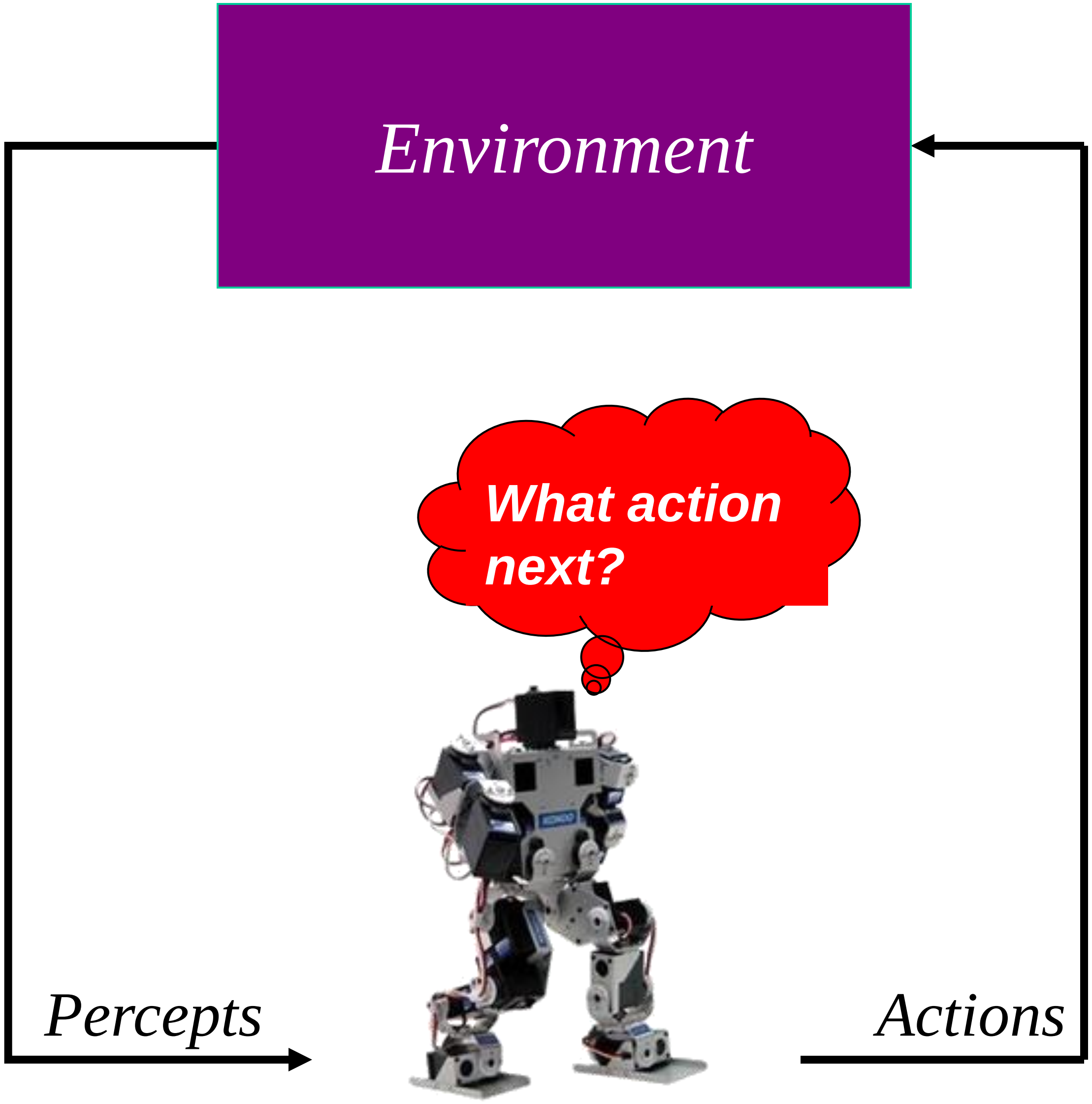
Markov Decision Processes
Chapter 17

Mausam

Planning Agent

Static vs. Dynamic



Environment

What action next?

Percepts

Actions

Fully
vs.
Partially
Observable

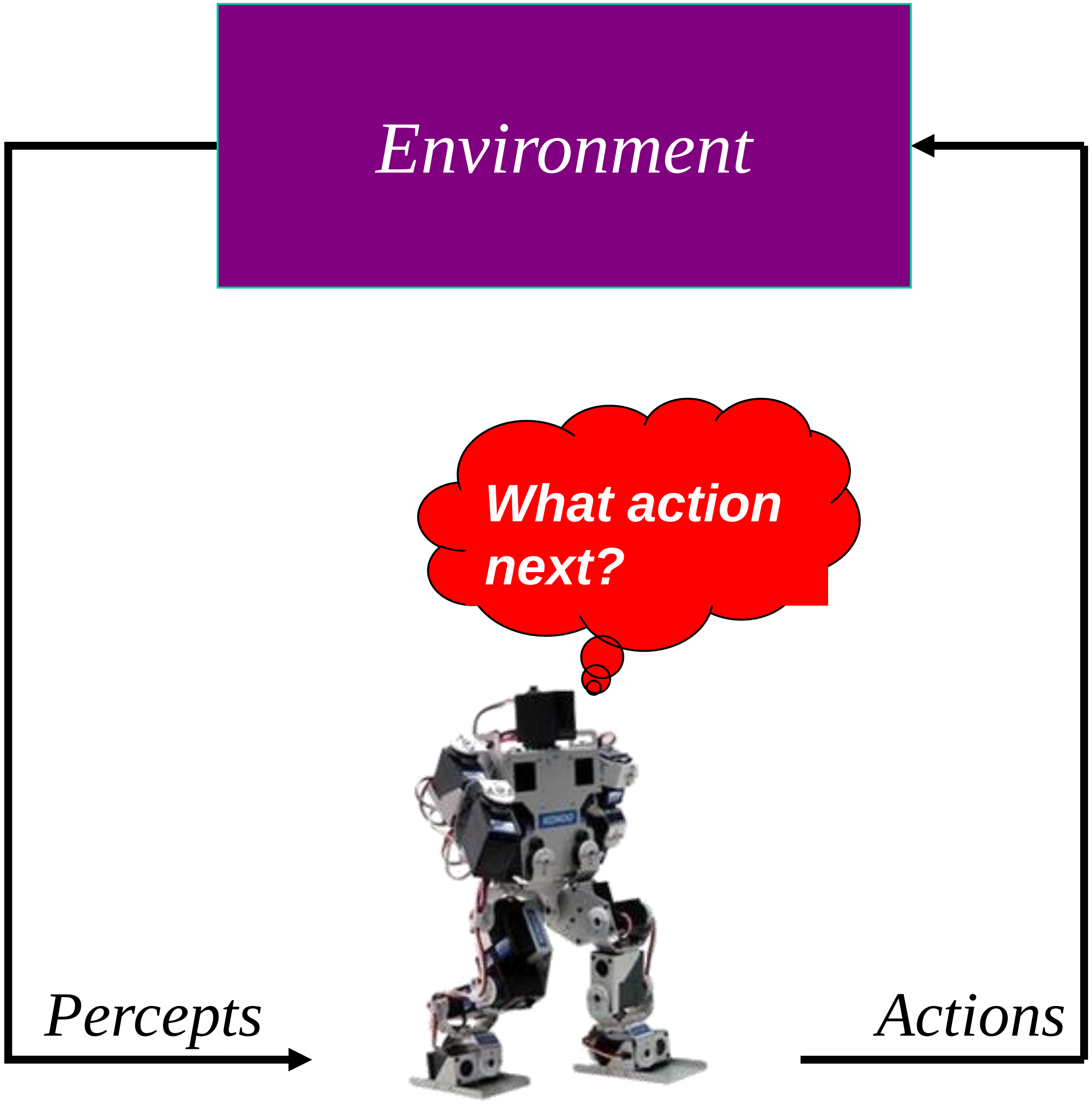
Perfect
vs.
Noisy

Deterministic
vs.
Stochastic

Instantaneous
vs.
Durative

Classical Planning

Static



Environment

Fully Observable

What action next?

Deterministic

Instantaneous

Perfect

Percepts

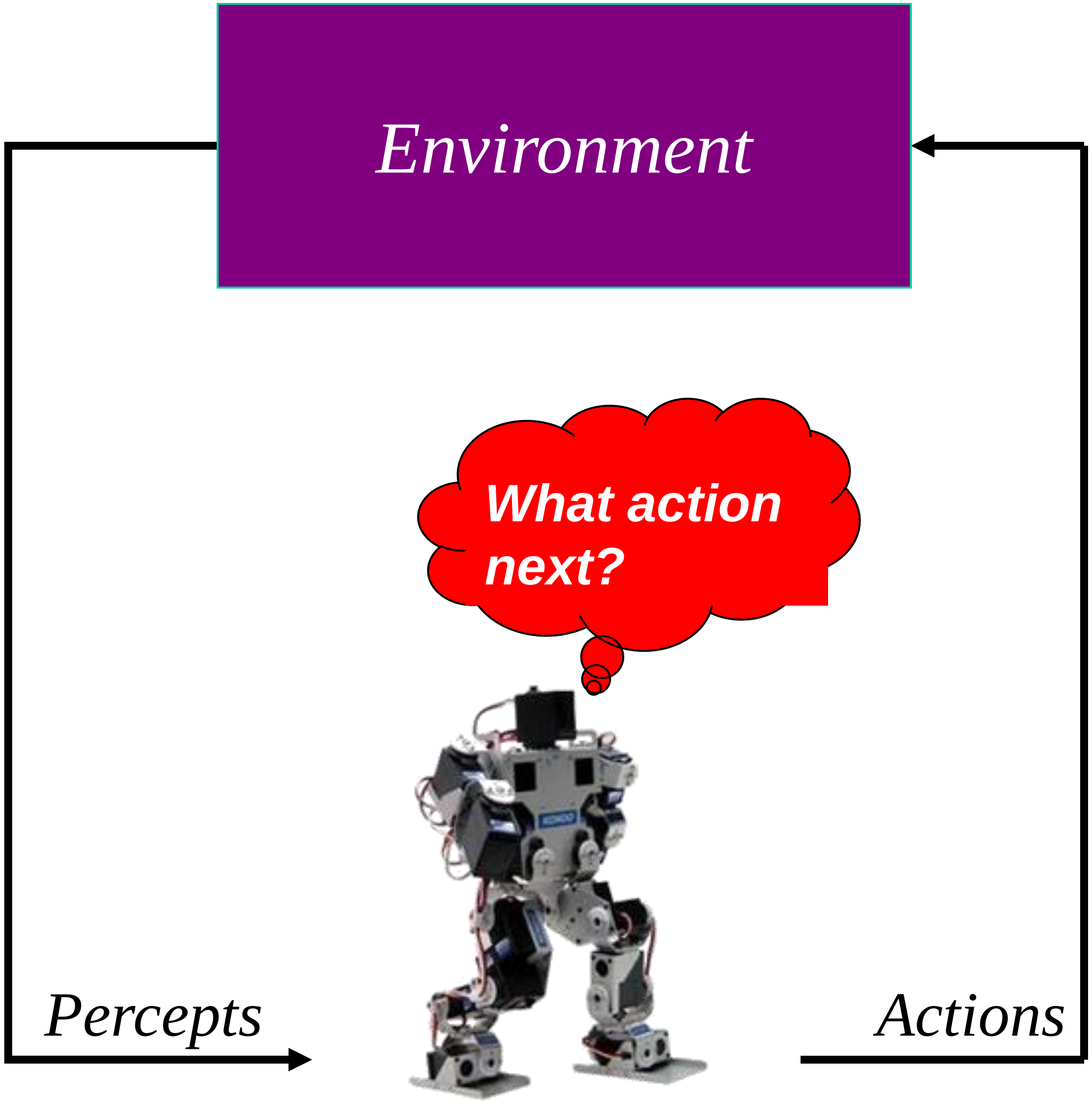
Actions

Stochastic Planning: MDPs

Static

Fully Observable

Perfect



Environment

What action next?

Percepts

Actions

Stochastic

Instantaneous

MDP vs. Decision Theory

- Decision theory - episodic
- MDP -- sequential

Markov Decision Process (MDP)

- $\mathcal{S}$: A set of states

Markov Decision Process (MDP)

- $\mathcal{S}$: A set of states
- $\mathcal{A}$: A set of actions

Markov Decision Process (MDP)

- $\mathcal{S}$: A set of states
- $\mathcal{A}$: A set of actions
- $\mathcal{T}$(s,a,s'): transition model

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

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

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

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

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

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

Markov Decision Process (MDP)

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

Markov Decision Process (MDP)

- $\mathcal{S}$: A set of states
- $\mathcal{A}$: A set of actions
- $\mathcal{T}$(s,a,s'): transition model
- γ: discount factor
- $\mathcal{R}$(s,a,s'): reward model

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

Objective of an MDP

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

- which optimizes

$$\begin{array}{ll}\bullet & \text{minimizes}\\ \bullet & \text{maximizes}\\ \bullet & \text{maximizes}\end{array} \begin{pmatrix}\text{discounted}\\ \text{or}\\ \text{undiscount.}\end{pmatrix} \begin{array}{l}\text{expected cost to reach a goal}\\ \text{expected reward}\\ \text{expected (reward-cost)}\end{array}$$

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

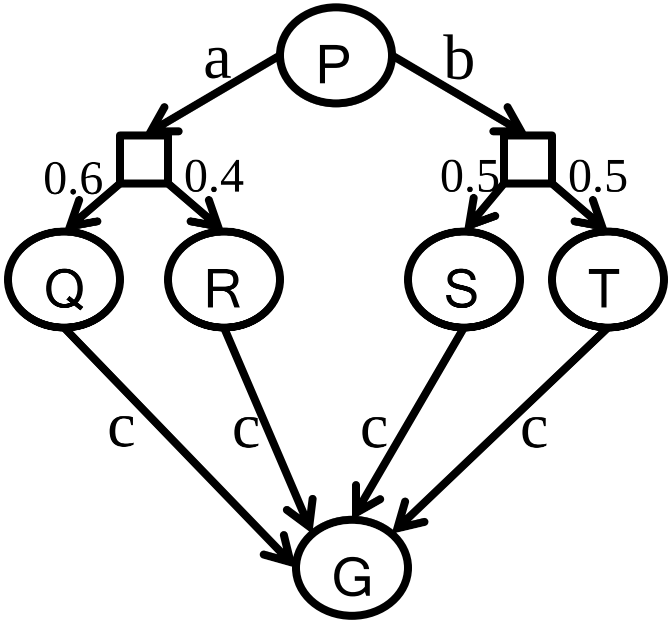
Examples of MDPs

- 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

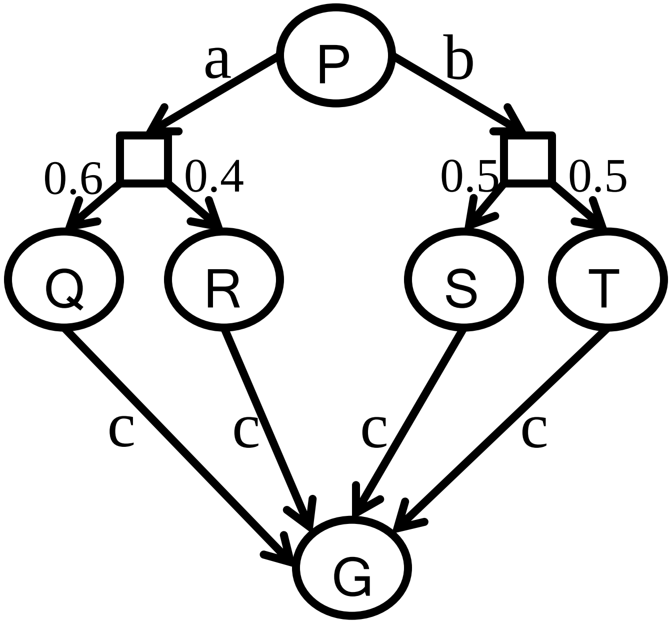
- Oversubscription Planning: Non absorbing goals, Reward Max. MDP
  - $<\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{G}, \mathcal{R}, s_0>$
  - Relatively recent model

Acyclic vs. Cyclic MDPs



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

Acyclic vs. Cyclic MDPs

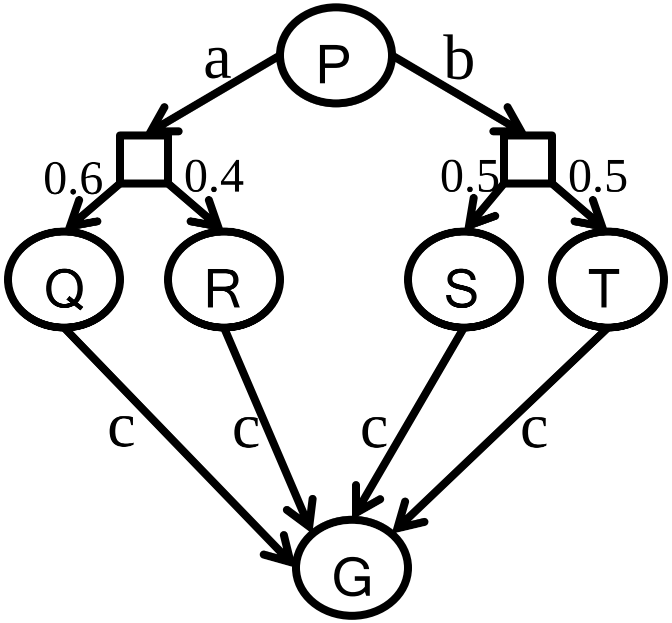


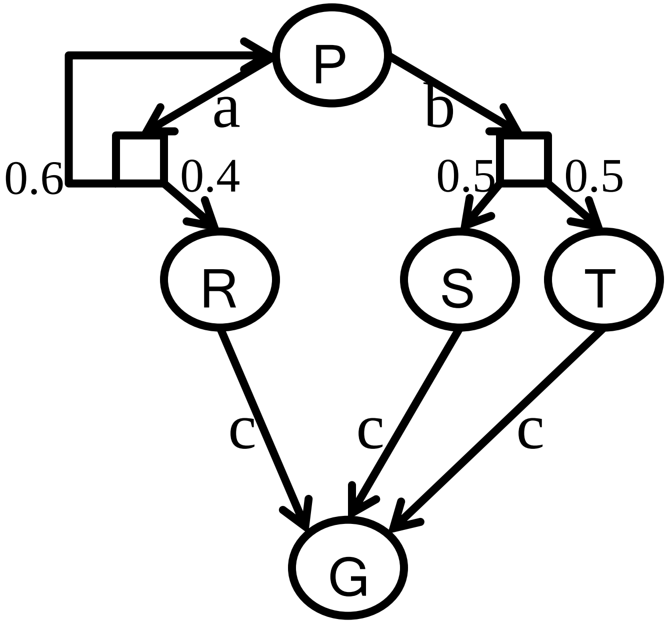
C(a) = 5, C(b) = 10, C(c) =1

Expectimin works

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

Acyclic vs. Cyclic MDPs



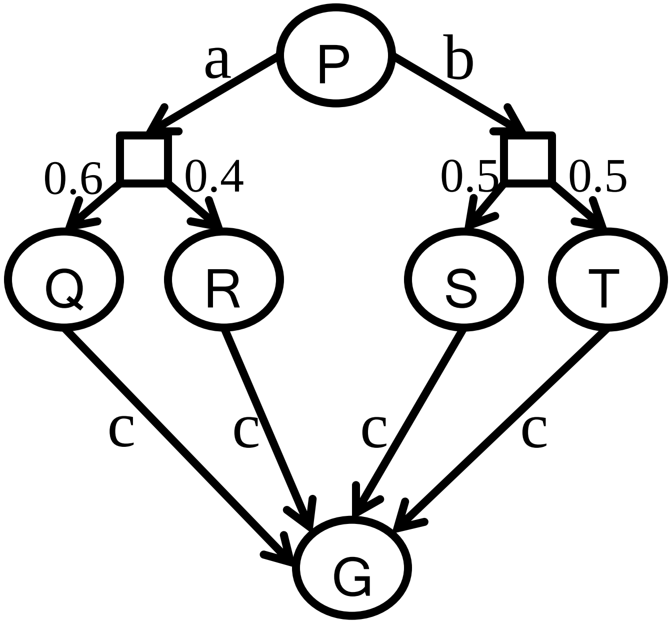


C(a) = 5, C(b) = 10, C(c) =1

Expectimin works

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

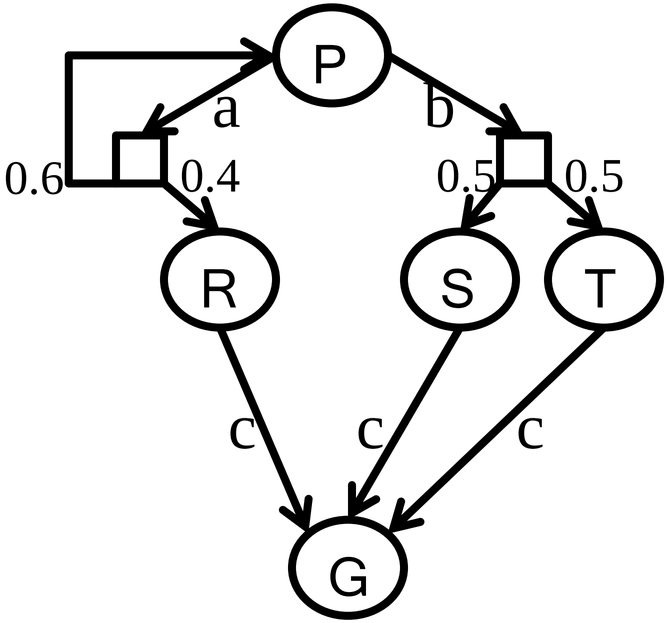
Acyclic vs. Cyclic MDPs



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

Expectimin works

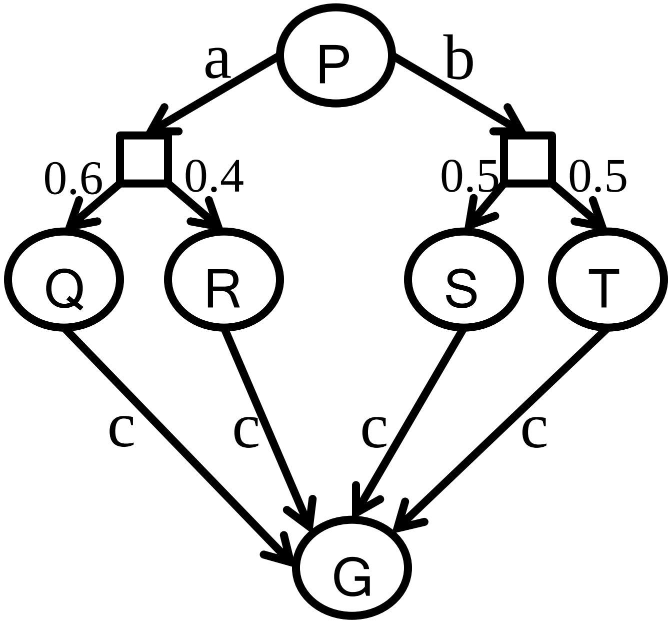
- V(Q/R/S/T) = 1
- V(P) = 6 – action a

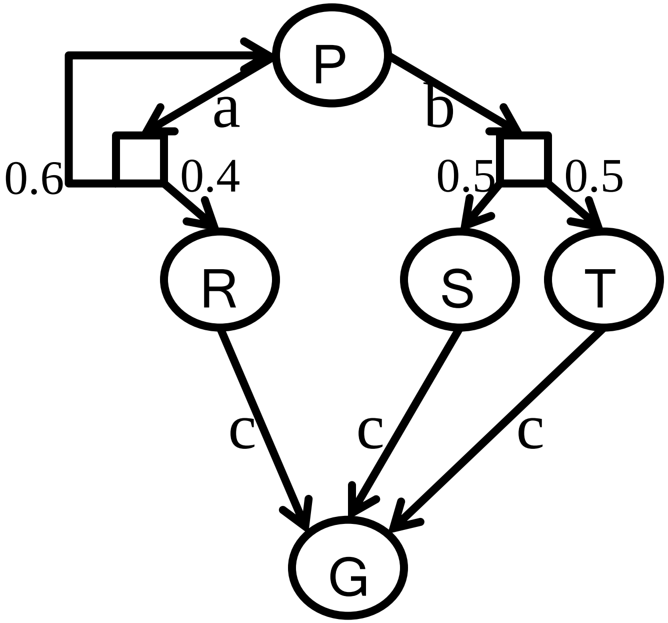


Expectimin doesn’t work

- infinite loop

Acyclic vs. Cyclic MDPs





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

Expectimin works

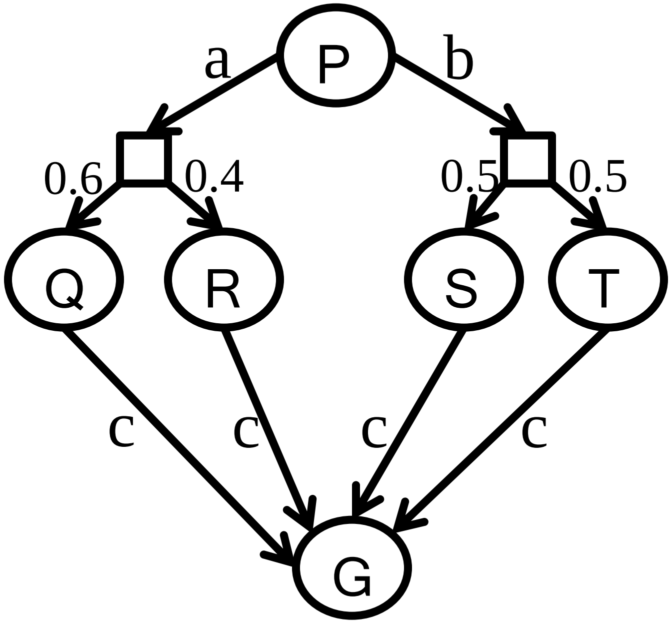
- V(Q/R/S/T) = 1
- V(P) = 6 – action a

Expectimin doesn't work

- infinite loop

- V(R/S/T) = 1
- Q(P,b) = 11
- Q(P,a) = ????

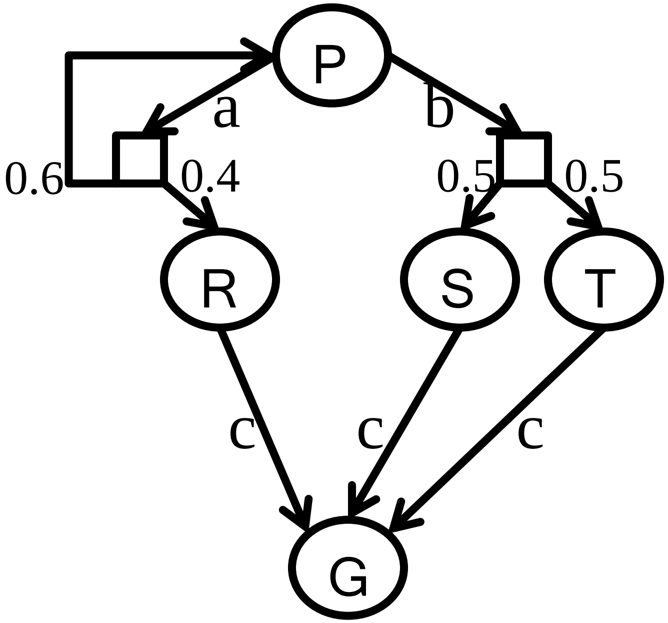
Acyclic vs. Cyclic MDPs



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

Expectimin works

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

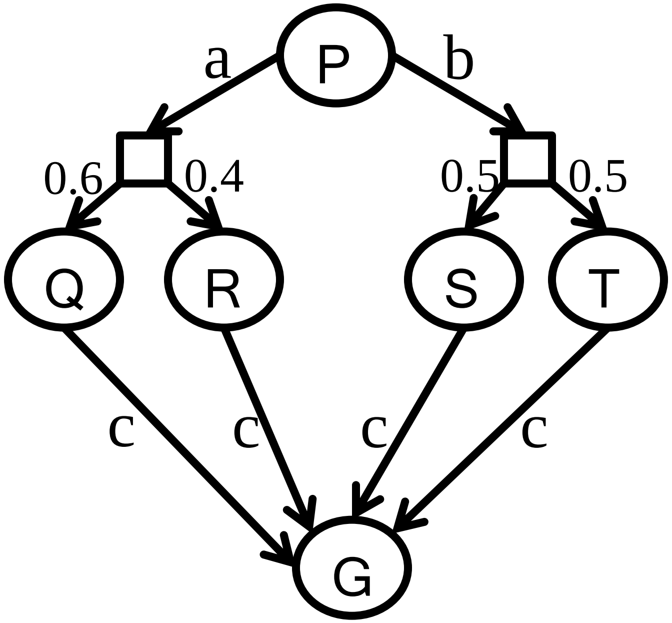


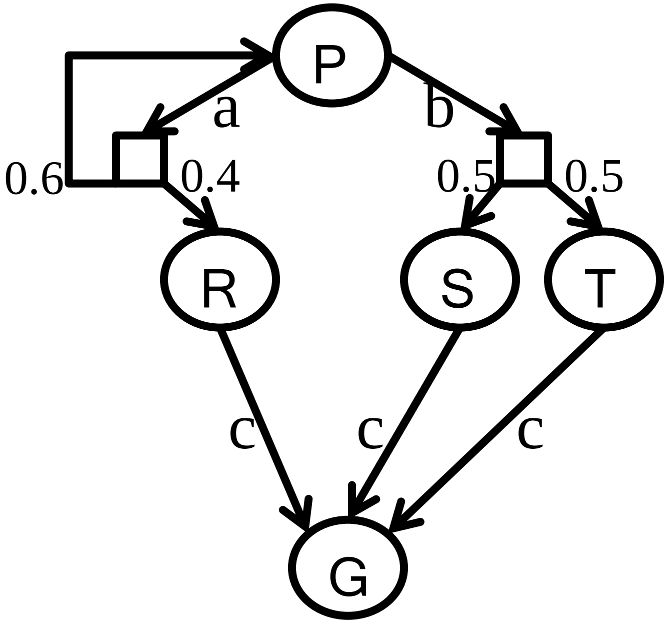
Expectimin doesn't work

- infinite loop

- V(R/S/T) = 1
- Q(P,b) = 11
- Q(P,a) = ????
- suppose I decide to take a in P

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

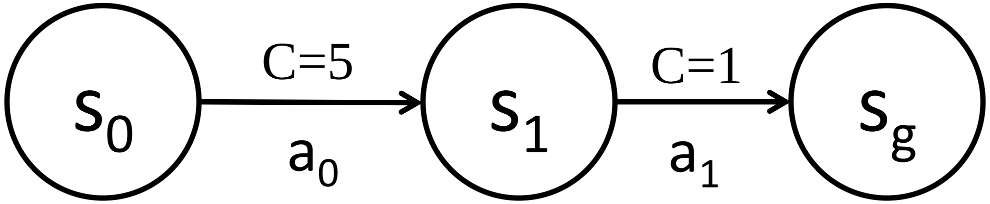
Policy Evaluation

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

Deterministic MDPs

- Policy Graph for π

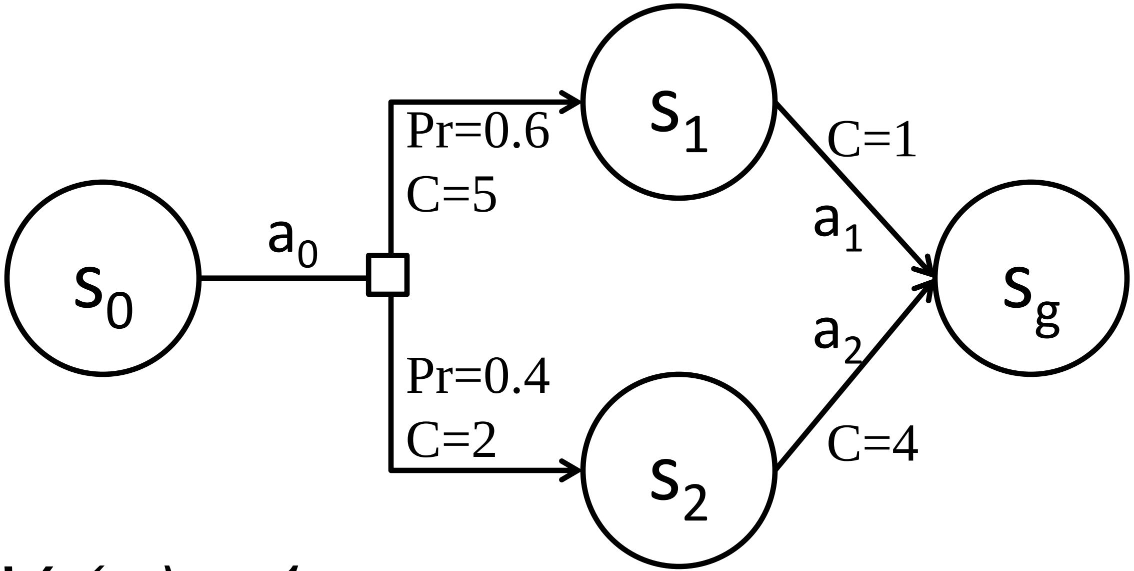
  $\pi(s_0) = a_0; \pi(s_1) = a_1$



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

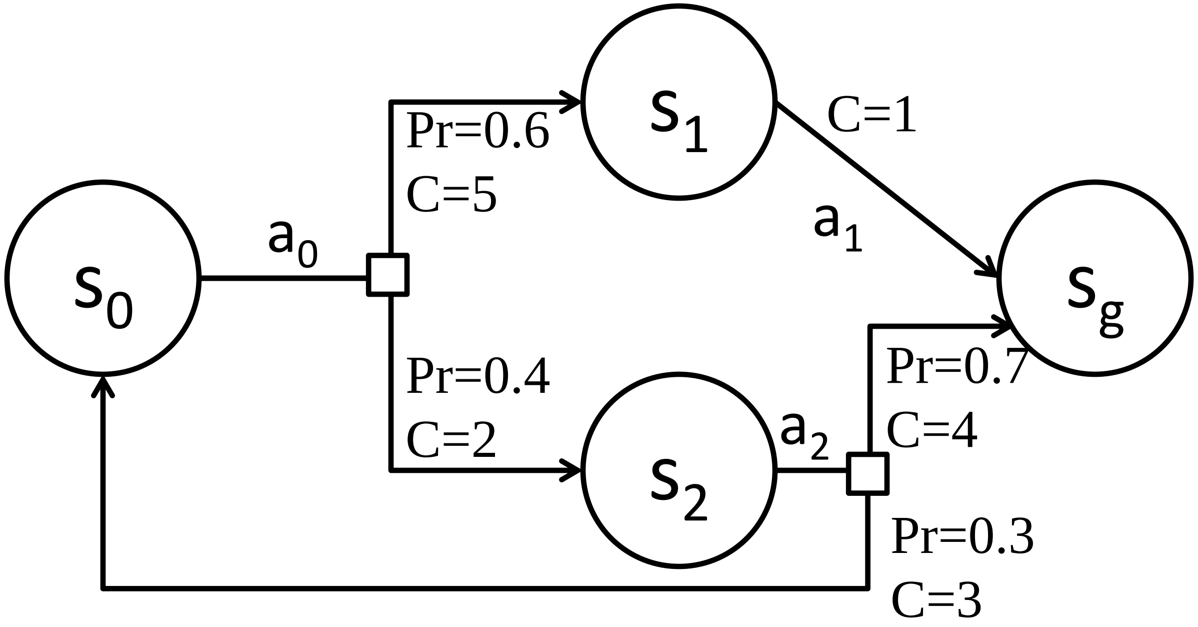
Acyclic MDPs

- Policy Graph for π



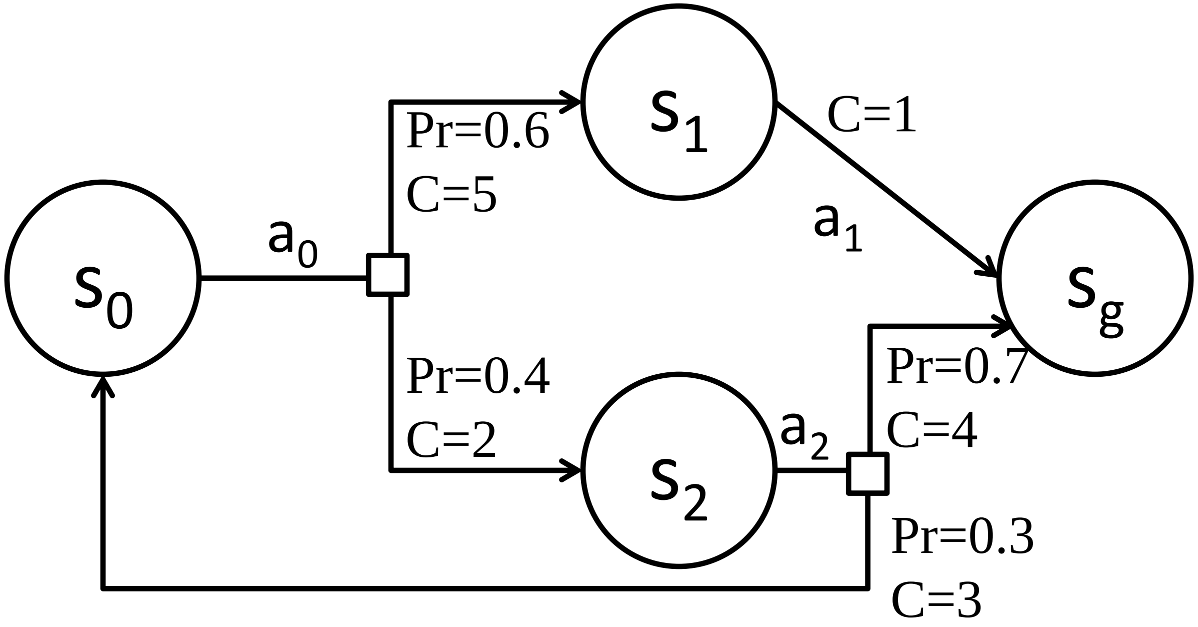
- $V^{\pi}(s_1) = 1$
- $V^{\pi}(s_2) = 4$
- $V^{\pi}(s_0) = 0.6(5+1) + 0.4(2+4) = 6$

General MDPs can be cyclic!



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

General SSPs can be cyclic!



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

Policy Evaluation (Approach 1)

- Solving the System of Linear Equations

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

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

Policy Evaluation (Approach 1)

- Solving the System of Linear Equations

$$
\begin{aligned}
V^\pi(s) &= 0 \qquad \text{if } s \in \mathcal{G} \\
&= \qquad\qquad\qquad [\mathcal{C}(s, \pi(s), s')
\end{aligned}
$$

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

Policy Evaluation (Approach 1)

- Solving the System of Linear Equations

$$\begin{aligned} V^{\pi}(s) &= 0 \qquad \text{if } s \in \mathcal{G} \\ &= \qquad\qquad [\mathcal{C}(s, \pi(s), s') + V^{\pi}(s')] \end{aligned}$$

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

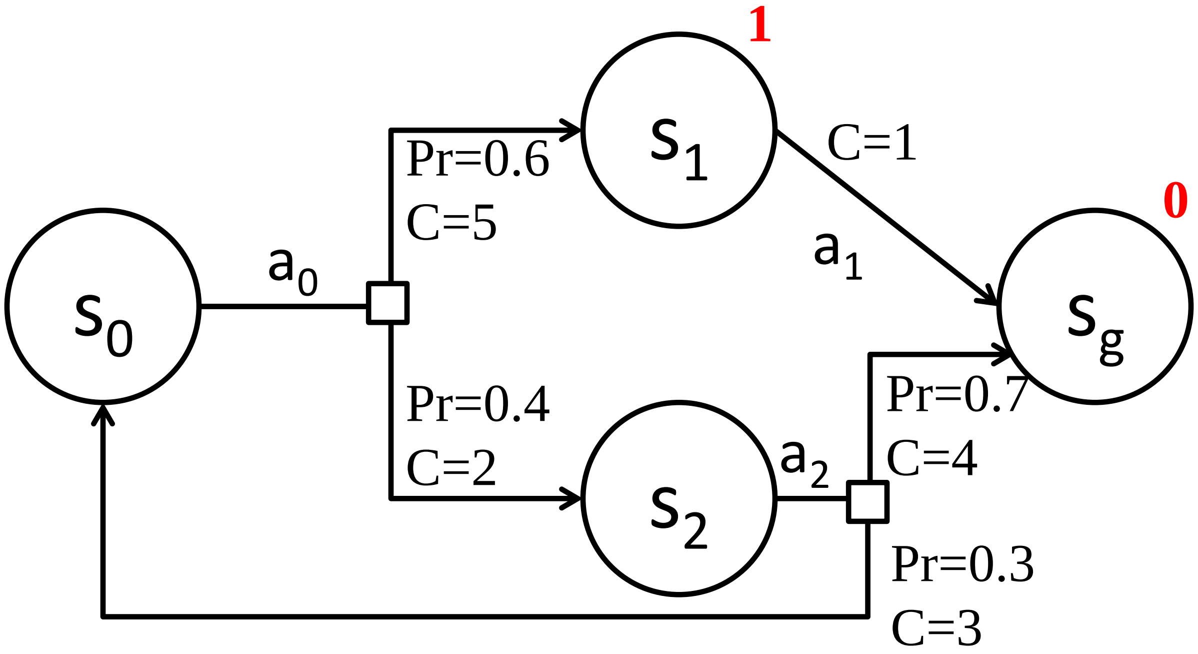
Policy Evaluation (Approach 1)

- Solving the System of Linear Equations

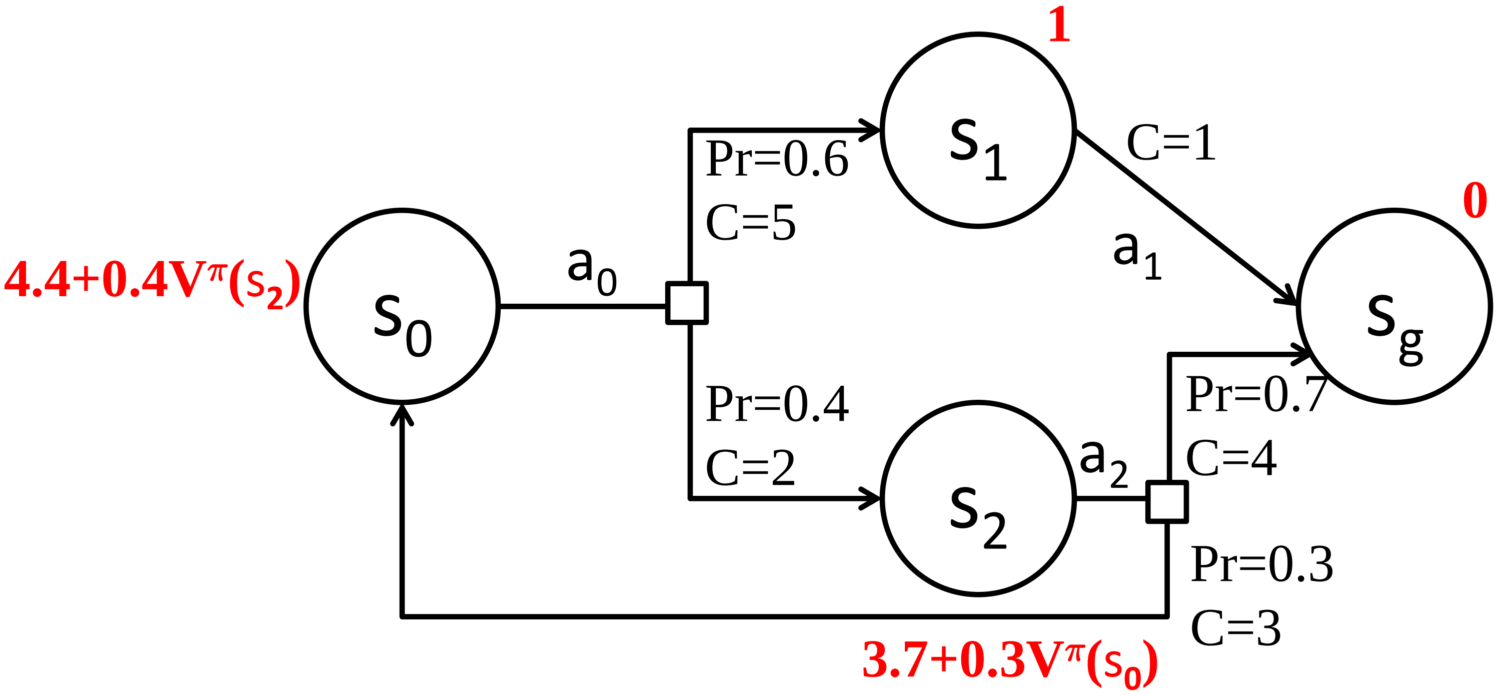
$$\boxed{\begin{aligned} V^{\pi}(s) &= 0 \qquad \text{if } s \in \mathcal{G} \\ &= \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s') \left[\mathcal{C}(s, \pi(s), s') + V^{\pi}(s')\right] \end{aligned}}$$

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

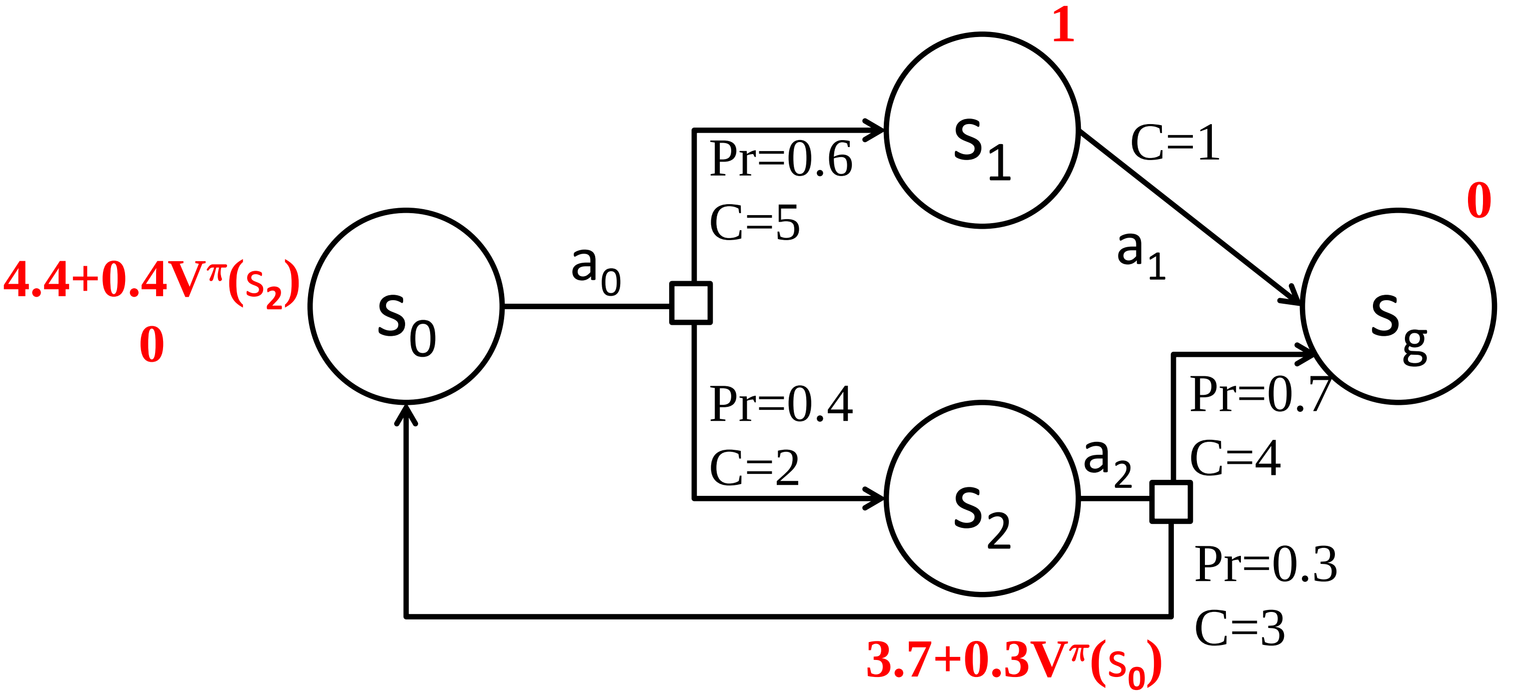
Iterative Policy Evaluation



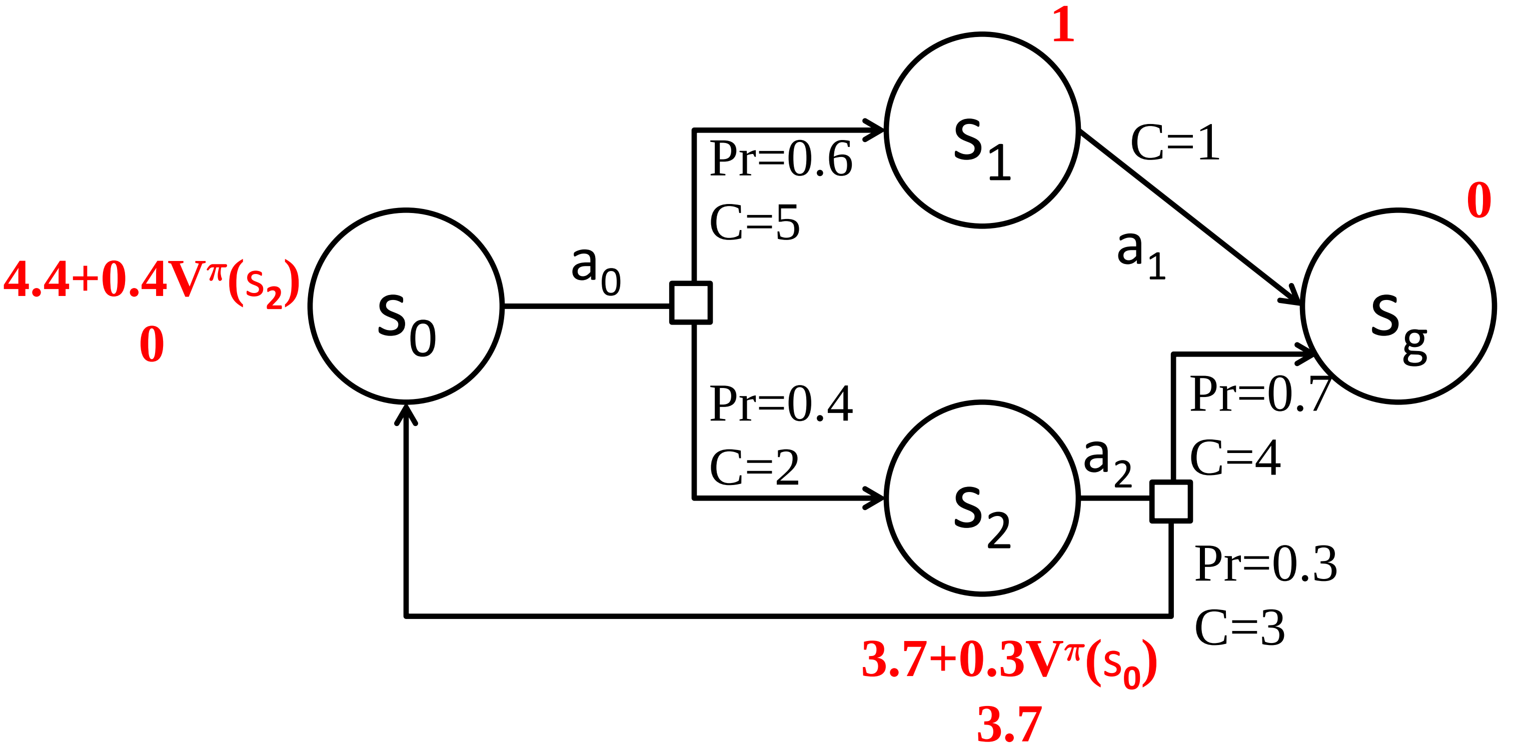
Iterative Policy Evaluation



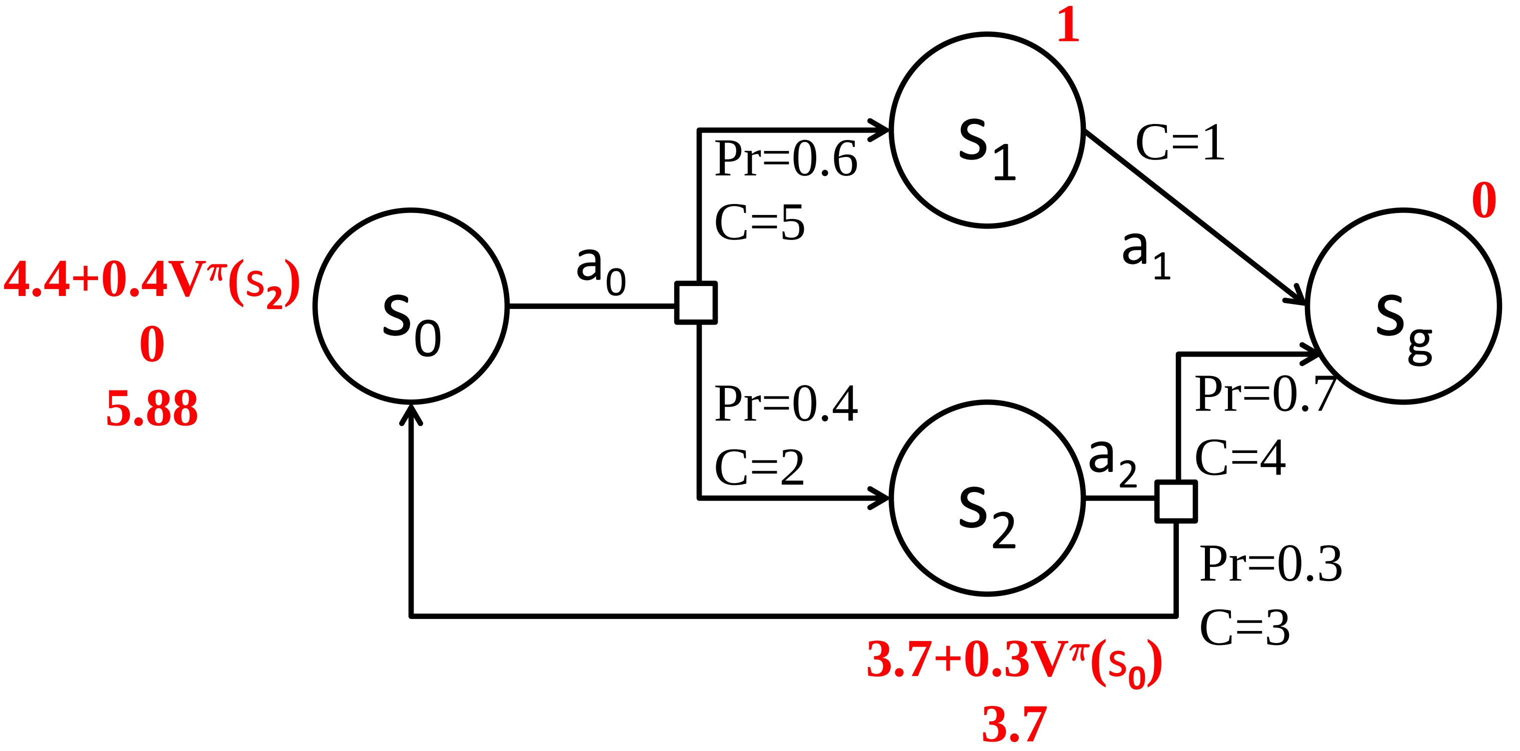
Iterative Policy Evaluation



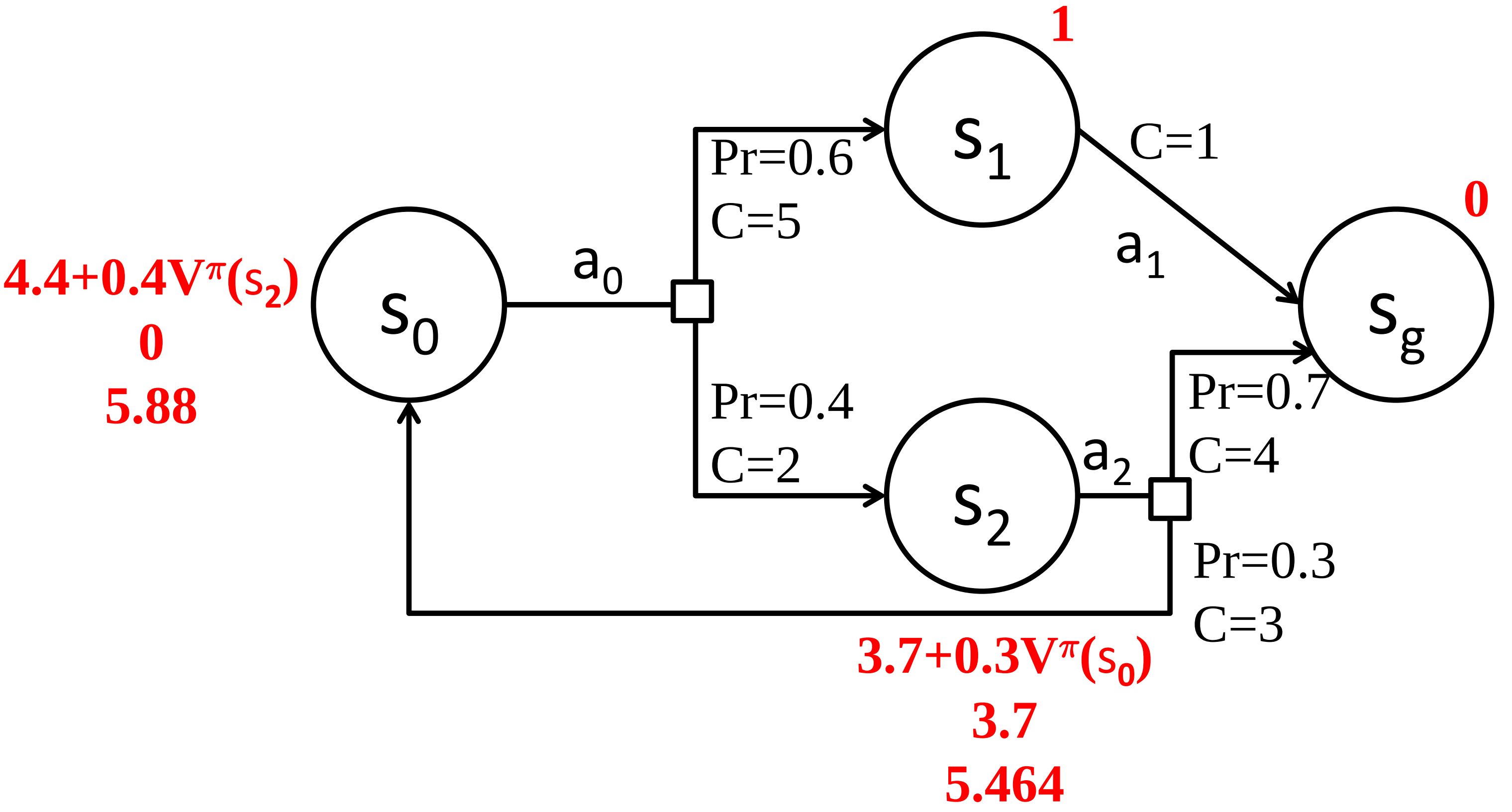
Iterative Policy Evaluation



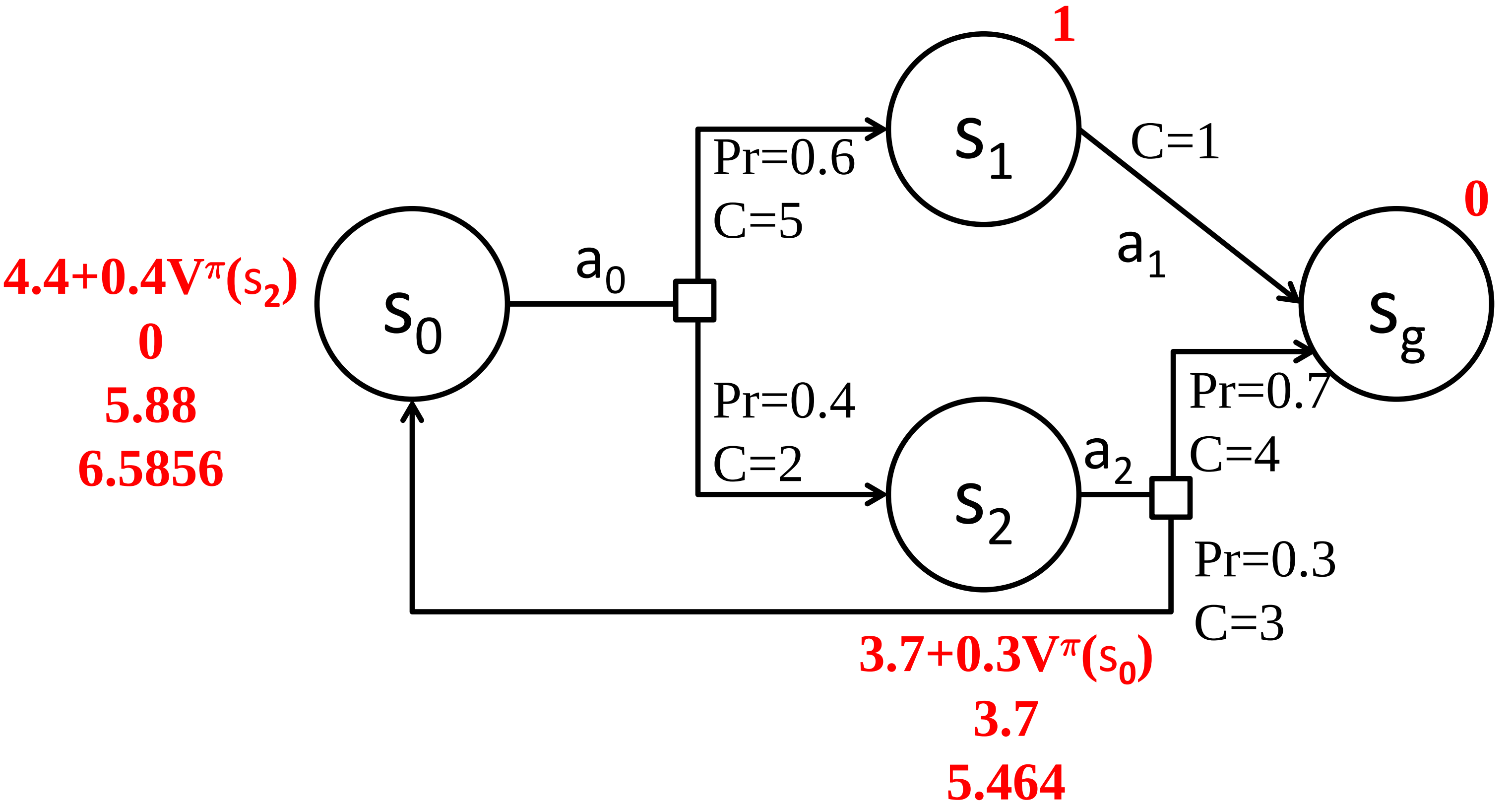
Iterative Policy Evaluation



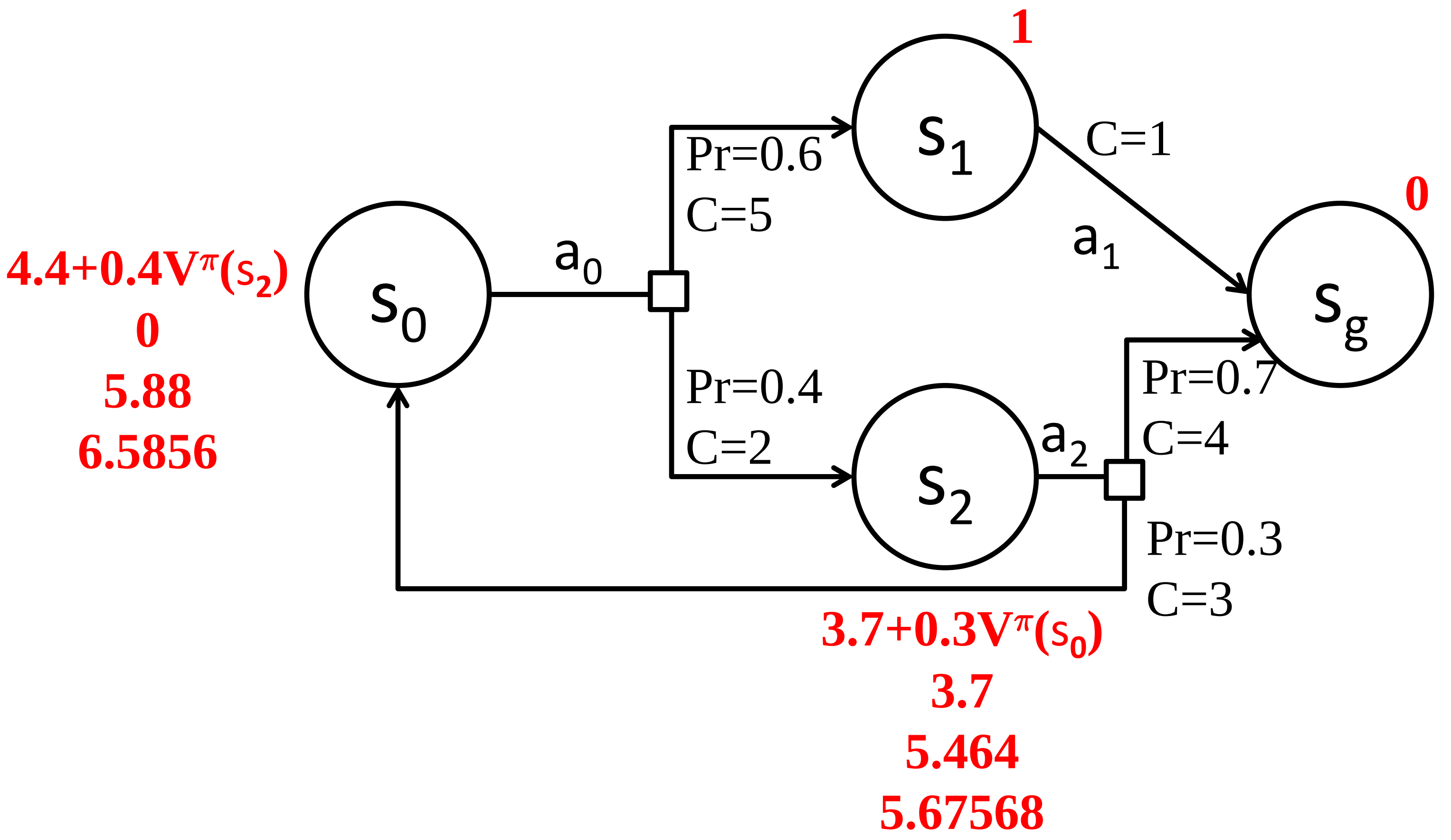
Iterative Policy Evaluation



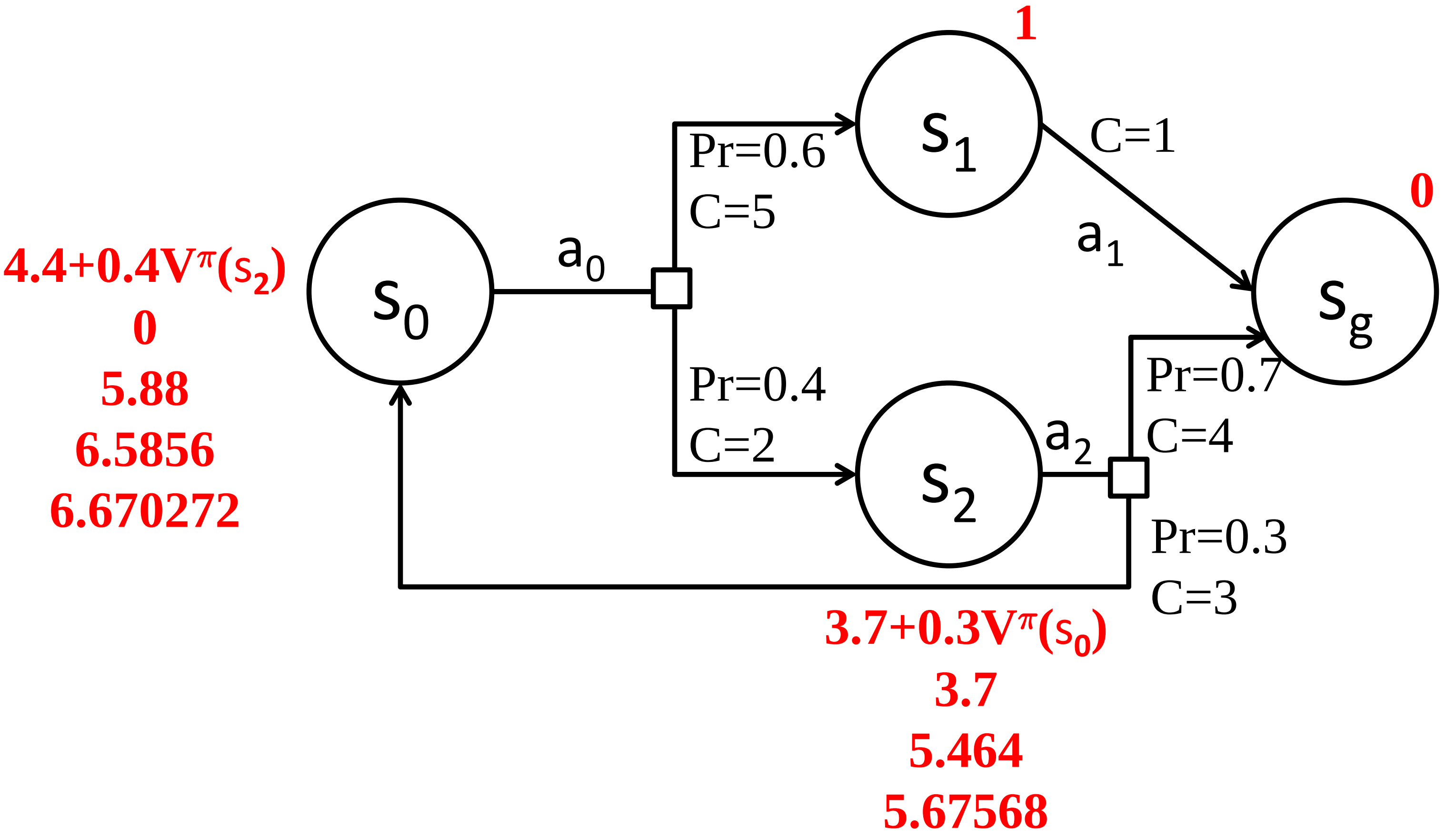
Iterative Policy Evaluation



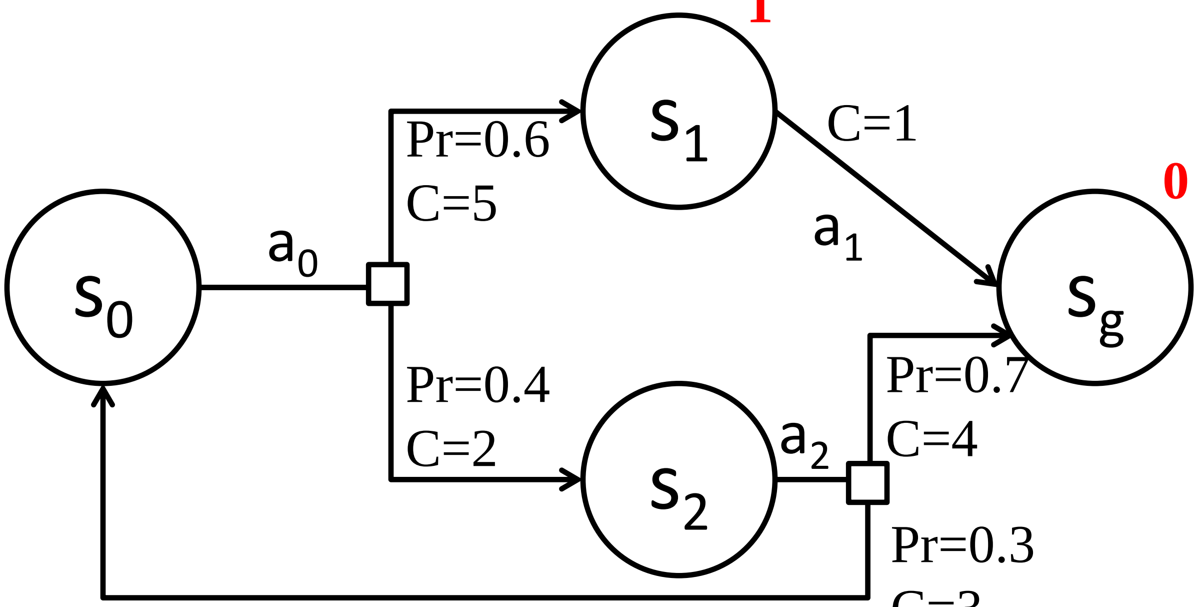
Iterative Policy Evaluation



Iterative Policy Evaluation



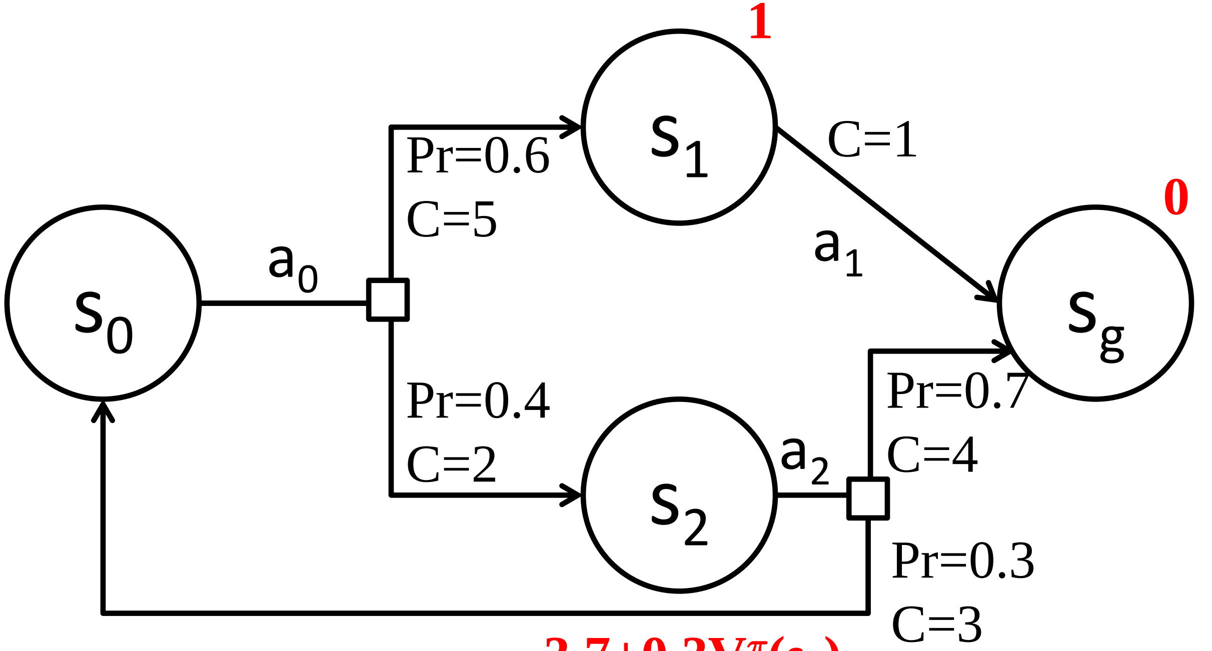
Iterative Policy Evaluation



$4.4+0.4V^{\pi}(s_2)$
0
5.88
6.5856
6.670272

$3.7+0.3V^{\pi}(s_0)$
3.7
5.464
5.67568
5.7010816

Iterative Policy Evaluation



$4.4+0.4V^{\pi}(s_2)$

0

5.88

6.5856

6.670272

6.68043..

$3.7+0.3V^{\pi}(s_0)$

3.7

5.464

5.67568

5.7010816

Iterative Policy Evaluation

$4.4+0.4V^{\pi}(s_2)$

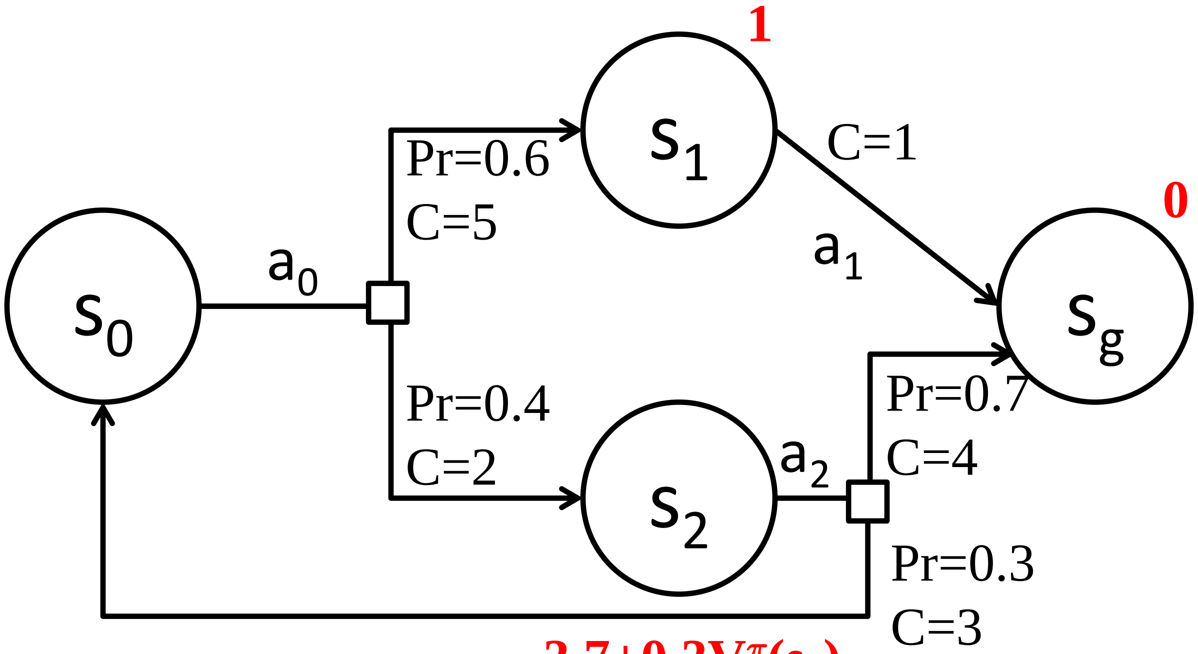
0

5.88

6.5856

6.670272

6.68043..



$3.7+0.3V^{\pi}(s_0)$

3.7

5.464

5.67568

5.7010816

5.704129…

Policy Evaluation (Approach 2)

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

Policy Evaluation (Approach 2)

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

$$\Big\downarrow$$

iterative refinement

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

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

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

Policy Evaluation (Approach 2)

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

iterative refinement

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

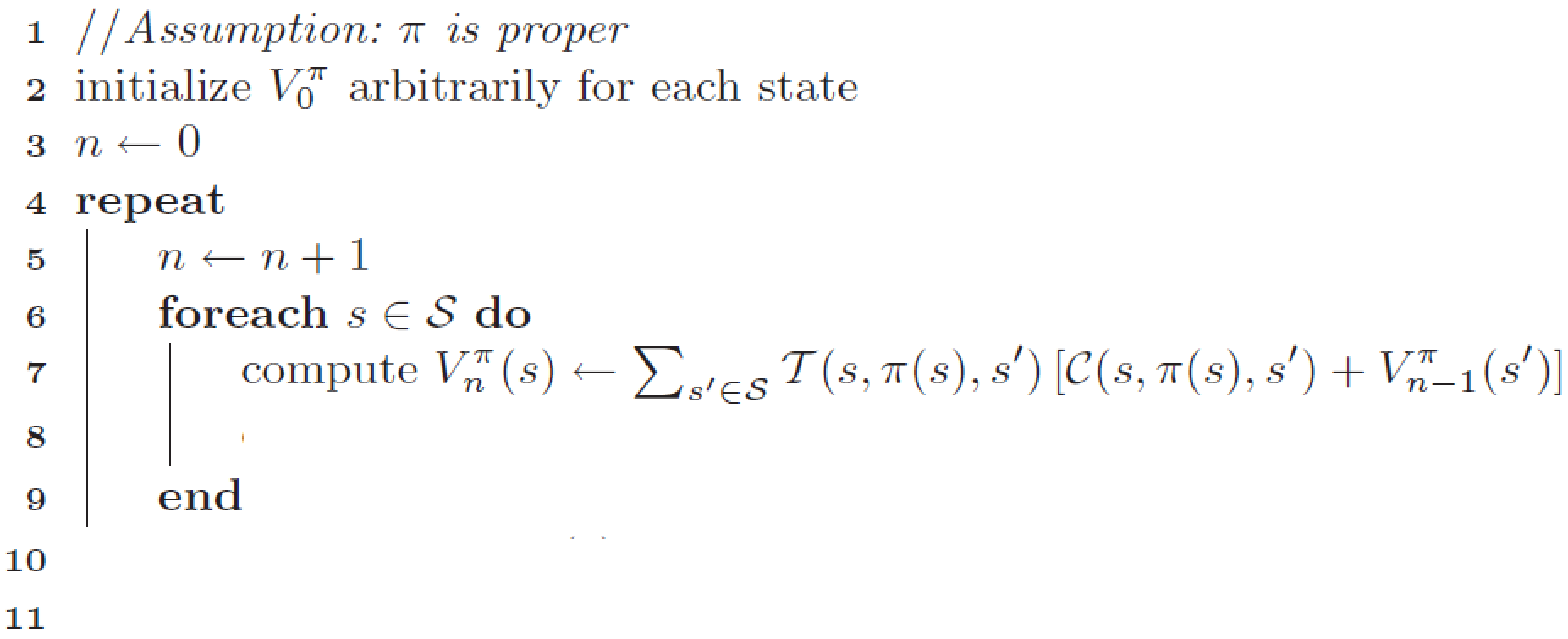
Iterative Policy Evaluation

```
//Assumption: $\pi$ is proper
initialize $V_0^\pi$ arbitrarily for each state
```

Iterative Policy Evaluation

```
//Assumption: $\pi$ is proper
initialize $V_0^\pi$ arbitrarily for each state
```

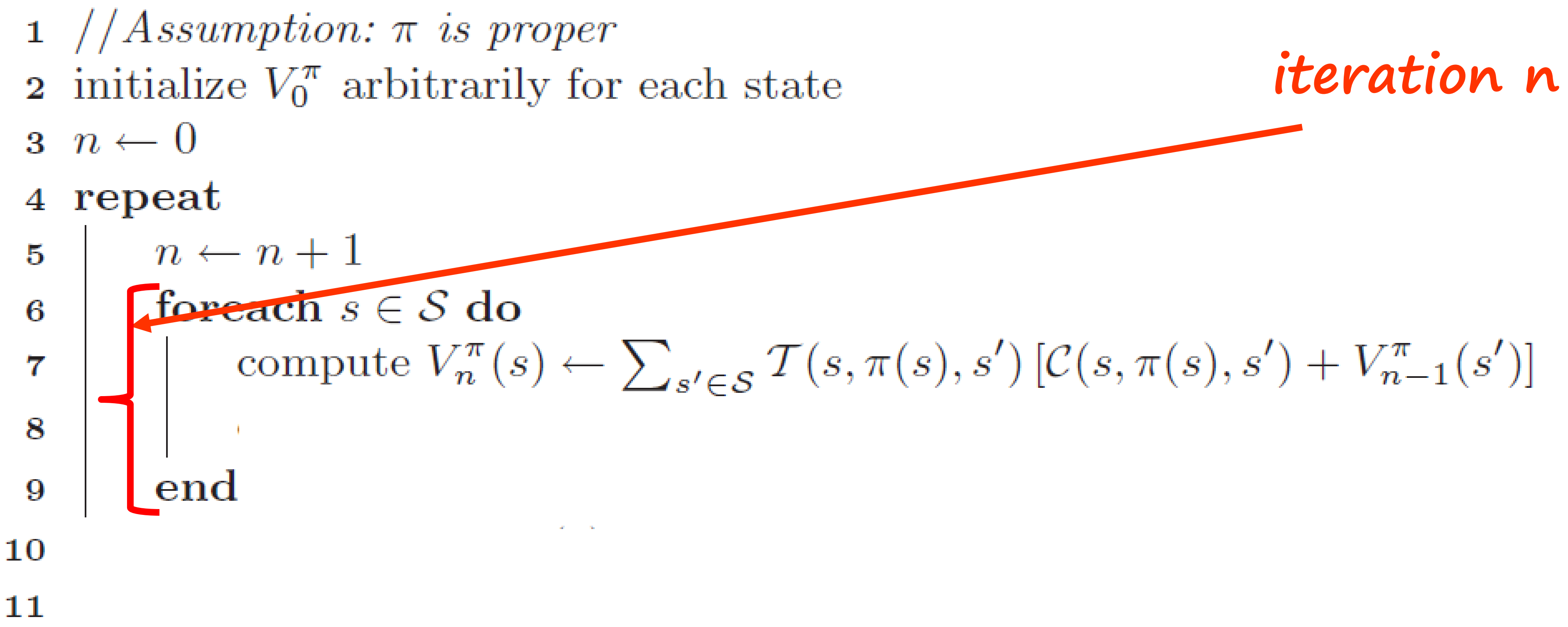
Iterative Policy Evaluation



```
//Assumption: $\pi$ is proper
initialize $V_0^\pi$ arbitrarily for each state
$n \leftarrow 0$
repeat
    $n \leftarrow n + 1$
    foreach $s \in \mathcal{S}$ do
        compute $V_n^\pi(s) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s') \left[\mathcal{C}(s, \pi(s), s') + V_{n-1}^\pi(s')\right]$

    end
```

Iterative Policy Evaluation



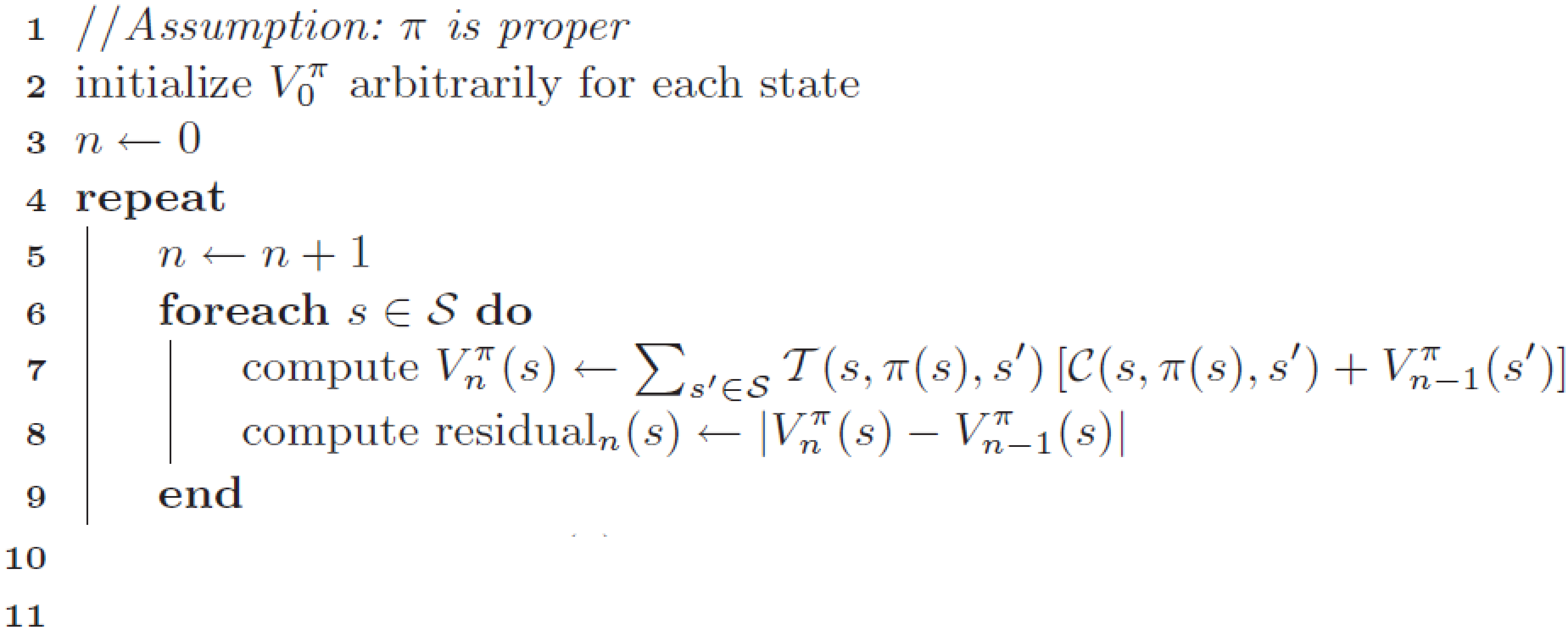
```
//Assumption: $\pi$ is proper
initialize $V_0^\pi$ arbitrarily for each state
$n \leftarrow 0$
repeat
    $n \leftarrow n + 1$
    foreach $s \in \mathcal{S}$ do
        compute $V_n^\pi(s) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s') [\mathcal{C}(s, \pi(s), s') + V_{n-1}^\pi(s')]$

    end


```

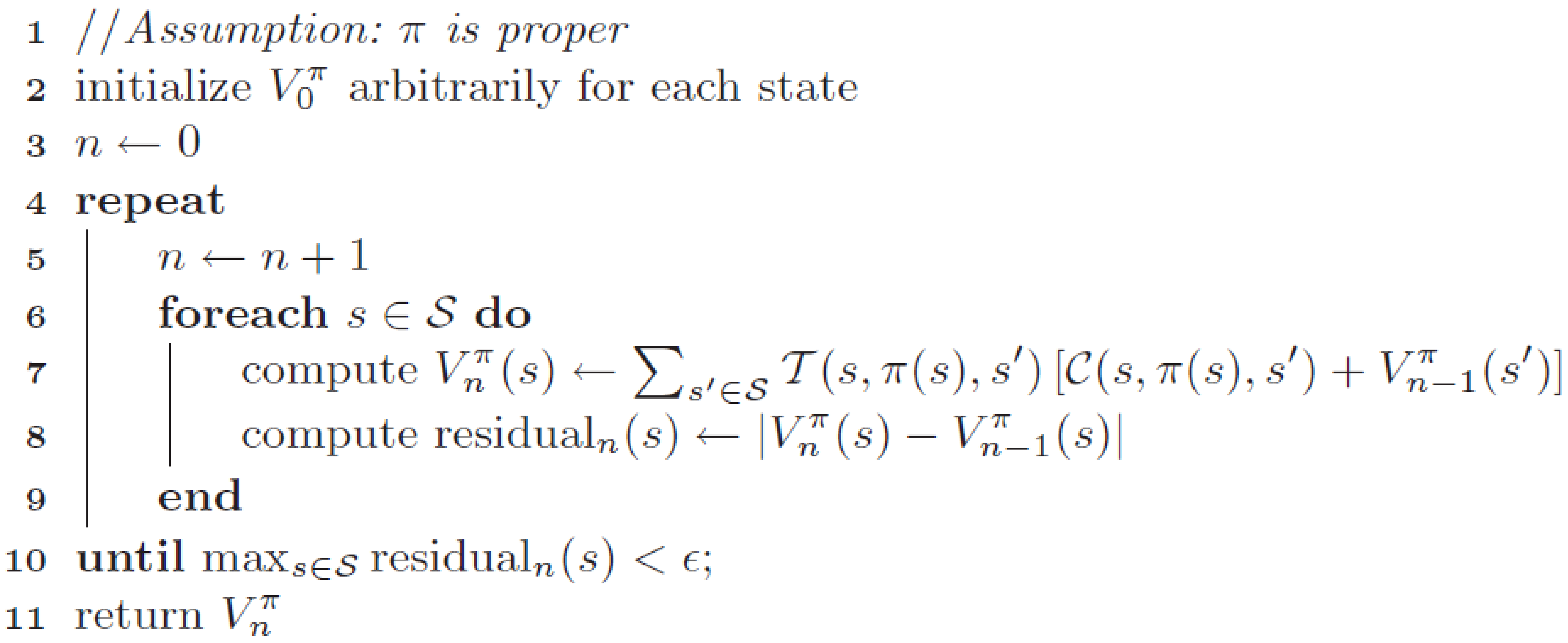
iteration n

Iterative Policy Evaluation



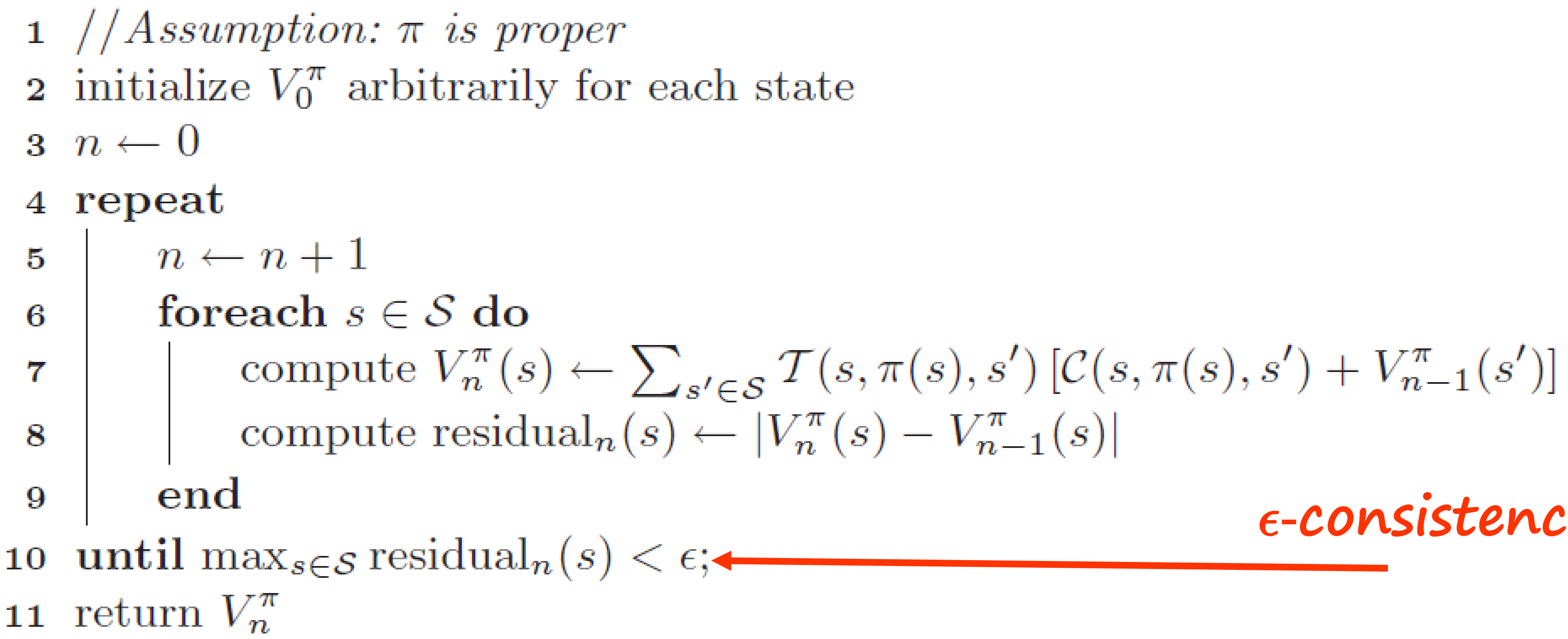
```
//Assumption: $\pi$ is proper
initialize $V_0^\pi$ arbitrarily for each state
$n \leftarrow 0$
repeat
    $n \leftarrow n + 1$
    foreach $s \in \mathcal{S}$ do
        compute $V_n^\pi(s) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s') \left[\mathcal{C}(s, \pi(s), s') + V_{n-1}^\pi(s')\right]$
        compute residual$_n(s) \leftarrow |V_n^\pi(s) - V_{n-1}^\pi(s)|$
    end
```

Iterative Policy Evaluation



```
//Assumption: π is proper
initialize $V_0^\pi$ arbitrarily for each state
$n \leftarrow 0$
repeat
    $n \leftarrow n + 1$
    foreach $s \in \mathcal{S}$ do
        compute $V_n^\pi(s) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s') [\mathcal{C}(s, \pi(s), s') + V_{n-1}^\pi(s')]$
        compute $\text{residual}_n(s) \leftarrow |V_n^\pi(s) - V_{n-1}^\pi(s)|$
    end
until $\max_{s \in \mathcal{S}} \text{residual}_n(s) < \epsilon$;
return $V_n^\pi$
```

Iterative Policy Evaluation



```
//Assumption: $\pi$ is proper
initialize $V_0^\pi$ arbitrarily for each state
$n \leftarrow 0$
repeat
    $n \leftarrow n + 1$
    foreach $s \in \mathcal{S}$ do
        compute $V_n^\pi(s) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s') \left[\mathcal{C}(s, \pi(s), s') + V_{n-1}^\pi(s')\right]$
        compute residual$_n(s) \leftarrow |V_n^\pi(s) - V_{n-1}^\pi(s)|$
    end
until $\max_{s \in \mathcal{S}}$ residual$_n(s) < \epsilon$;
return $V_n^\pi$
```

ϵ-consistency

Iterative Policy Evaluation

```
//Assumption: $\pi$ is proper
initialize $V_0^\pi$ arbitrarily for each state
$n \leftarrow 0$
repeat
    $n \leftarrow n + 1$
    foreach $s \in \mathcal{S}$ do
        compute $V_n^\pi(s) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s') \left[\mathcal{C}(s, \pi(s), s') + V_{n-1}^\pi(s')\right]$
        compute $\text{residual}_n(s) \leftarrow |V_n^\pi(s) - V_{n-1}^\pi(s)|$
    end
until $\max_{s \in \mathcal{S}} \text{residual}_n(s) < \epsilon$;
return $V_n^\pi$
```

ϵ-consistency

termination condition

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

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

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

Policy Evaluation → Value Iteration (Bellman Equations for MDP_1)

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

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

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

- <$\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, $\mathcal{C}$,$\mathcal{G}$, s$_0$>
- Define V*(s) {optimal cost} as the minimum expected cost to reach a goal from this state.
- V* should satisfy the following equation:

$$
\begin{aligned}
V^*(s) &= 0 \qquad \text{if } s \in \mathcal{G} \\
&= \qquad\qquad\qquad [\mathcal{C}(s,a,s') + V^*(s')]
\end{aligned}
$$

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

- <$\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, $\mathcal{C}$,$\mathcal{G}$, s$_0$>
- Define V*(s) {optimal cost} as the minimum expected cost to reach a goal from this state.
- V* should satisfy the following equation:

$$\boxed{\begin{aligned} V^*(s) &= 0 \qquad \text{if } s \in \mathcal{G} \\ &= \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s')\left[\mathcal{C}(s,a,s') + V^*(s')\right] \end{aligned}}$$

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

- <$\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, $\mathcal{C}$,$\mathcal{G}$, s$_0$>
- Define V*(s) {optimal cost} as the minimum expected cost to reach a goal from this state.
- V* should satisfy the following equation:

$$\boxed{\begin{aligned} V^*(s) &= 0 \qquad \text{if } s \in \mathcal{G} \\ &= \min_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s')\left[\mathcal{C}(s,a,s') + V^*(s')\right] \end{aligned}}$$

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

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

$$
\boxed{
\begin{aligned}
V^*(s) &= 0 \qquad \text{if } s \in \mathcal{G} \\
&= \min_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') \left[\mathcal{C}(s,a,s') + V^*(s')\right]
\end{aligned}
}
$$

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

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

$$\begin{aligned} V^*(s) &= 0 \qquad \text{if } s \in \mathcal{G} \\ &= \min_{a\in\mathcal{A}} \underbrace{\sum_{s'\in\mathcal{S}} \mathcal{T}(s,a,s')\left[\mathcal{C}(s,a,s') + V^*(s')\right]}_{Q^*(s,a)} \end{aligned}$$

$\mathbf{V^*(s)} = \min_{\mathbf{a}} \mathbf{Q^*(s,a)}$

Bellman Equations for MDP$_2$

- <$\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, $\mathcal{R}$, s$_{0,}$ γ>
- Define **V*(s)** {optimal **value**} as the **maximum** expected **discounted reward** from this state.
- V* should satisfy the following equation:

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

Fixed Point Computation in VI

$$\begin{tikzcd}
\boxed{V^*(s) = \min_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') \left[\mathcal{C}(s,a,s') + V^*(s') \right]} \arrow[d, "\text{iterative refinement}"] \\
\boxed{V_n(s) \leftarrow \min_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') \left[\mathcal{C}(s,a,s') + V_{n-1}(s') \right]}
\end{tikzcd}$$

Fixed Point Computation in VI

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

↓ *iterative refinement*

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

Fixed Point Computation in VI

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

$$\Big\downarrow \ \textit{iterative refinement}$$

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

Fixed Point Computation in VI

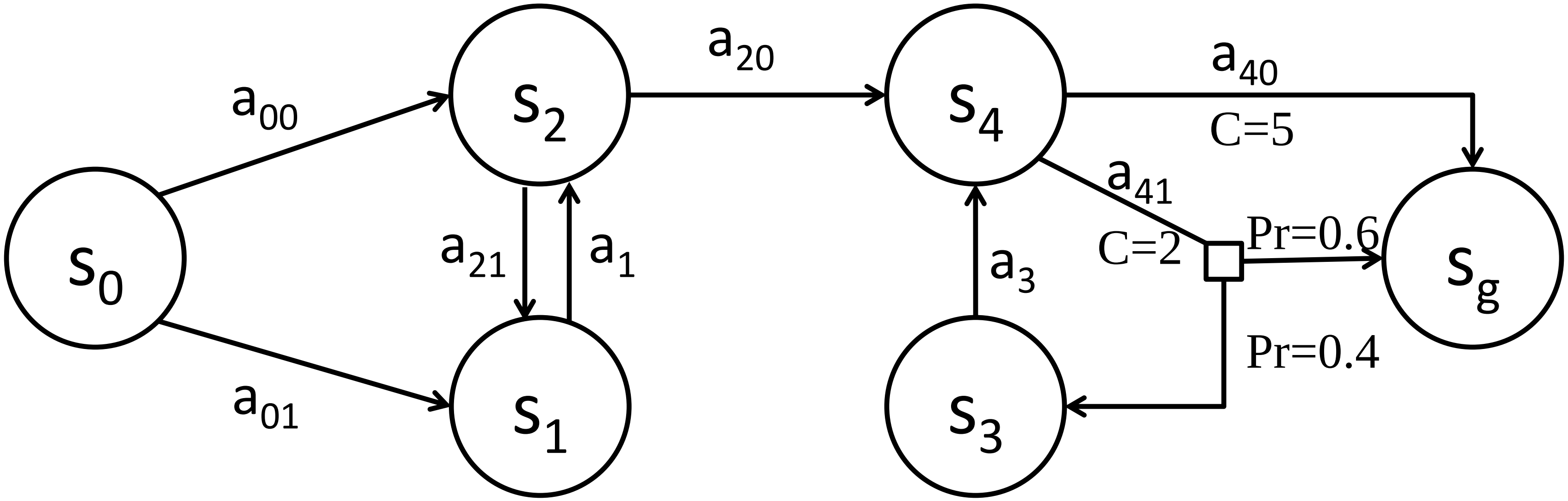
$$\boxed{V^*(s) = \min_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') \left[\mathcal{C}(s,a,s') + V^*(s')\right]}$$

iterative refinement

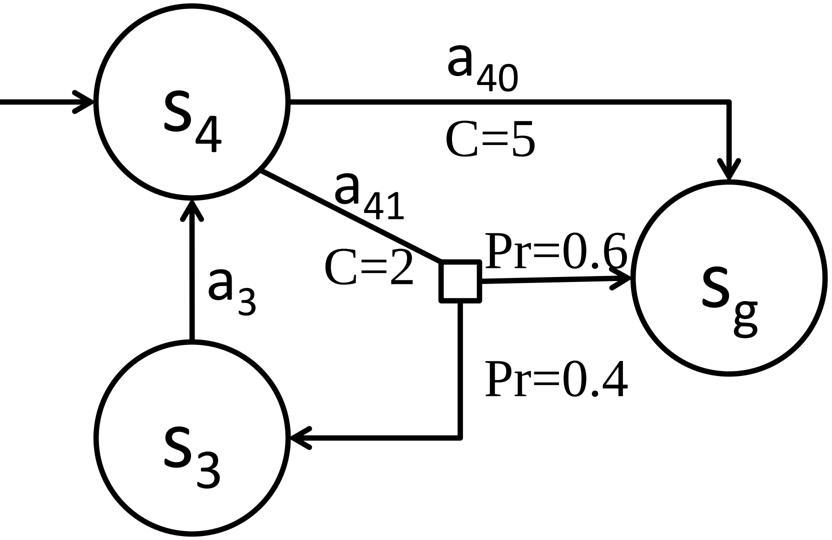
$$\boxed{V_n(s) \leftarrow \min_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') \left[\mathcal{C}(s,a,s') + V_{n-1}(s')\right]}$$

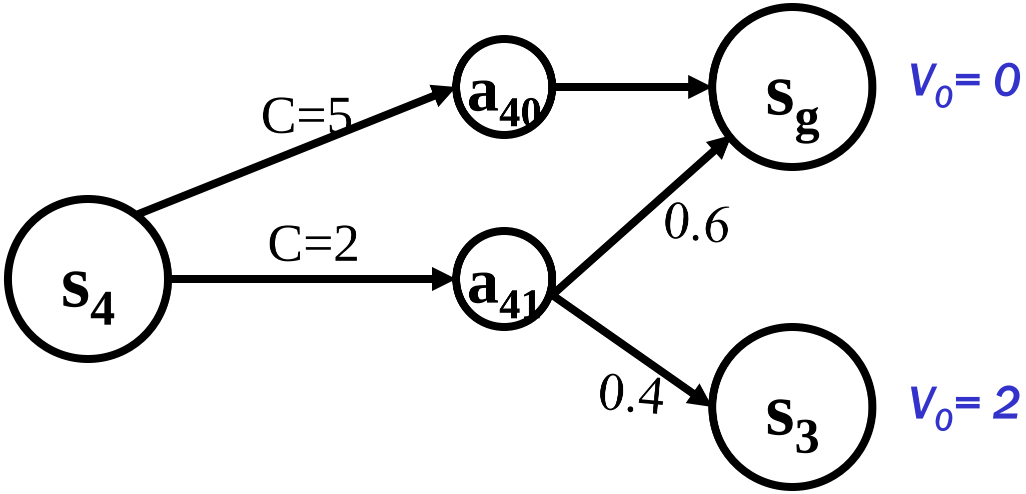
non-linear

Example

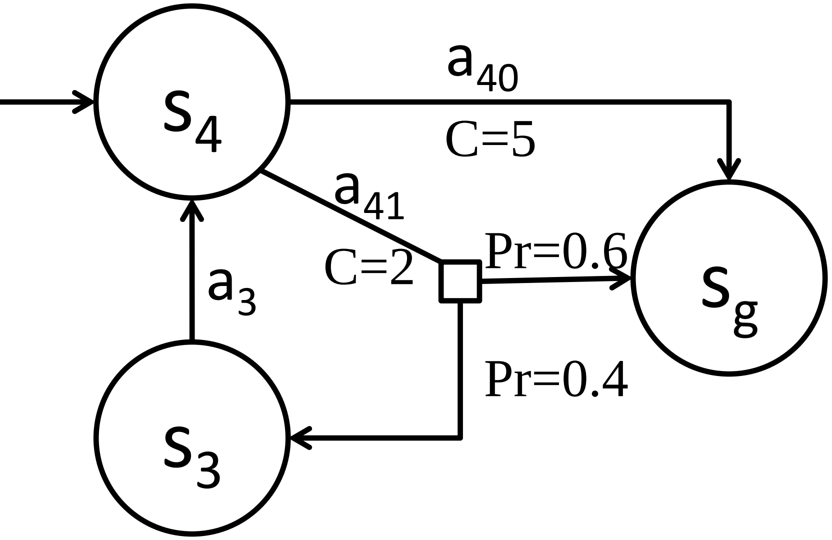


Bellman Backup

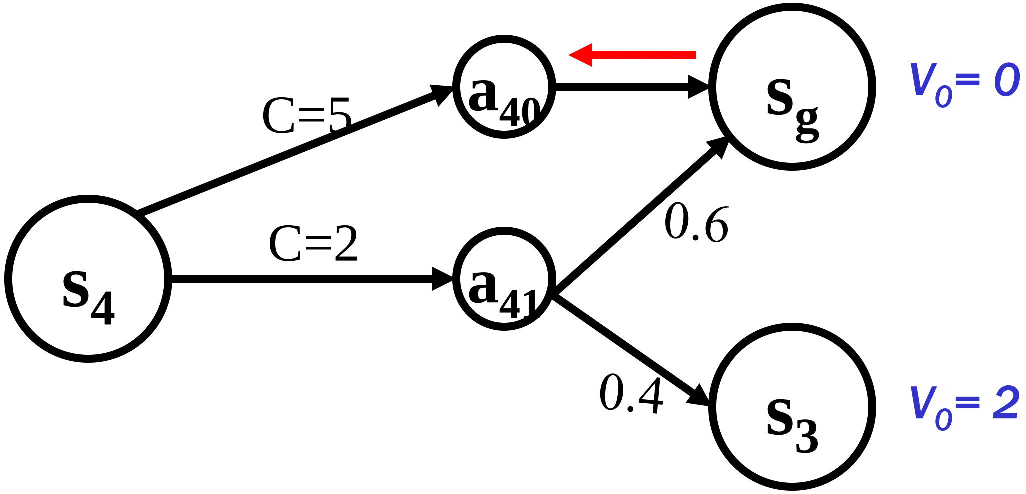




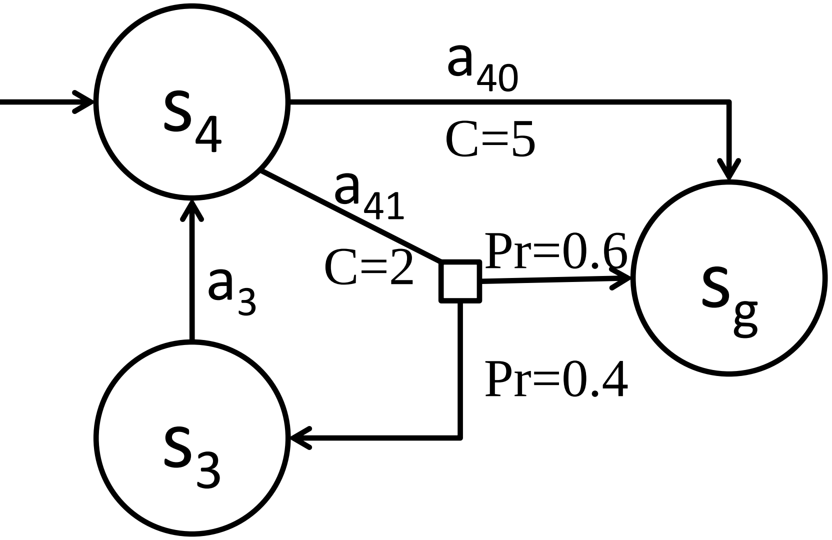
Bellman Backup



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



Bellman Backup

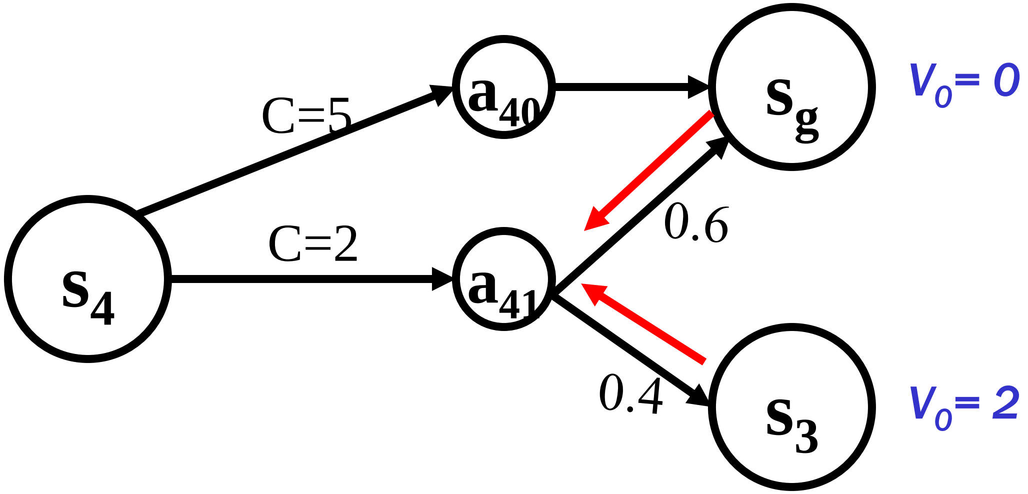


$Q_1(s_4,a_{40}) = 5 + 0$

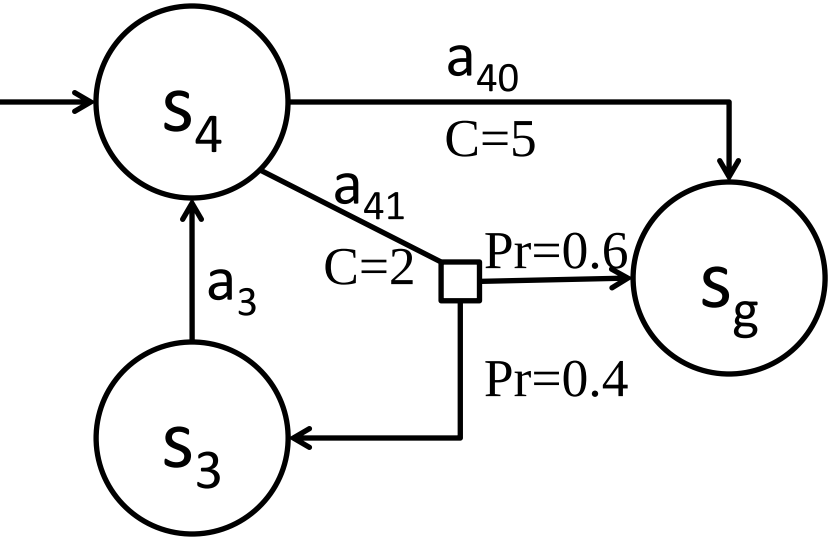
$Q_1(s_4,a_{41}) = 2 + 0.6\times 0$

$+ 0.4\times 2$

$= 2.8$

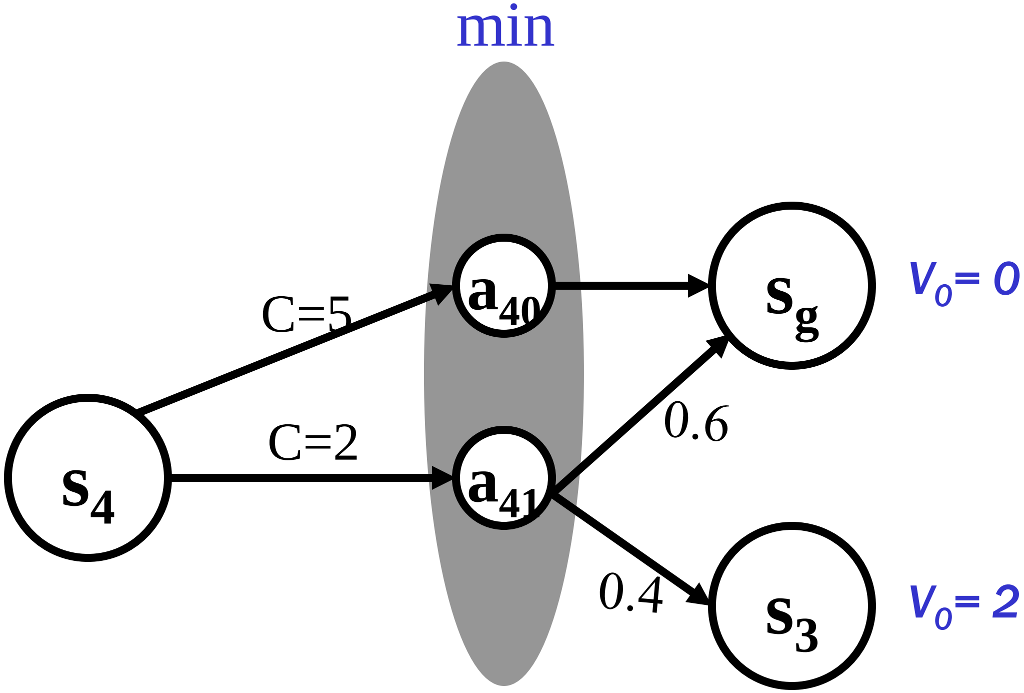


Bellman Backup

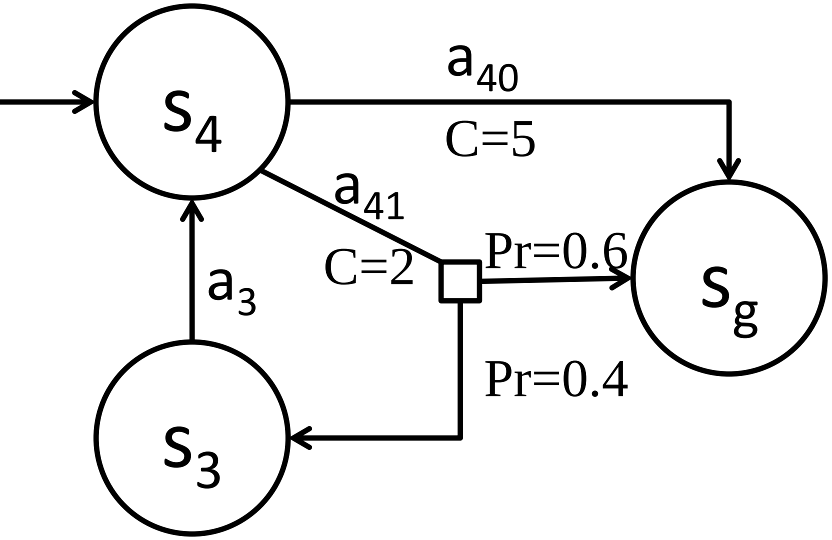


$$Q_1(s_4,a_{40}) = 5 + 0$$

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



Bellman Backup

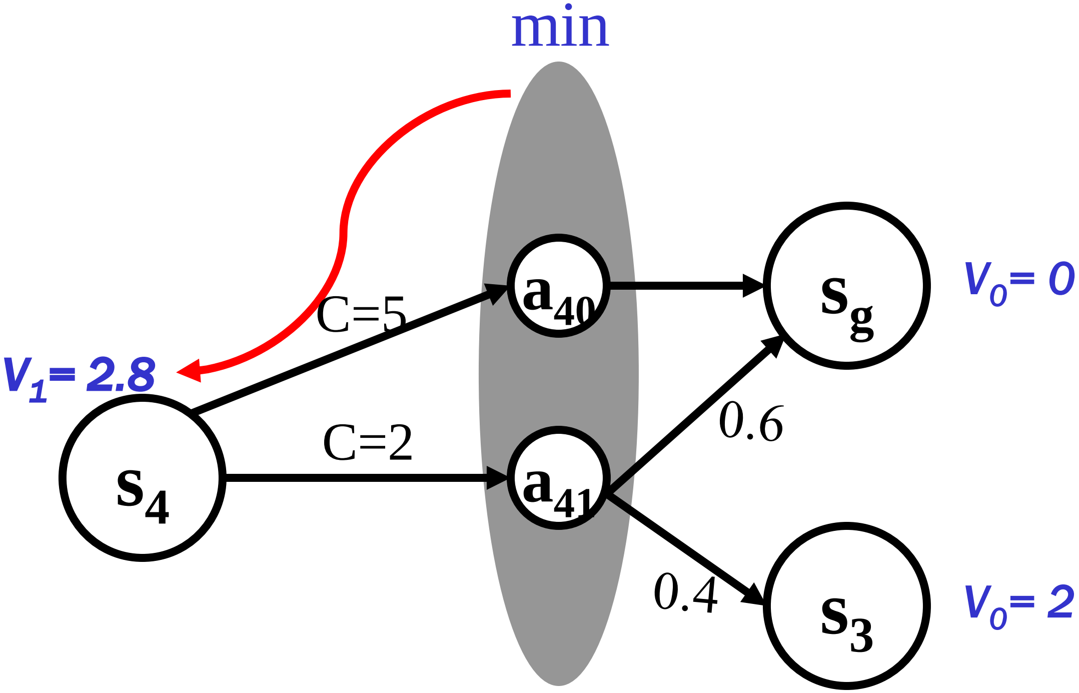


$Q_1(s_4,a_{40}) = 5 + 0$

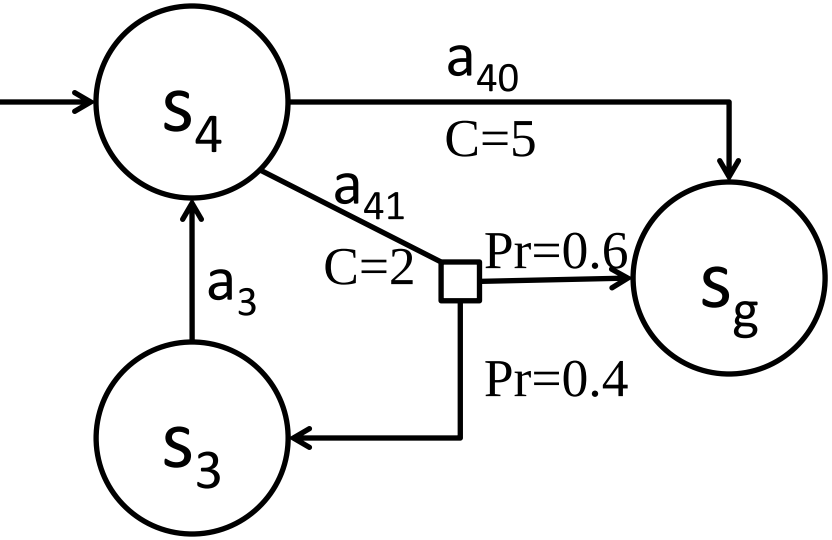
$Q_1(s_4,a_{41}) = 2 + 0.6 \times 0$

$+ 0.4 \times 2$

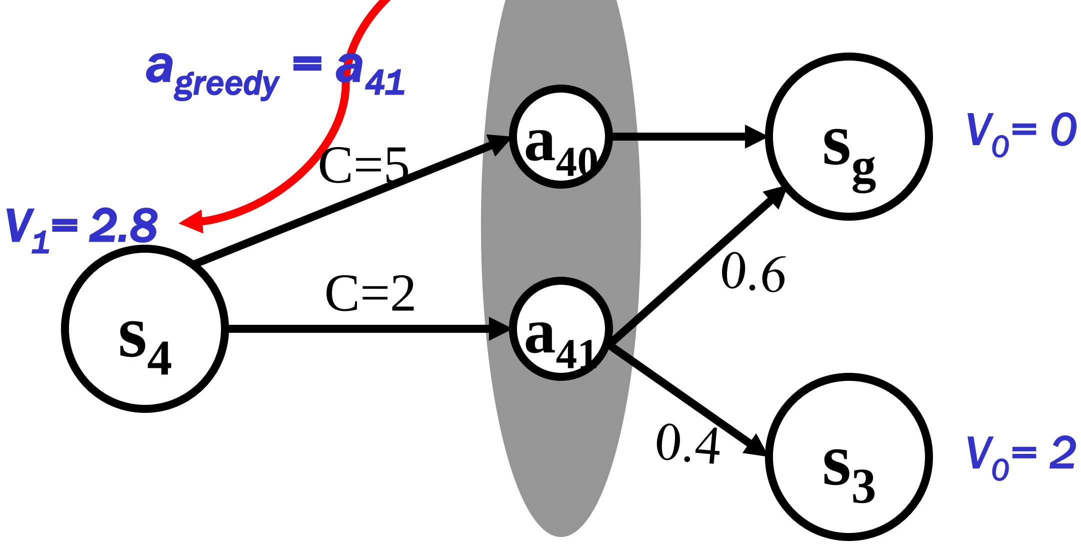
$= 2.8$



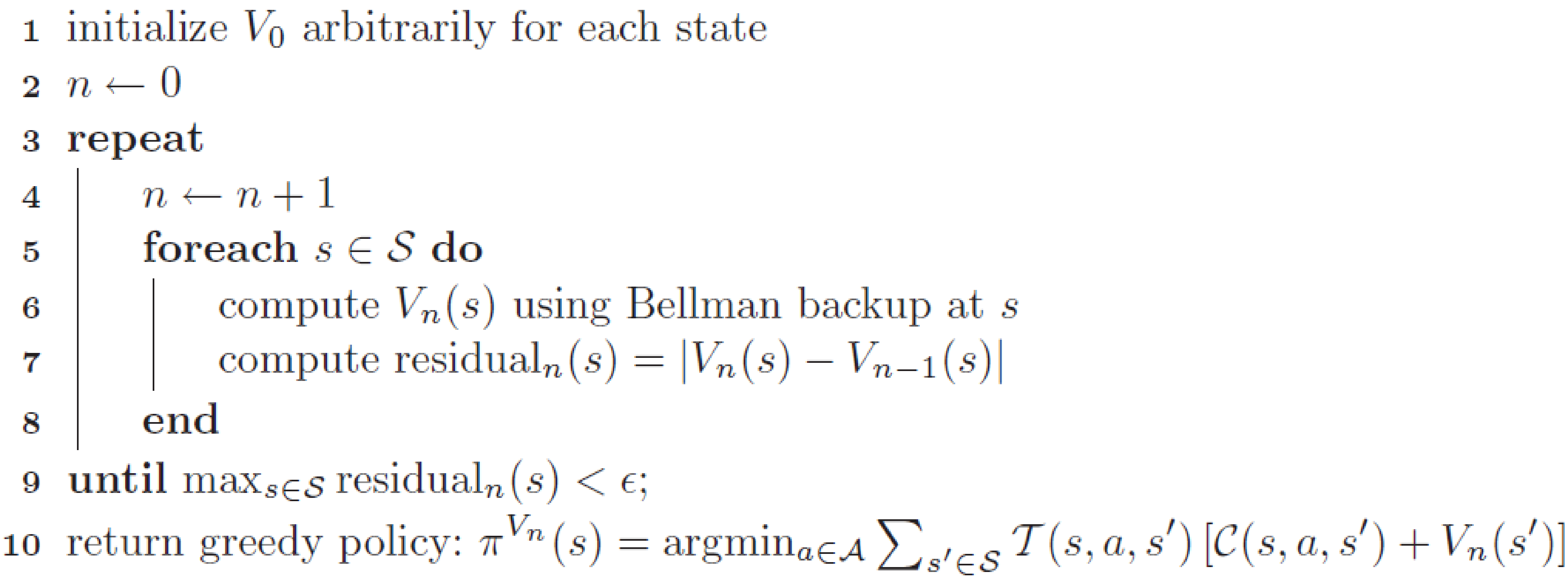
Bellman Backup



$$
\begin{aligned}
Q_1(s_4,a_{40}) &= 5 + 0\\
Q_1(s_4,a_{41}) &= 2 + 0.6\times 0\\
&\quad + 0.4\times 2\\
&= 2.8
\end{aligned}
$$



Value Iteration [Bellman 57]



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

Value Iteration [Bellman 57]

No restriction on initial value function

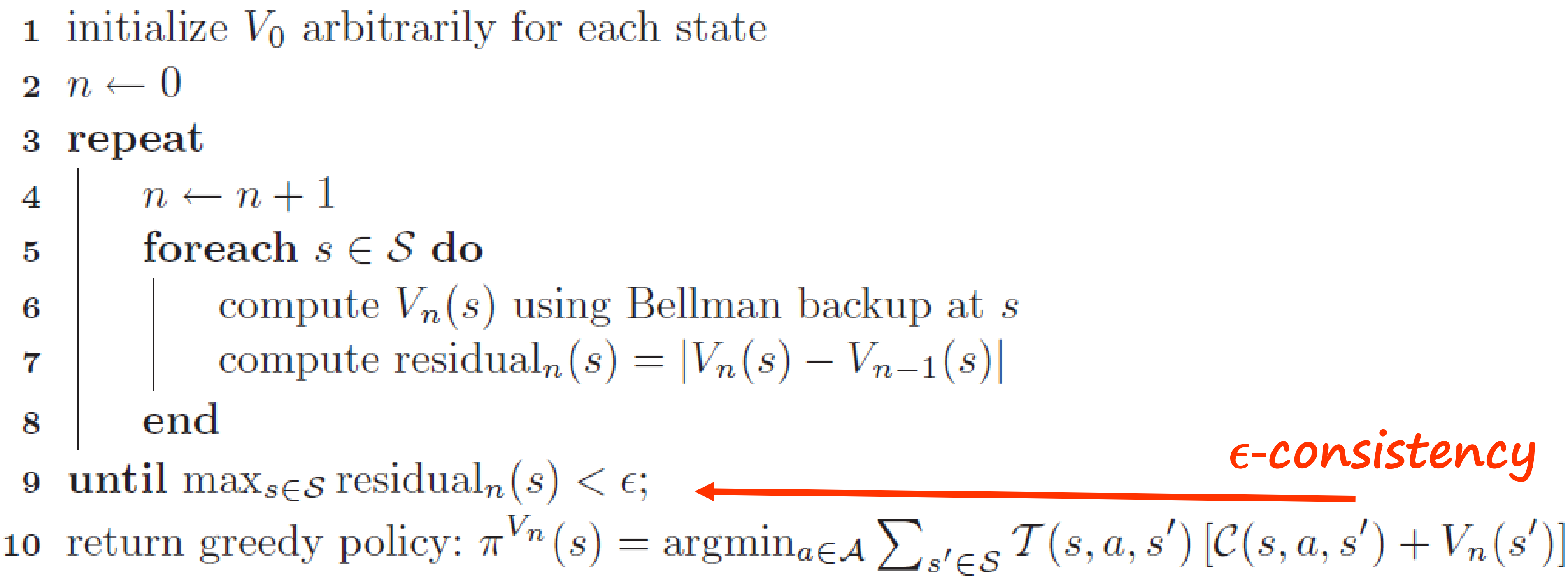
```
initialize $V_0$ arbitrarily for each state
$n \leftarrow 0$
repeat
    $n \leftarrow n + 1$
    foreach $s \in \mathcal{S}$ do
        compute $V_n(s)$ using Bellman backup at $s$
        compute residual$_n(s) = |V_n(s) - V_{n-1}(s)|$
    end
until $\max_{s \in \mathcal{S}}$ residual$_n(s) < \epsilon$;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

Value Iteration [Bellman 57]

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

iteration n

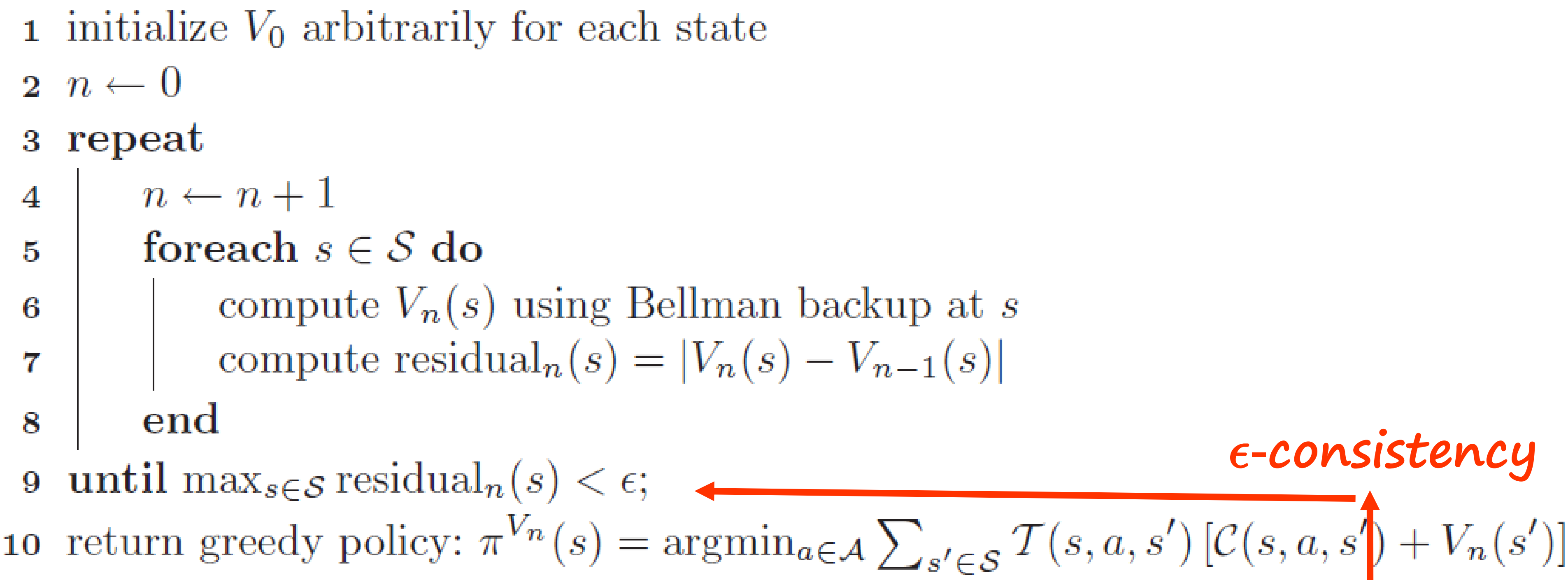
Value Iteration [Bellman 57]



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

ϵ-consistency

Value Iteration [Bellman 57]



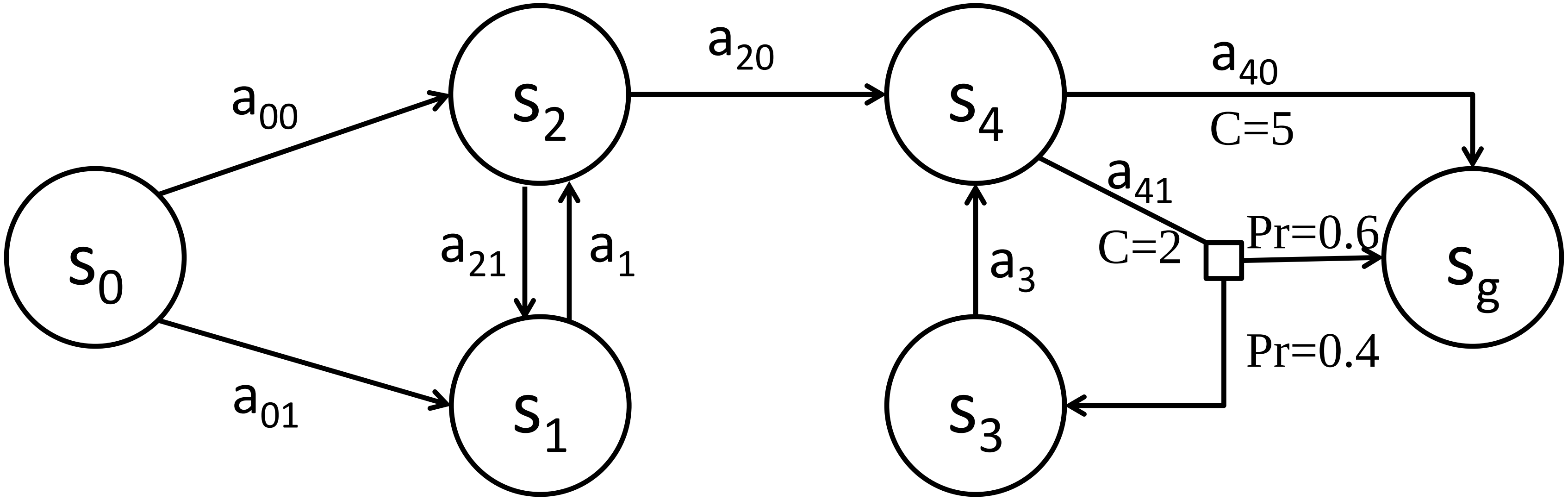
```
initialize $V_0$ arbitrarily for each state
$n \leftarrow 0$
**repeat**
    $n \leftarrow n + 1$
    **foreach** $s \in \mathcal{S}$ **do**
        compute $V_n(s)$ using Bellman backup at $s$
        compute residual$_n(s) = |V_n(s) - V_{n-1}(s)|$
    **end**
**until** $\max_{s \in \mathcal{S}}$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

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: $\text{poly}(|\mathcal{S}|, |\mathcal{A}|, 1/(1-\gamma))$
- 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)

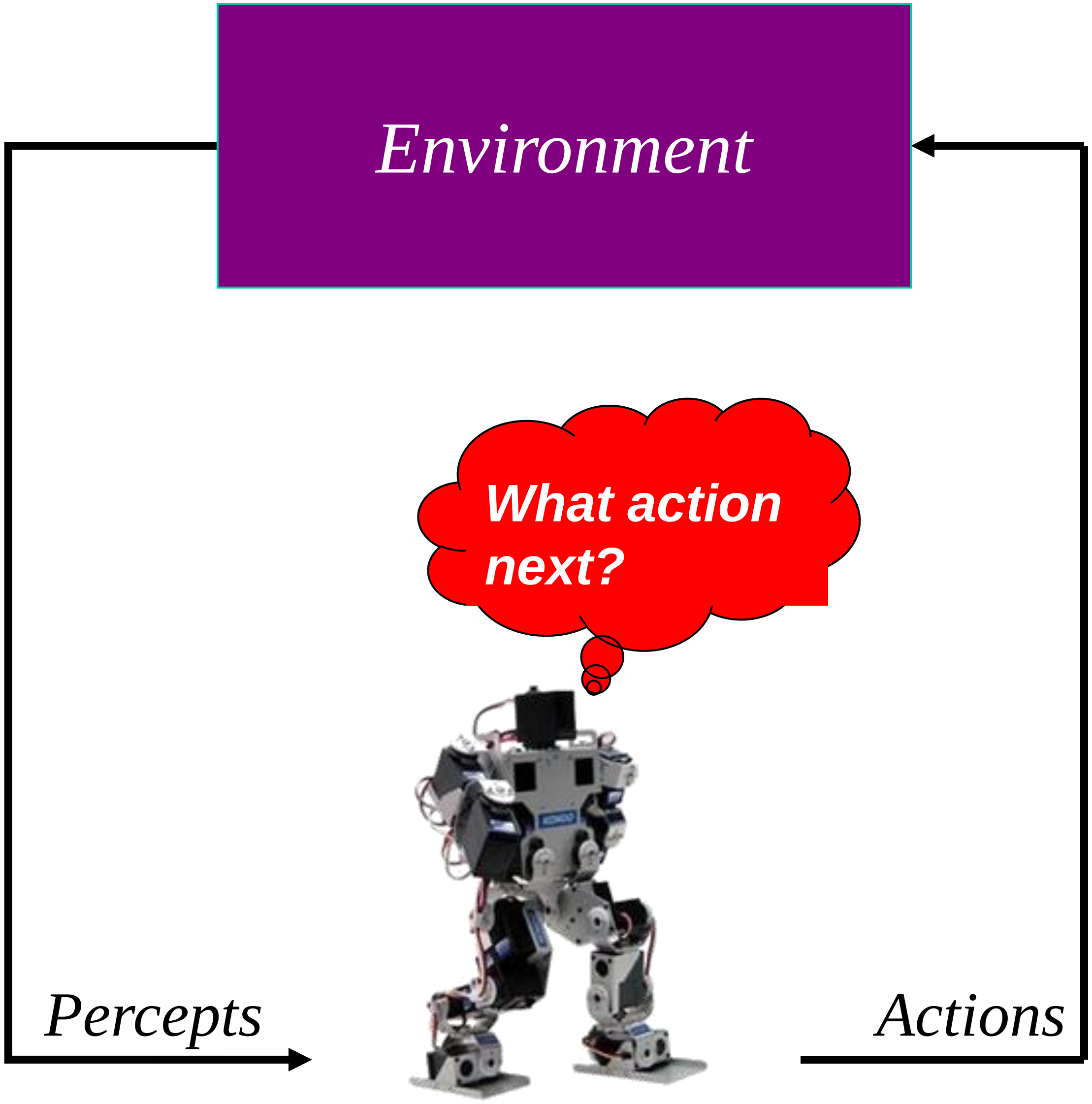
$V_k \geq_p V^* \Rightarrow V_n \geq_p V^*$ (V_n monotonic from above)

Extensions

- Heuristic Search + Dynamic Programming
  - AO*, LAO*, RTDP, …
- Factored MDPs
  - add planning graph style heuristics
  - use goal regression to generalize better
- Hierarchical MDPs
  - hierarchy of sub-tasks, actions to scale better
- Reinforcement Learning
  - learning the probability and rewards
  - acting while learning – connections to psychology
- Partially Observable Markov Decision Processes
  - noisy sensors; partially observable environment
  - popular in robotics

Partially Observable MDPs

Static



Environment

Partially Observable

What action next?

Stochastic

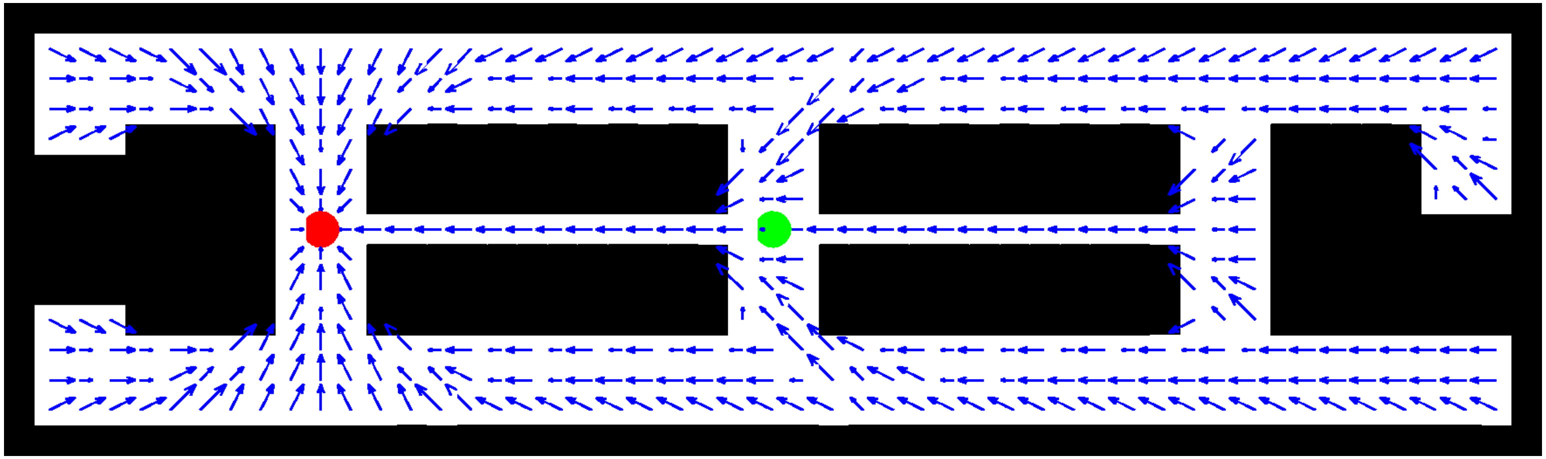
Instantaneous

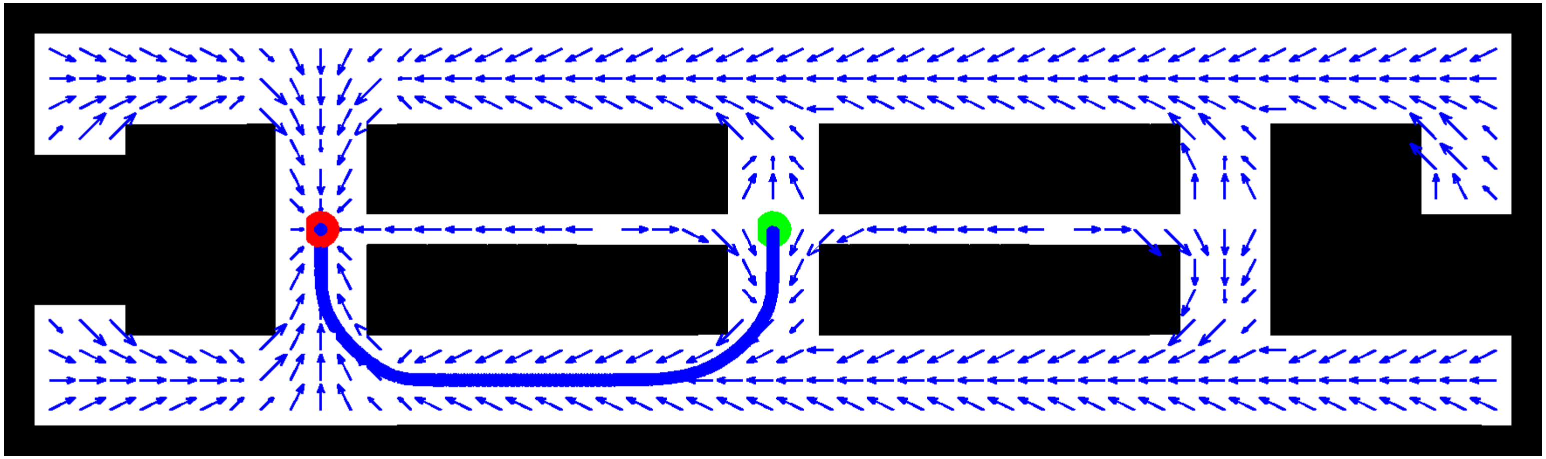
Noisy

Percepts

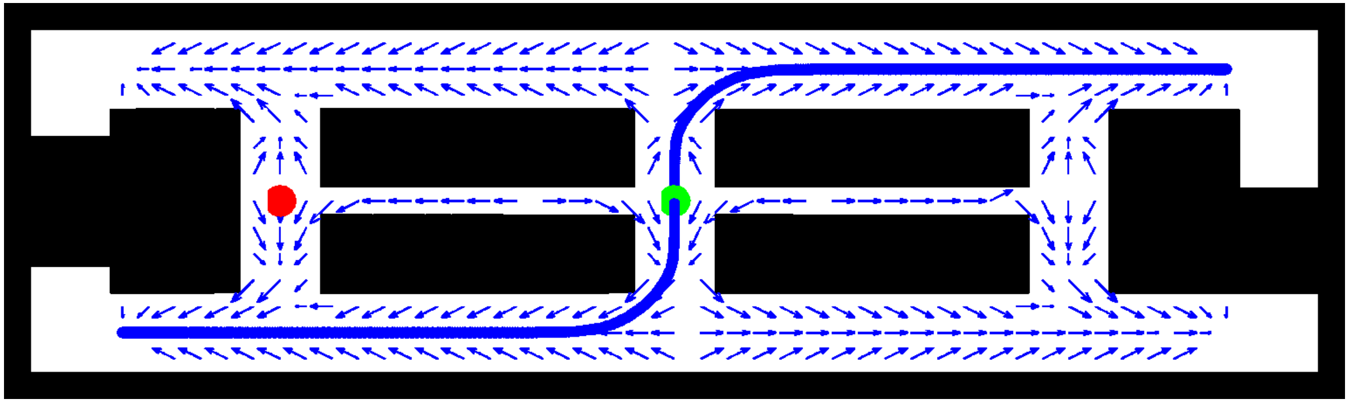
Actions

Stochastic, Fully Observable

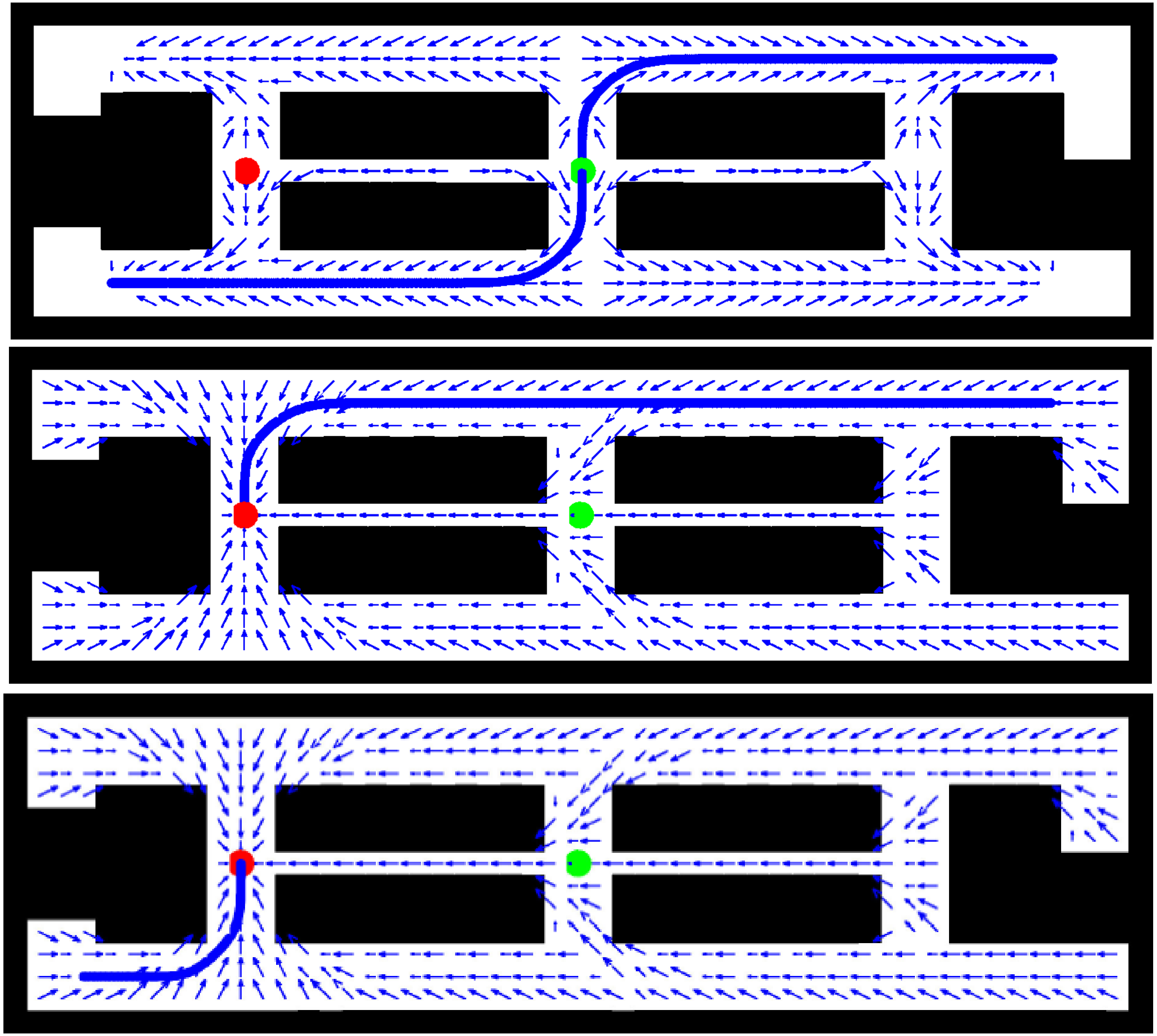




Stochastic, Partially Observable



Stochastic, Partially Observable



POMDPs

- In POMDPs we apply the very same idea as in MDPs.

- Since the state is not observable,

  the agent has to make its decisions based on the belief state which is a posterior distribution over states.

- Let b be the belief of the agent about the current state
- POMDPs compute a value function over belief space:

$$V_T(b) \quad = \quad \max_a \left[r(b,a) + \gamma \int V_{T-1}(b') p(b' \mid b,a)\, db' \right]$$

POMDPs

- Each belief is a probability distribution,
  - value fn is a function of an entire probability distribution.
- Problematic, since probability distributions are continuous.
- Also, we have to deal with huge complexity of belief spaces.

- For finite worlds with finite state, action, and observation spaces and finite horizons,
  - we can represent the value functions by piecewise linear functions.

Applications

- Robotic control
  - helicopter maneuvering, autonomous vehicles
  - Mars rover - path planning, oversubscription planning
  - elevator planning
- Game playing - backgammon, tetris, checkers
- Neuroscience
- Computational Finance, Sequential Auctions
- Assisting elderly in simple tasks
- Spoken dialog management
- Communication Networks – switching, routing, flow control
- War planning, evacuation planning