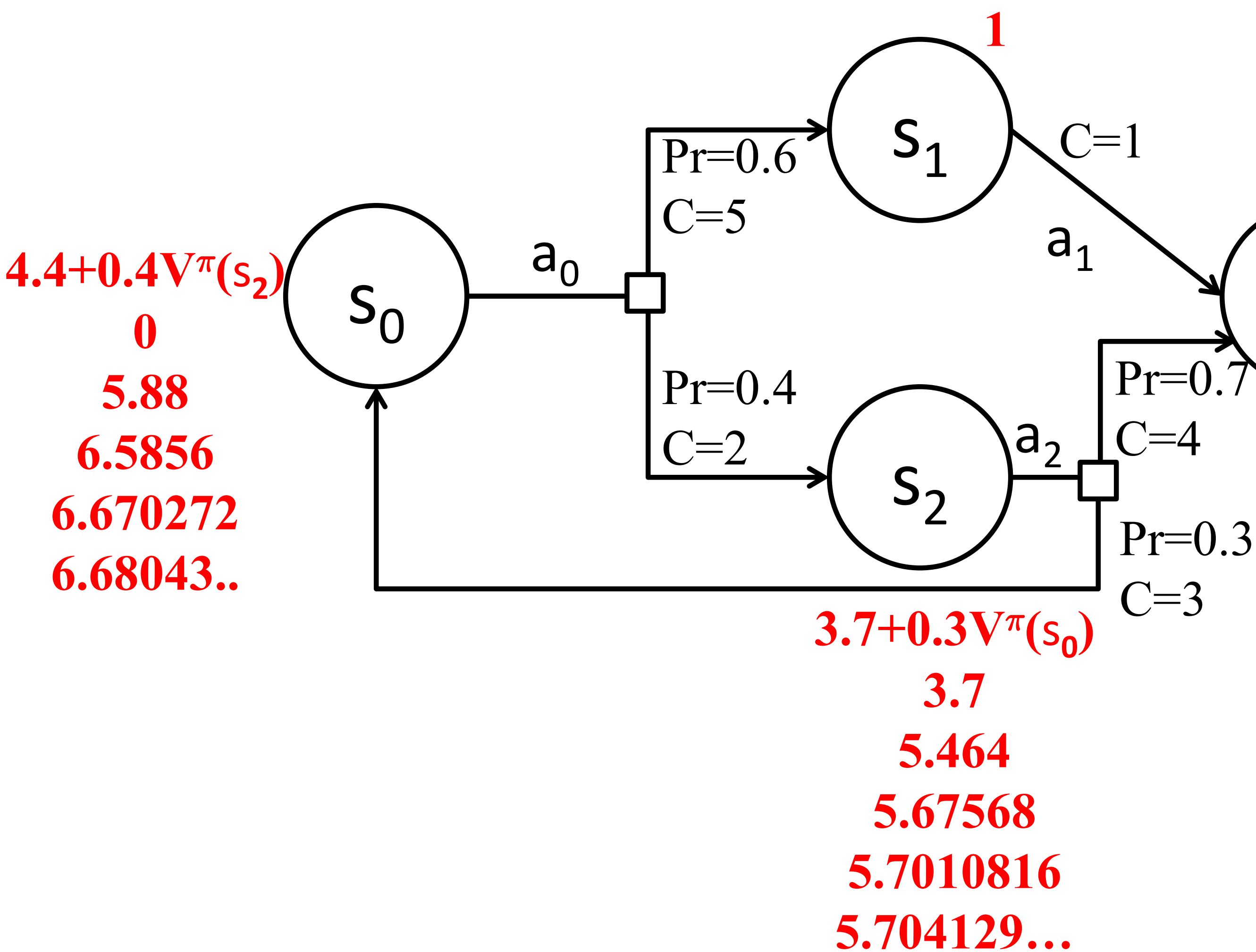
Policy Evaluation (Approach 1)

- Solving the System of Linear Equations

$$V^{\pi}(s) = 0 \quad \text{if } s \in \mathcal{G}$$
$$=$$

- $|S|$ variables.
- $O(|S|^3)$ running time

Iterative Policy Evaluation



Policy Evaluation (Approach 2)

$$V^{\pi}(s) = \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s') \left[\mathcal{C}(s, \pi(s), s') + V^{\pi}(s') \right]$$

iterative refinement

$$V^{\pi}_{n}(s) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s') \left[\mathcal{C}(s, \pi(s), s') + V^{\pi}_{n-1}(s') \right]$$

Iterative Policy Evaluation

```
//Assumption: π is proper
initialize V_0^π arbitrarily for each state
```

iteration n

ε-consistency

termination condition

Convergence & Optimality

For a proper policy π

Iterative policy evaluation

converges to the true value of the policy, i.e.

$$\lim_{n \to \infty} V_n^\pi = V^\pi$$

irrespective of the initialization V_0

Policy Evaluation → Value Iteration (Bellman Equations for MDP$_1$)

- $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, \mathcal{G}, s_0 \rangle$
- Define V*(s) {optimal cost} as the minimum expected cost to reach a goal from this state.
- V* should satisfy the following equation:

$$\begin{aligned} V^*(s) &= 0 \qquad \text{if } s \in \mathcal{G} \\ &= \end{aligned}$$

Q*(s,a)

V*(s) = $\min_a$ **Q*(s,a)**

Bellman Equations for MDP_2

- $<\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, s_0, \gamma>$
- Define **V*(s)** {optimal **value**} as the **maximum** expected **discounted reward** from this state.
- V* should satisfy the following equation:

$$V^*(s) = \max_{a \in Ap(s)} \sum_{s' \in \mathcal{S}} \mathcal{P}r(s'|s,a) \left[\mathcal{R}(s,a,s') + \gamma V^*(s') \right]$$

Fixed Point Computation in VI

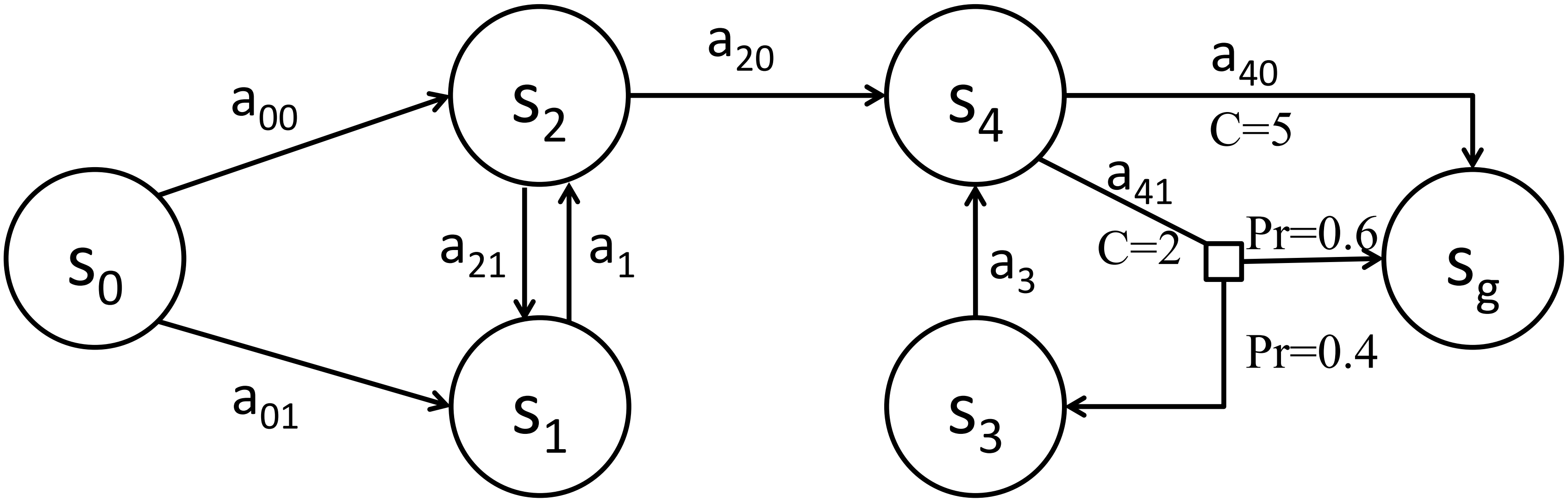
$$V^*(s) = \min_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s')\left[\mathcal{C}(s,a,s') + V^*(s')\right]$$

iterative refinement

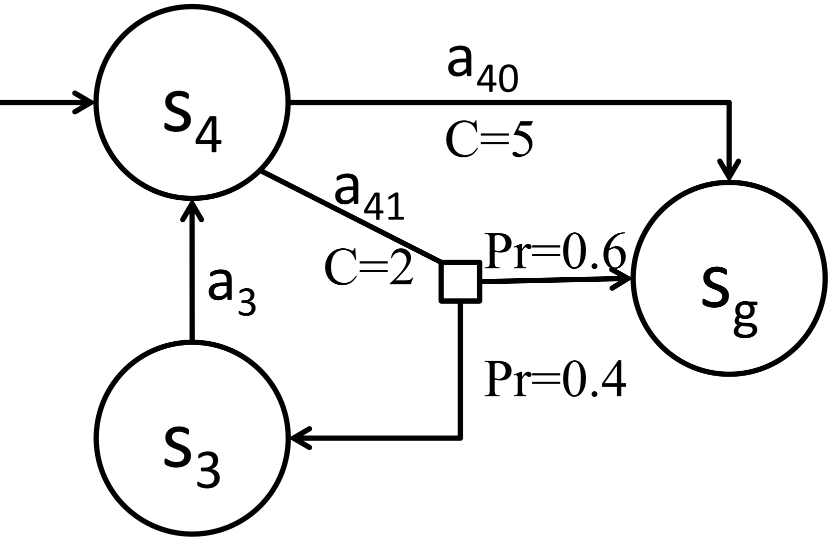
$$V_n(s) \leftarrow \min_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s')\left[\mathcal{C}(s,a,s') + V_{n-1}(s')\right]$$

non-linear

Example

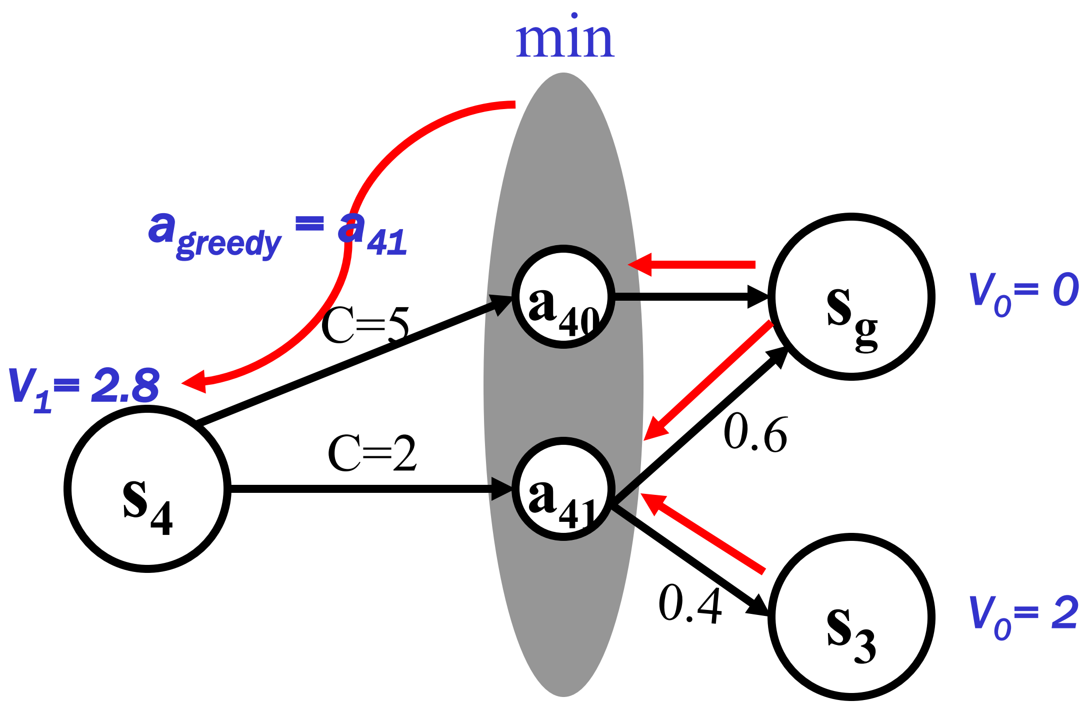


Bellman Backup



$Q_1(s_4,a_{40}) = 5 + 0$

$Q_1(s_4,a_{41}) = 2 + 0.6\times 0$
$+ 0.4\times 2$
$= 2.8$



Value Iteration [Bellman 57]

No restriction on initial value function

iteration n

```
initialize V_0 arbitrarily for each state
n ← 0
repeat
    n ← n + 1
    foreach s ∈ S do
        compute V_n(s) using Bellman backup at s
        compute residual_n(s) = |V_n(s) − V_{n−1}(s)|
    end
until max_{s∈S} residual_n(s) < ε;
return greedy policy: π^{V_n}(s) = argmin_{a∈A} Σ_{s'∈S} T(s,a,s') [C(s,a,s') + V_n(s')]
```

initialize V_0 arbitrarily for each state

$n \leftarrow 0$

repeat

$\quad n \leftarrow n + 1$

$\quad$ **foreach** $s \in \mathcal{S}$ **do**

$\quad\quad$ compute $V_n(s)$ using Bellman backup at s

$\quad\quad$ compute $\text{residual}_n(s) = |V_n(s) - V_{n-1}(s)|$

$\quad$ **end**

until $\max_{s \in \mathcal{S}} \text{residual}_n(s) < \epsilon$;

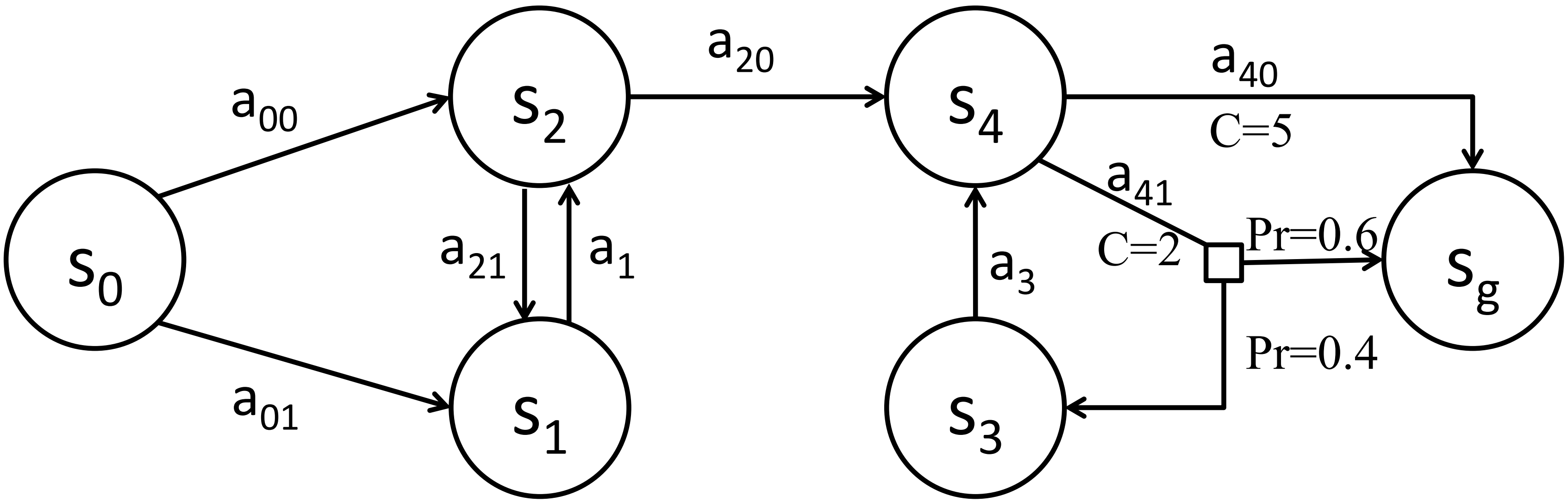
return greedy policy: $\pi^{V_n}(s) = \text{argmin}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') \left[\mathcal{C}(s, a, s') + V_n(s')\right]$

ϵ-consistency

termination condition

Example

(all actions cost 1 unless otherwise stated)



n	$V_n(s_0)$	$V_n(s_1)$	$V_n(s_2)$	$V_n(s_3)$	$V_n(s_4)$
0	3	3	2	2	1
1	3	3	2	2	2.8
2	3	3	3.8	3.8	2.8
3	4	4.8	3.8	3.8	3.52
4	4.8	4.8	4.52	4.52	3.52
5	5.52	5.52	4.52	4.52	3.808
20	5.99921	5.99921	4.99969	4.99969	3.99969

Comments

- Decision-theoretic Algorithm
- Dynamic Programming
- Fixed Point Computation
- Probabilistic version of Bellman-Ford Algorithm
  - for shortest path computation
  - MDP_1 : Stochastic Shortest Path Problem

- Time Complexity
  - one iteration: $O(|\mathcal{S}|^2|\mathcal{A}|)$
  - number of iterations: poly($|\mathcal{S}|$, $|\mathcal{A}|$, $1/\epsilon$, $1/(1-\gamma)$)
- Space Complexity: $O(|\mathcal{S}|)$

Monotonicity

For all n>k

$V_k \leq_p V^* \Rightarrow V_n \leq_p V^*$ (V_n monotonic from below)

$V_k \geq_p V^* \Rightarrow V_n \geq_p V^*$ (V_n monotonic from above)

Changing the Search Space

- Value Iteration
  - Search in value space
  - Compute the resulting policy
- Policy Iteration
  - Search in policy space
  - Compute the resulting value

Policy iteration [Howard'60]

- assign an arbitrary assignment of π_0 to each state.
- repeat
  - Policy Evaluation: compute V_{n+1}: the evaluation of π_n
  - Policy Improvement: for all states s
    - compute $\pi_{n+1}(s)$: $\text{argmin}_{a \in Ap(s)} Q_{n+1}(s,a)$
- until $\pi_{n+1} = \pi_n$

costly: $O(n^3)$

Modified Policy Iteration → approximate by value iteration using fixed policy

Advantage

- searching in a finite (policy) space as opposed to uncountably infinite (value) space ⇒ convergence in fewer number of iterations.
- all other properties follow!

Modified Policy iteration

- assign an arbitrary assignment of π_0 to each state.
- repeat
  - Policy Evaluation: compute V_{n+1} the *approx.* evaluation of π_n
  - Policy Improvement: for all states s
    - compute $\pi_{n+1}(s)$: $\text{argmax}_{a\in Ap(s)} Q_{n+1}(s,a)$
- until $\pi_{n+1} = \pi_n$

Advantage

- probably the most competitive synchronous dynamic programming algorithm.

Applications

- Stochastic Games
- Robotics: navigation, helicopter manuevers…
- Finance: options, investments
- Communication Networks
- Medicine: Radiation planning for cancer
- Controlling workflows
- Optimize bidding decisions in auctions
- Traffic flow optimization
- Aircraft queueing for landing; airline meal provisioning
- Optimizing software on mobiles
- Forest firefighting
- …