16-782
Planning & Decision-making in Robotics

Planning under Uncertainty: Expected Formulation, Solving MDPs

Maxim Likhachev

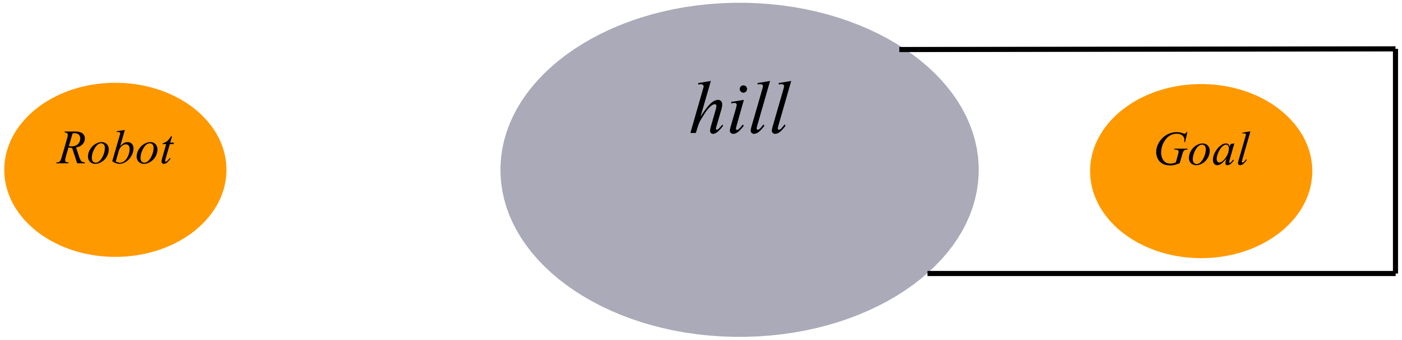
Robotics Institute

Carnegie Mellon University

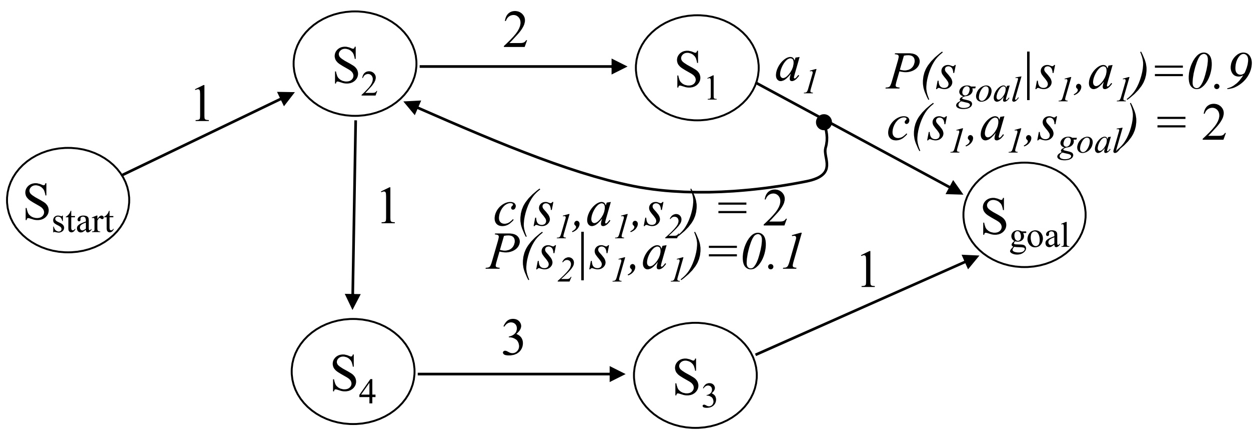
Minimax Formulation is Often Too Conservative

Example:

moving over the hill has 10% chance of slipping



Expected Cost Formulation



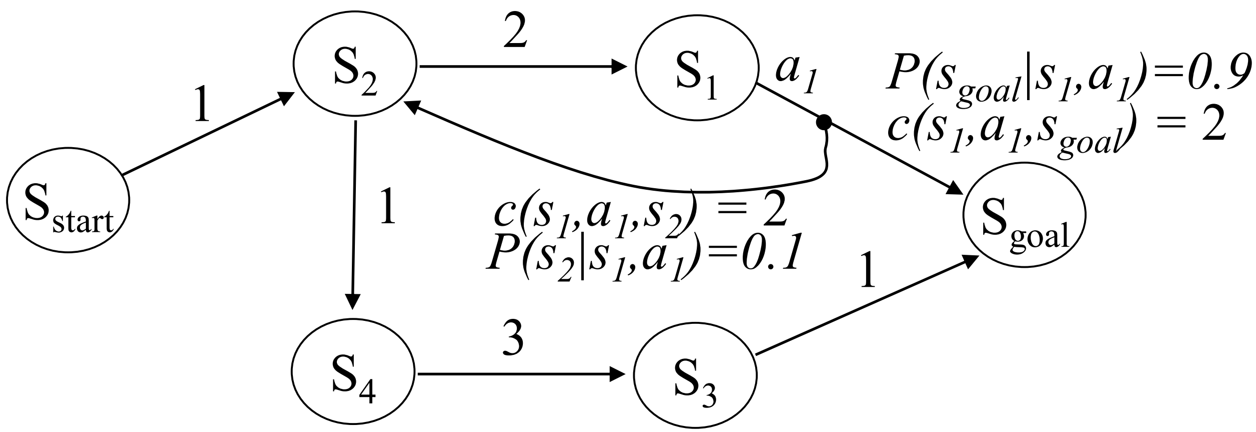
- Optimal policy π^*:

  minimizes the *expected* cost-to-goal

  $\pi^* = argmin_\pi E\{cost\text{-}to\text{-}goal\}$

expectation over outcomes

Expected Cost Formulation



- Optimal policy π^*:
  minimizes the *expected* cost-to-goal
  $\pi^* = argmin_\pi\ E\{cost\text{-}to\text{-}goal\}$

expectation over outcomes

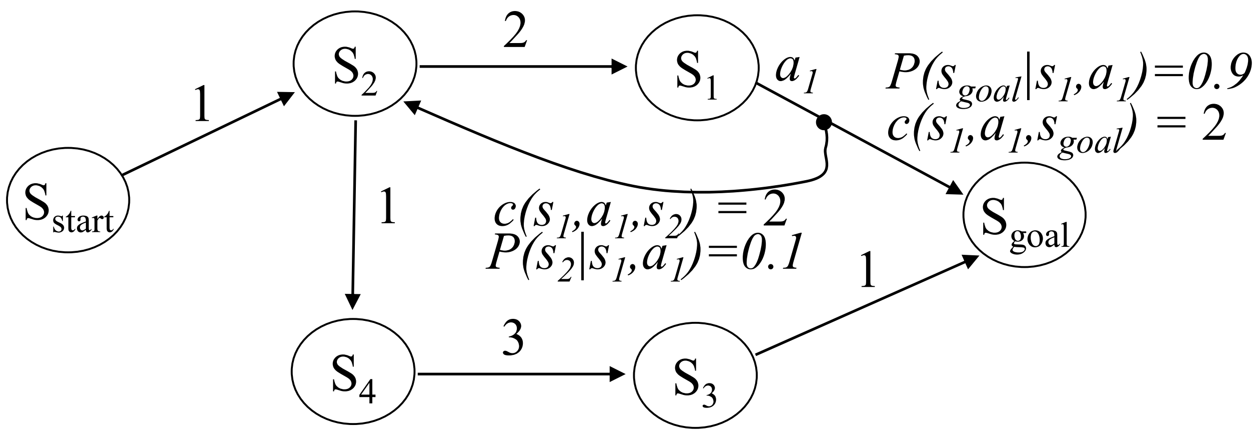
- expected cost-to-goal for π_1=(go through s_4) is

  1+1+3+1=6

- cost-to-goal for π_2=(try to go through s_1) is:

0.9*(1+2+2) + 0.9*0.1*(1+2+2+2+2) + 0.9*0.1*0.1*(1+2+2+2+2+2+2) + …=$5.44\overline{4}$

Expected Cost Formulation



- Optimal policy π*:
  minimizes the *expected* cost-to-goal

Given a policy, its value can be computed by solving a system of linear equations

expectation over outcome

How to compute it?

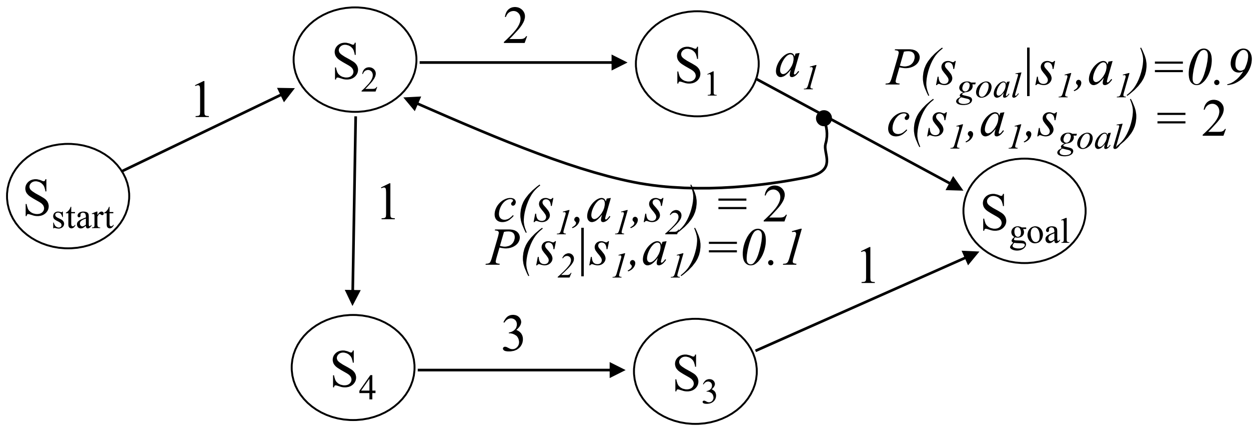
- expected cost-to-goal for π_1=(go through s_4

  1+1+3+1=6

- cost-to-goal for π_2=(try to go through s_1) is:

$0.9*(1+2+2) + 0.9*0.1*(1+2+2+2+2) + 0.9*0.1*0.1*(1+2+2+2+2+2+2) + \ldots = 5.44\overline{4}$

Expected Cost Formulation



- Optimal policy π^*:
  minimizes the *expected* cost-to-goal
  π^*

<u>Given</u> a policy, its value can be computed by solving a system of linear equations

expectati

- expected cost-to-goal fo

  1+1+3+1=6

- cost-to-goal for π_2=(try to g

0.9*(1+2+2) + 0.9*0.1*(1+2+2+2+2) + 0.9*0.1 ... 2) + …=5.44$\overline{4}$

for π_2:

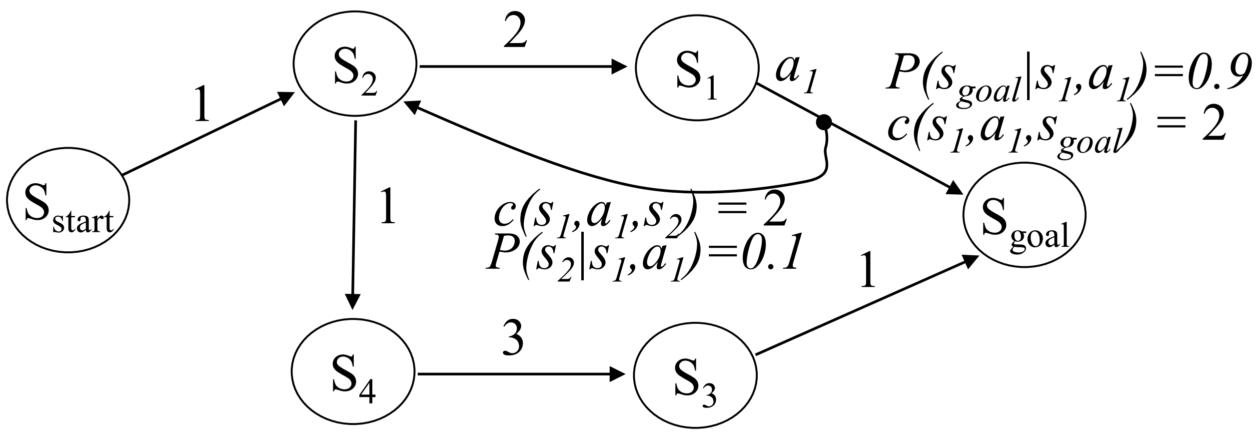
$v(s_{start}) = 1+v(s_2)$

$v(s_2) = 2+v(s_1)$

$v(s_1) = 0.9*(2+v(s_{goal}))+0.1*(2+v(s_2))$

$v(s_{goal}) = 0$

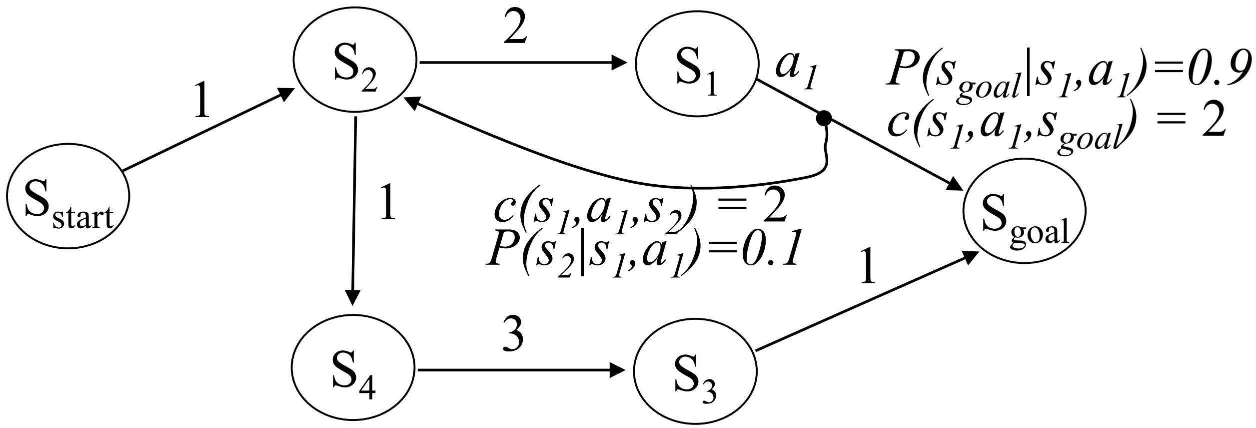
Expected Cost Formulation



- Optimal policy π^*:
  minimizes the *expected* cost-to-goal
  $\pi^* = argmin_\pi\, E\{cost\text{-}to\text{-}goal\}$

- Optimal expected cost policy $\pi^* = \pi_2 = (go\ through\ s_1)$

Expected Cost Formulation

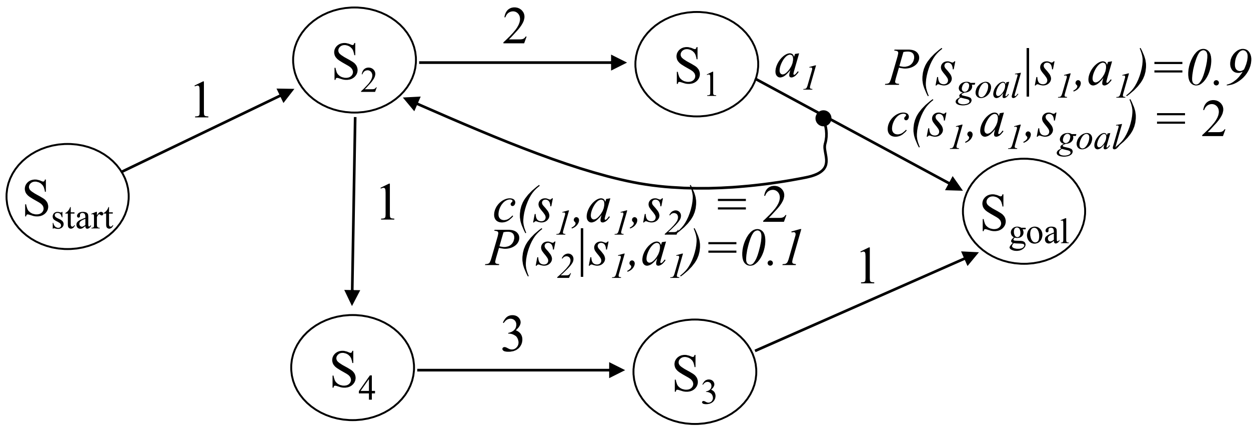


- Optimal policy π^*:
  minimizes the *expected* cost-to-goal
  $\pi^* = argmin_\pi\, E\{cost\text{-}to\text{-}goal\}$

- Optimal expected cost policy $\pi^* = \pi_2$=*(go through* s_1*)*

In contrast,
optimal policy for minimax formulation
was π_1=(go through s_4)

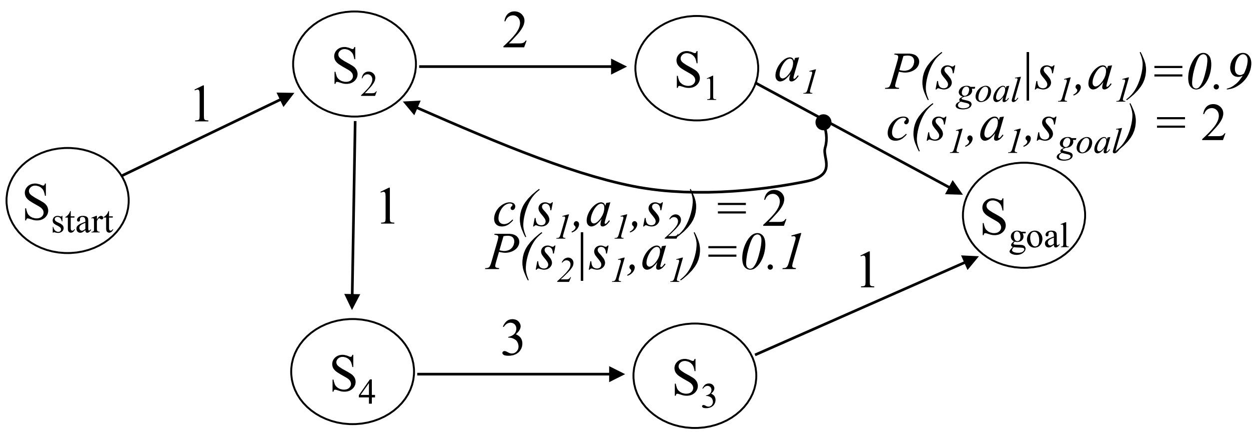
Computing Expected Cost Minimal Plans



- Optimal policy π^*:
  minimizes the *expected* cost-to-goal
  $\pi^* = argmin_\pi E\{cost\text{-}to\text{-}goal\}$

- Let $v^*(s)$ be minimal expected cost-to-goal for state s

Computing Expected Cost Minimal Plans



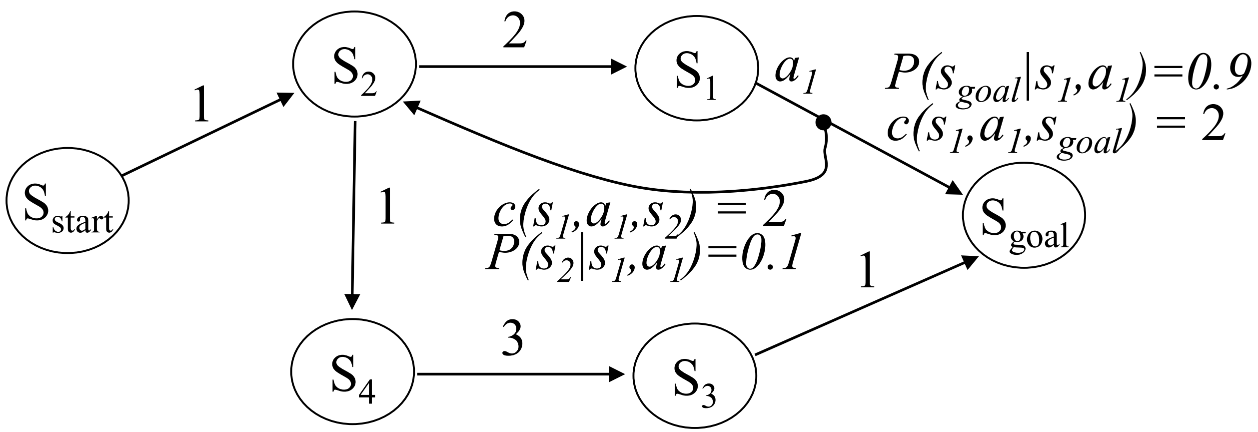
- Optimal policy π^*:

  $\pi^*(s) = \text{argmin}_a\, E\{c(s,a,s')+v^*(s')\}$

  (expectation over outcomes s' of action a executed at state s)

Why?

Computing Expected Cost Minimal Plans



- Optimal expected cost-to-goal values v^* satisfy:

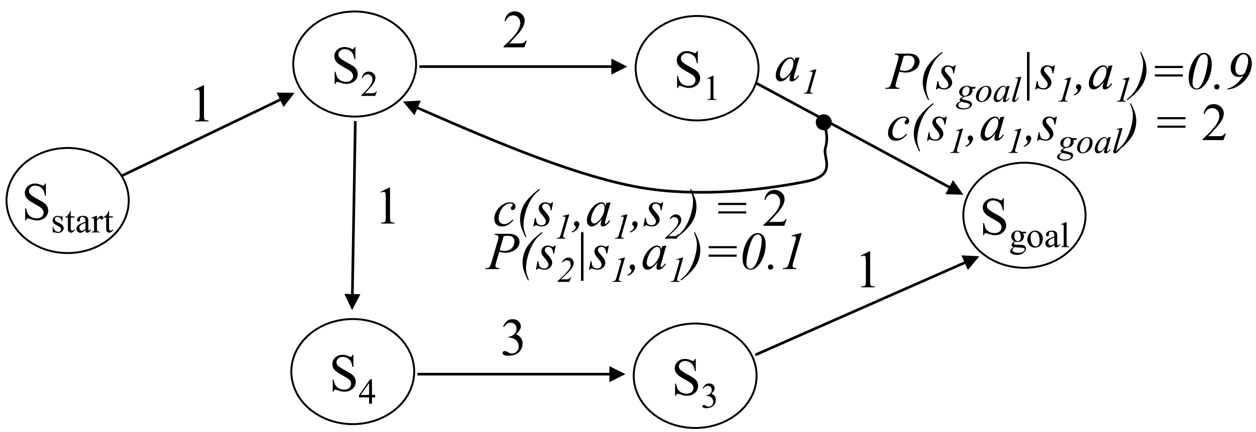
  $v^*(s_{goal})=0$

  $v^*(s) = \min_a E\{c(s,a,s')+v^*(s')\}$ *for all* $s \neq s_{goal}$

  (expectation over outcomes s' of action a executed at state s)

Bellman optimality equation

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

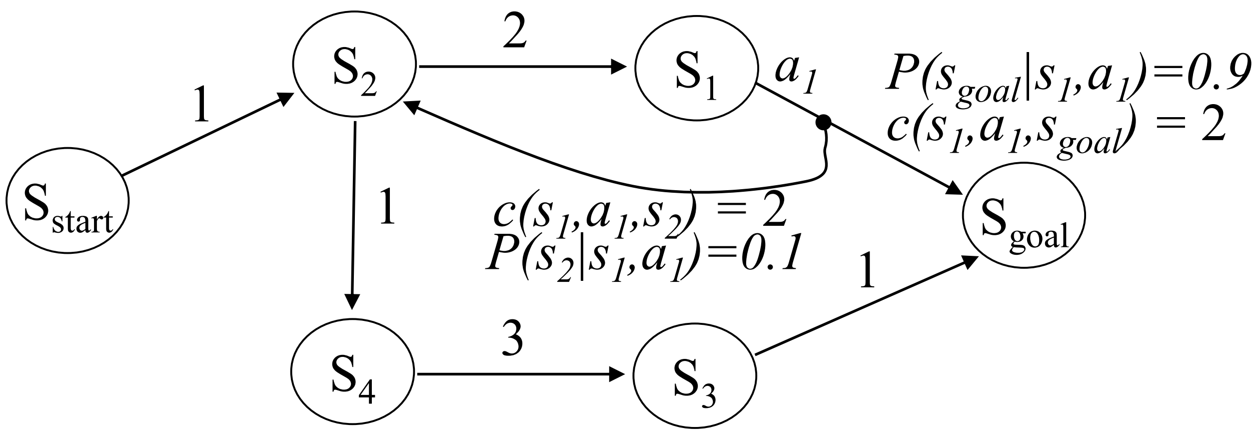
  Initialize v-values of all states to finite values;

  Iterate over all s in MDP and re-compute until convergence:

  $v(s_{goal}) = 0$

  $v(s) = \min_a E\{c(s,a,s')+v(s')\}$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

  Initialize *v*-values of all states to finite values;

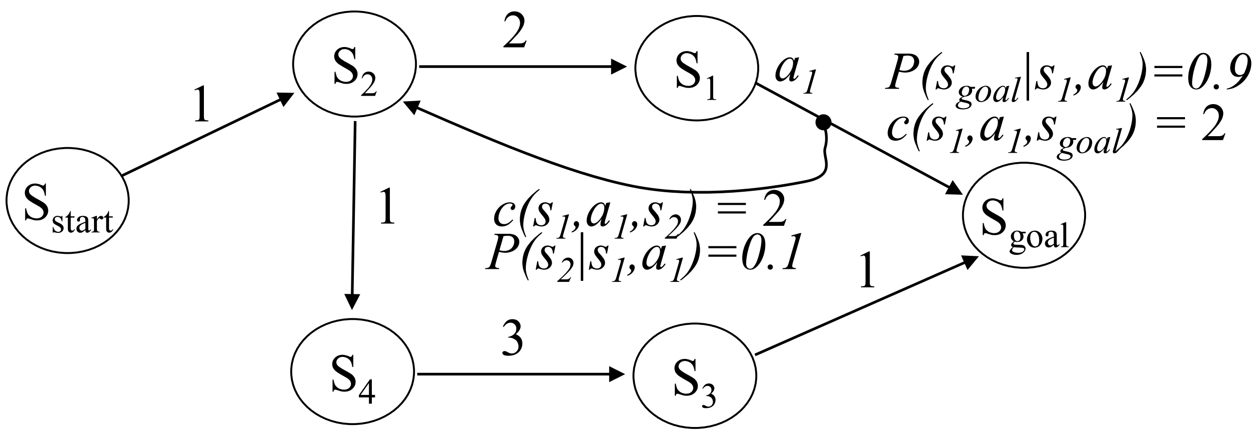
  Iterate over all *s* in MDP and re-compute until convergence:

  $$v(s_{goal}) = 0$$

  $$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

Bellman update equation (or backup)

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

best to initialize to admissible values (under-estimates of the actual costs-to-goal)

Initialize *v*-values of all states to finite values;

Iterate over all *s* in MDP and re-compute until convergence:

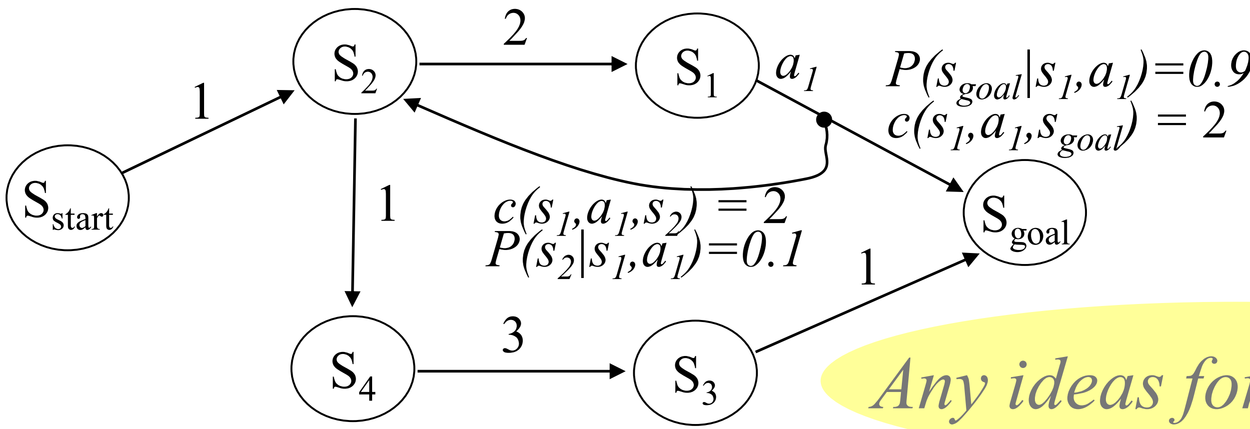
$$v(s_{goal}) = 0$$

$$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

converges to an optimal value function ($v(s)=v^*(s)$ for all s) for any iteration order

the speed of convergence depends on iteration order

Computing Expected Cost Minimal Plans



Any ideas for the order?

best to initialize to admissible values (under-estimates of the actual costs-to-goal)

- Value Iteration (VI):

  Initialize *v*-values of all states to finite values;

  Iterate over all *s* in MDP and re-compute until convergence:

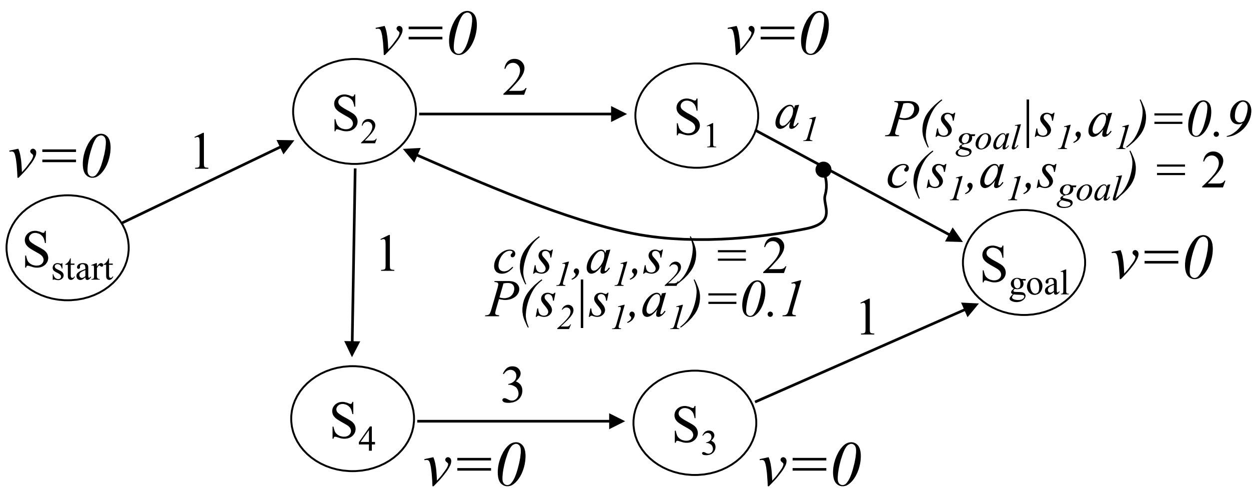
  $$v(s_{goal}) = 0$$

  $$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

converges to an optimal value function ($v(s)=v^*(s)$ for all s) for any iteration order

the speed of convergence depends on iteration order

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

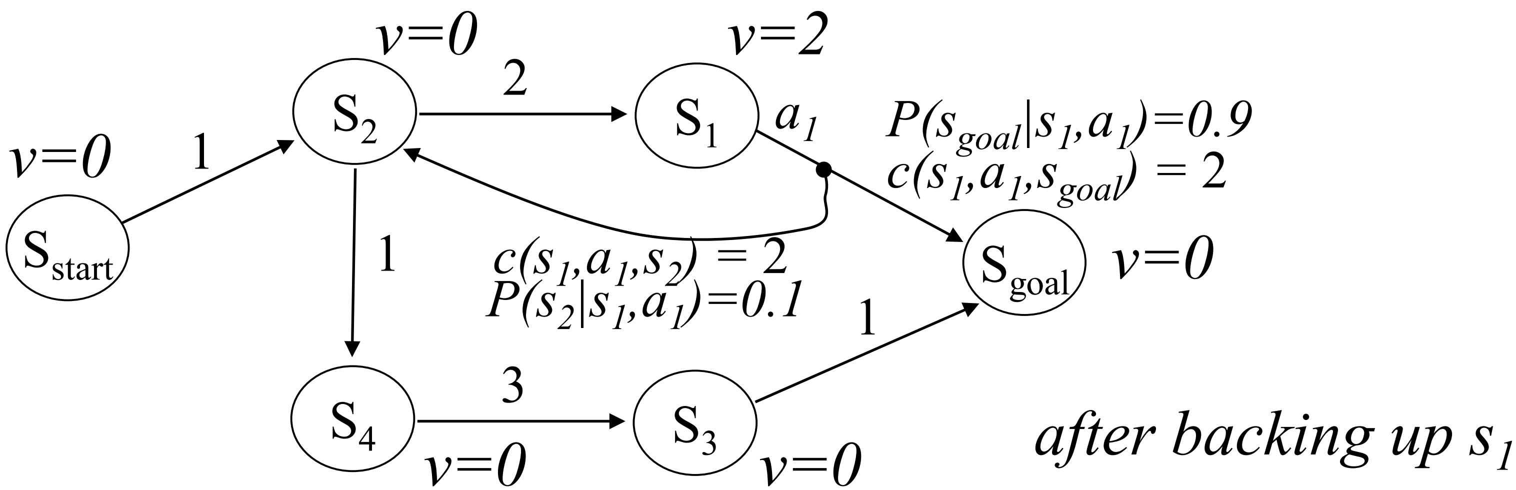
  Initialize *v*-values of all states to finite values;

  Iterate over all *s* in MDP and re-compute until convergence:

  $v(s_{goal}) = 0$

  $v(s) = \min_a E\{c(s,a,s')+v(s')\}$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

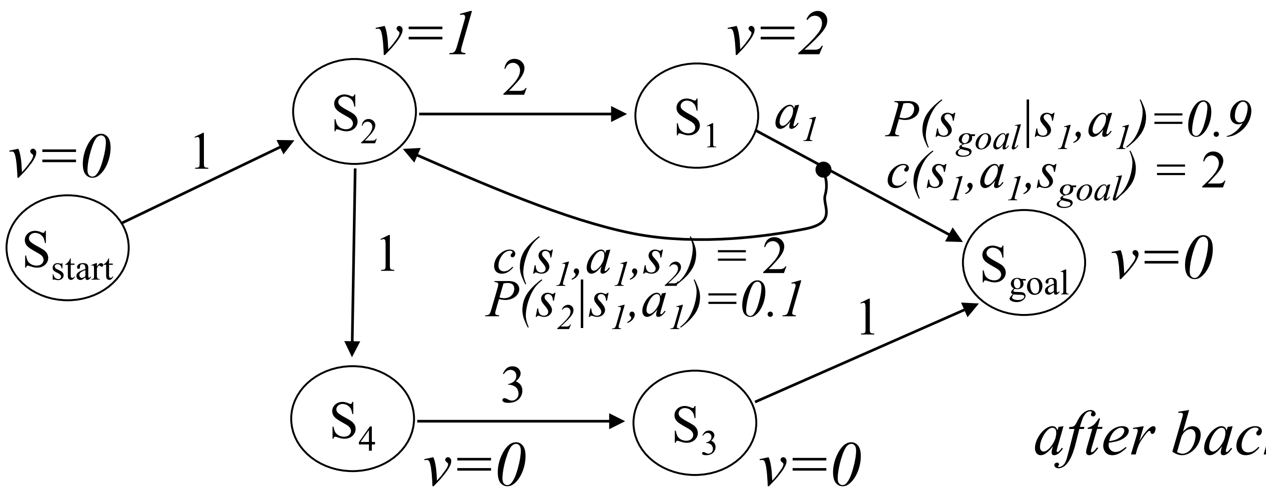
  Initialize *v*-values of all states to finite values;

  Iterate over all *s* in MDP and re-compute until convergence:

  $v(s_{goal}) = 0$

  $v(s) = min_a E\{c(s,a,s')+v(s')\}$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



after backing up s_2

- Value Iteration (VI):

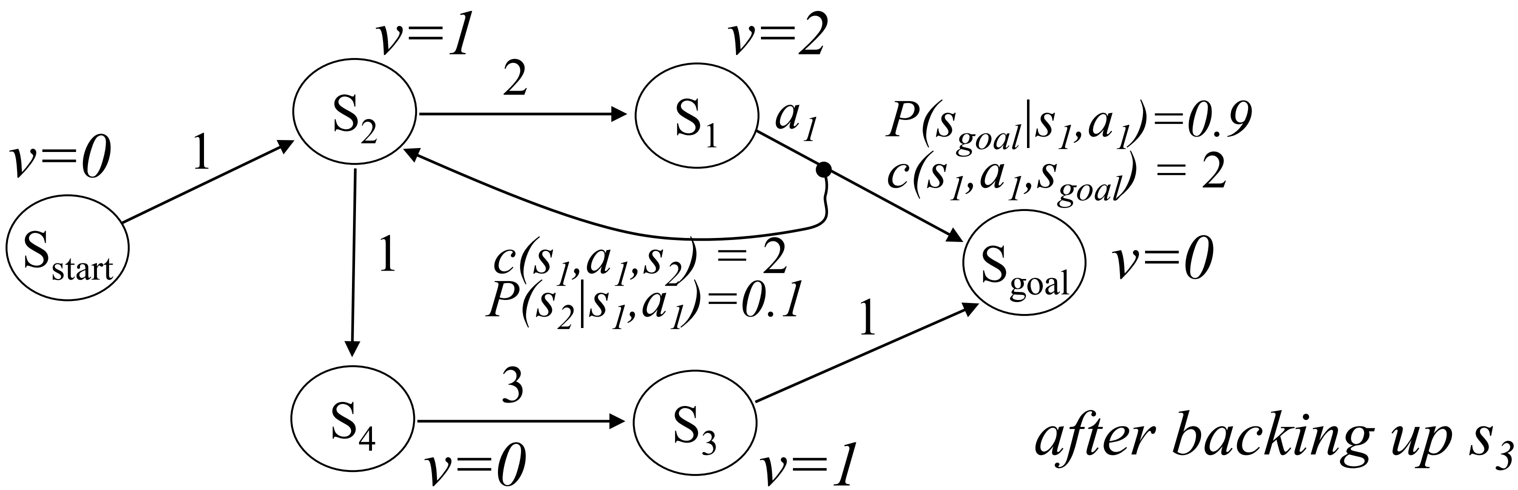
  Initialize *v*-values of all states to finite values;

  Iterate over all *s* in MDP and re-compute until convergence:

  $v(s_{goal}) = 0$

  $v(s) = min_a\ E\{c(s,a,s')+v(s')\}$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

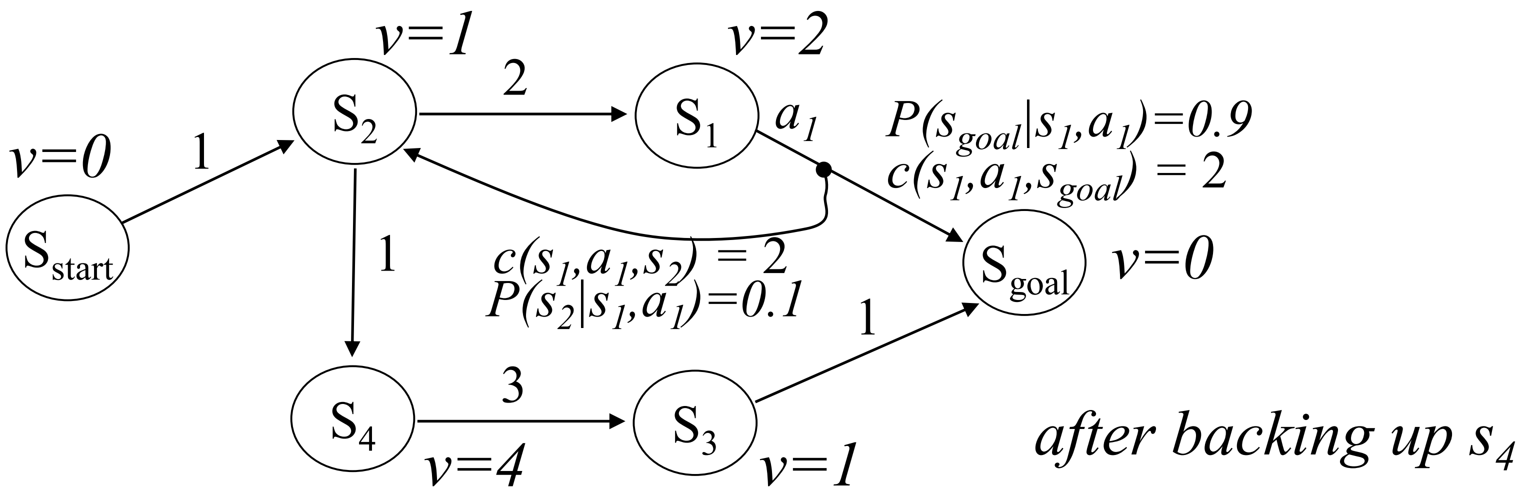
  Initialize *v*-values of all states to finite values;

  Iterate over all *s* in MDP and re-compute until convergence:

  $v(s_{goal}) = 0$

  $v(s) = \min_a E\{c(s,a,s')+v(s')\}$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

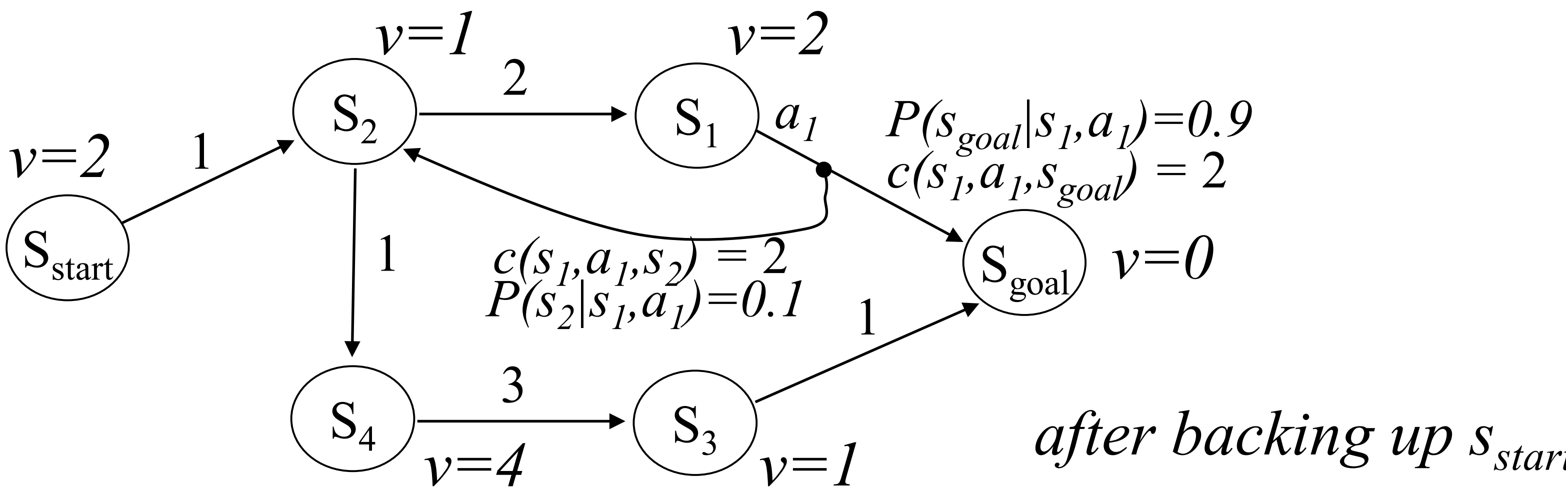
  Initialize *v*-values of all states to finite values;

  Iterate over all *s* in MDP and re-compute until convergence:

  $v(s_{goal}) = 0$

  $v(s) = min_a\, E\{c(s,a,s')+v(s')\}$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

Initialize v-values of all states to finite values;

Iterate over all s in MDP and re-compute until convergence:

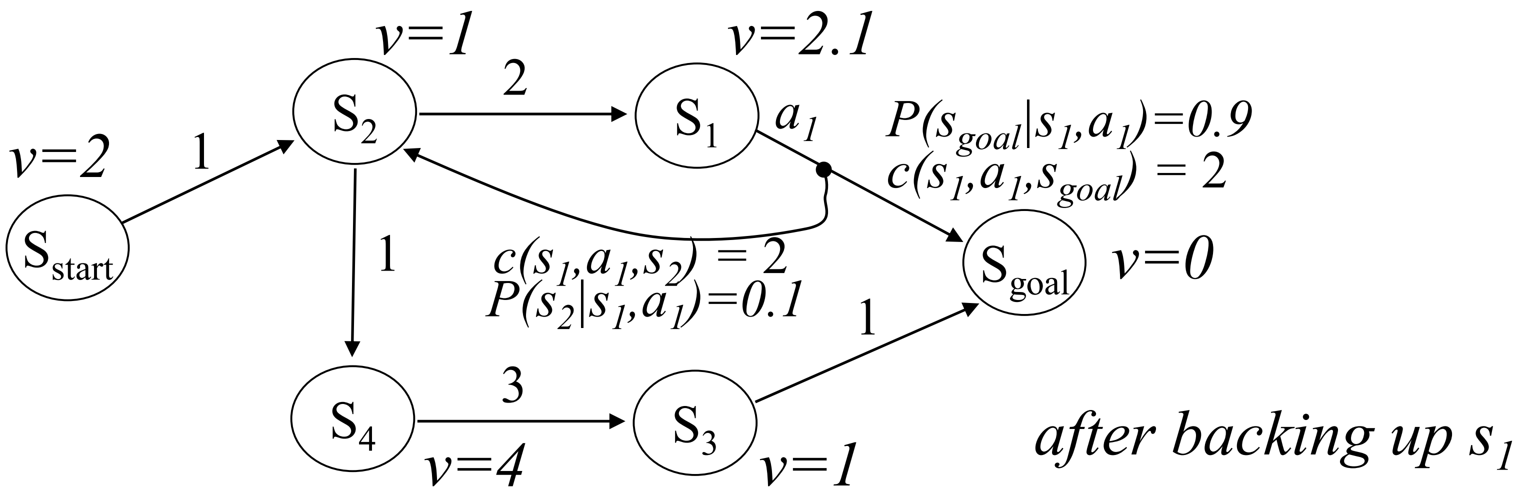
$$v(s_{goal}) = 0$$

$$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

Usual convergence condition: Bellman error over all states $< \Delta$

Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

  Initialize v-values of all states to finite values;

  Iterate over all s in MDP and re-compute until convergence:

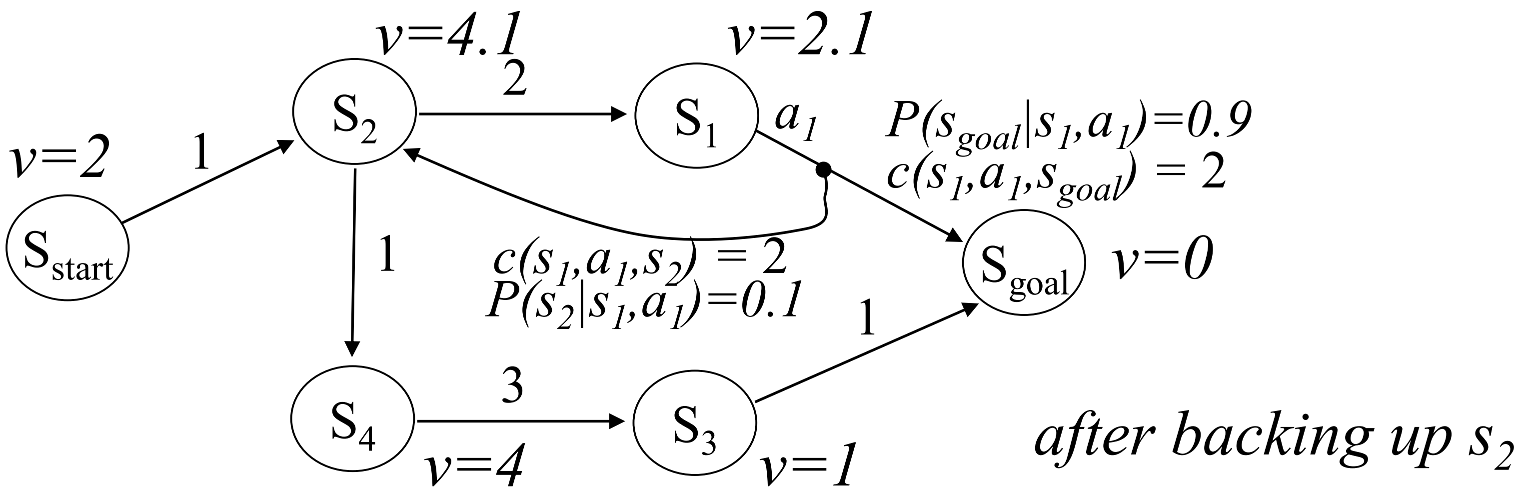
  $$v(s_{goal}) = 0$$

  $$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

  Usual convergence condition: Bellman error over all states $< \Delta$

  Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

  Initialize *v*-values of all states to finite values;

  Iterate over all *s* in MDP and re-compute until convergence:

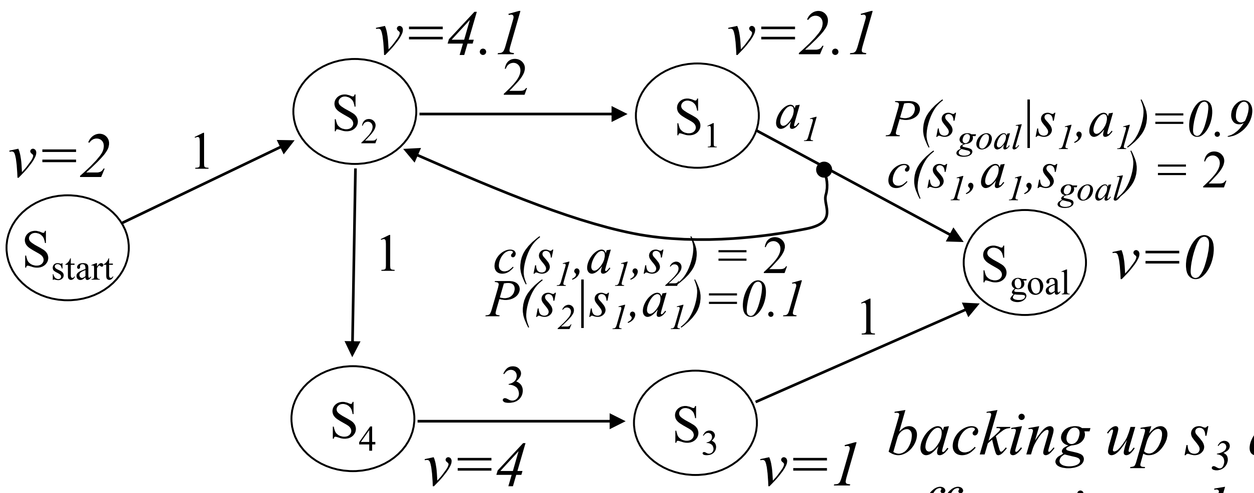
  $$v(s_{goal}) = 0$$

  $$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

  Usual convergence condition: Bellman error over all states < Δ

  Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



backing up s_3 and s_4 has no effect since their Bellman errors are zero

- Value Iteration (VI):

  Initialize *v*-values of all states to finite values;

  Iterate over all *s* in MDP and re-compute until convergence:

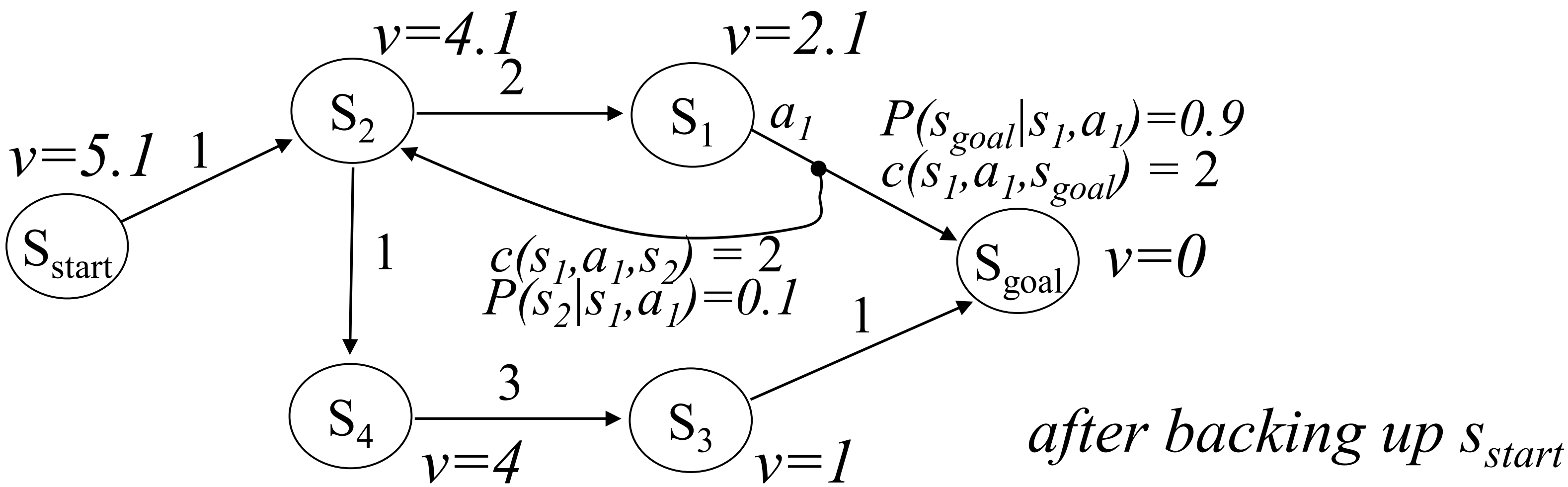
  $v(s_{goal}) = 0$

  $v(s) = \min_a E\{c(s,a,s')+v(s')\}$ *for any* $s \neq s_{goal}$

  Usual convergence condition: Bellman error over all states $< \Delta$

  Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

  Initialize v-values of all states to finite values;

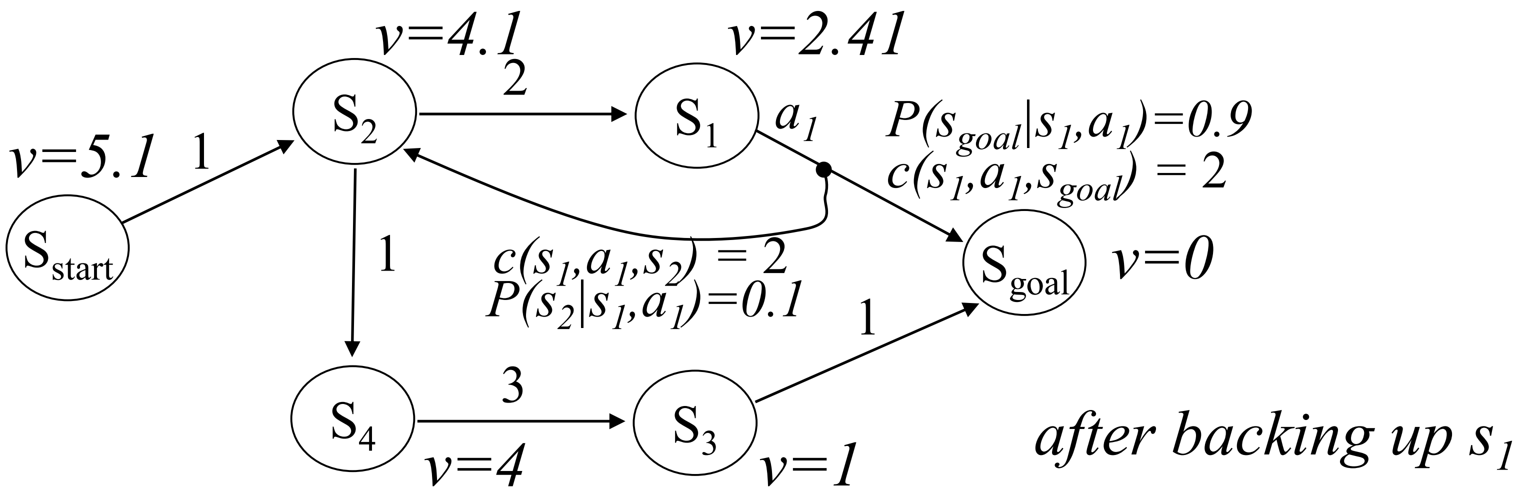
  Iterate over all s in MDP and re-compute until convergence:

  $$v(s_{goal}) = 0$$
  $$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

  Usual convergence condition: Bellman error over all states $< \Delta$

  Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

  Initialize v-values of all states to finite values;

  Iterate over all s in MDP and re-compute until convergence:

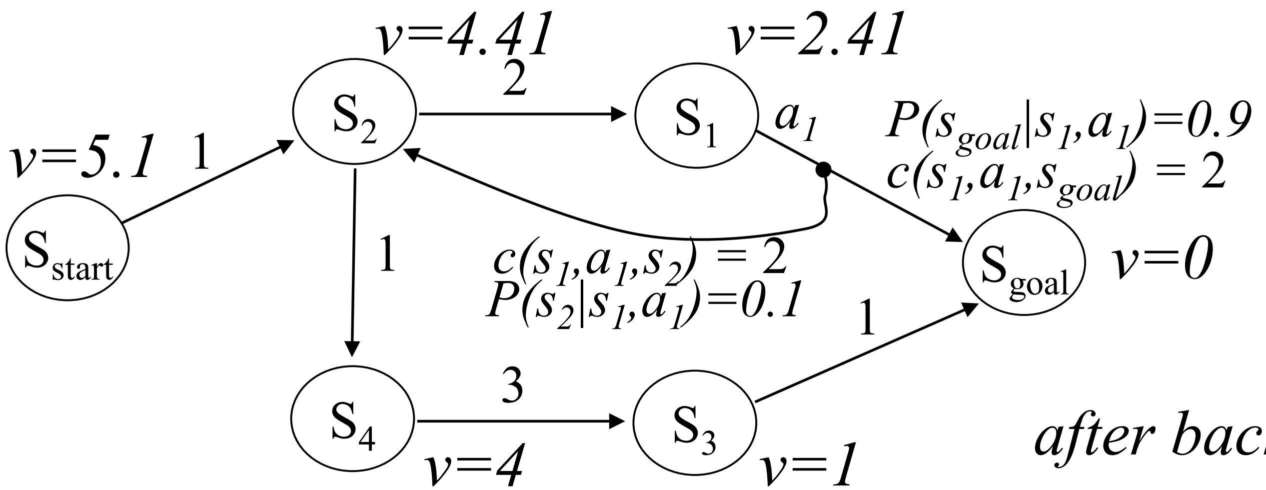
  $$v(s_{goal}) = 0$$

  $$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

  Usual convergence condition: Bellman error over all states $< \Delta$

  Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



after backing up s_2

- Value Iteration (VI):

  Initialize *v*-values of all states to finite values;

  Iterate over all *s* in MDP and re-compute until convergence:

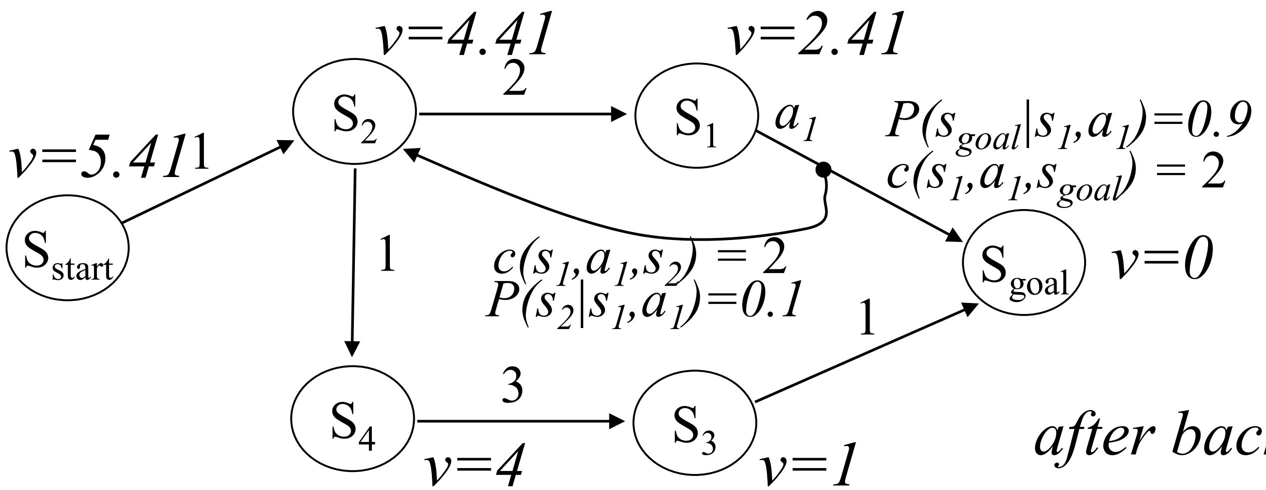
  $$v(s_{goal}) = 0$$

  $$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

  Usual convergence condition: Bellman error over all states $< \Delta$

  Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



after backing up s_{start}

- Value Iteration (VI):

  Initialize *v*-values of all states to finite values;

  Iterate over all *s* in MDP and re-compute until convergence:

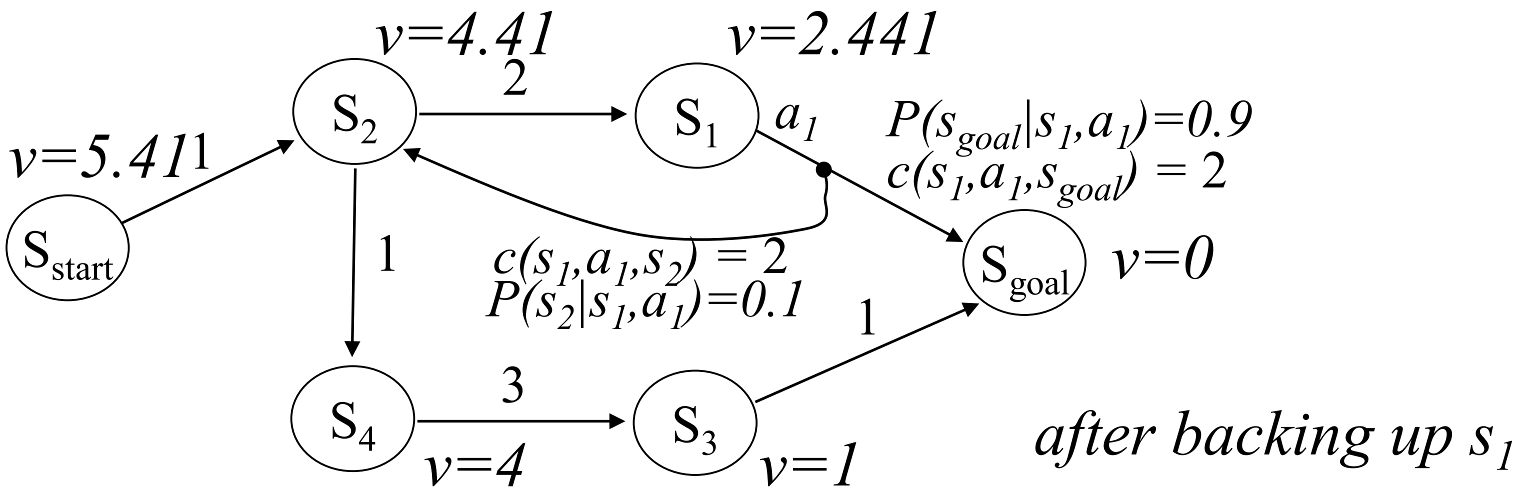
  $$v(s_{goal}) = 0$$

  $$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

  Usual convergence condition: Bellman error over all states $< \Delta$

  Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

  Initialize v-values of all states to finite values;

  Iterate over all s in MDP and re-compute until convergence:

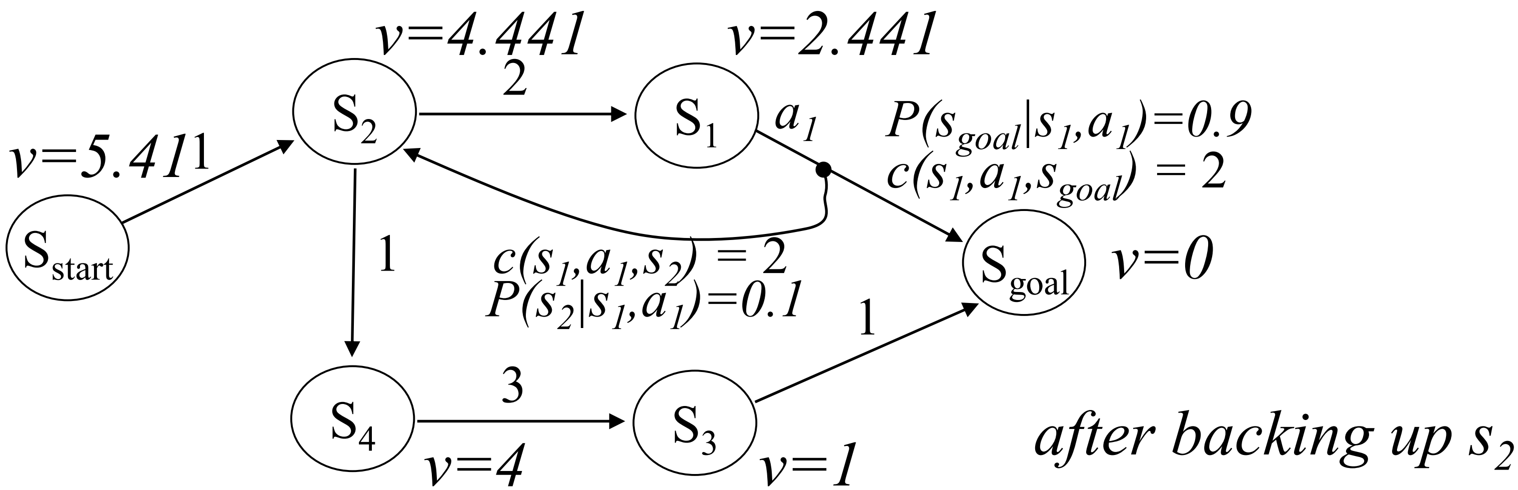
  $$v(s_{goal}) = 0$$

  $$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

  Usual convergence condition: Bellman error over all states $< \Delta$

  Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

  Initialize v-values of all states to finite values;

  Iterate over all s in MDP and re-compute until convergence:

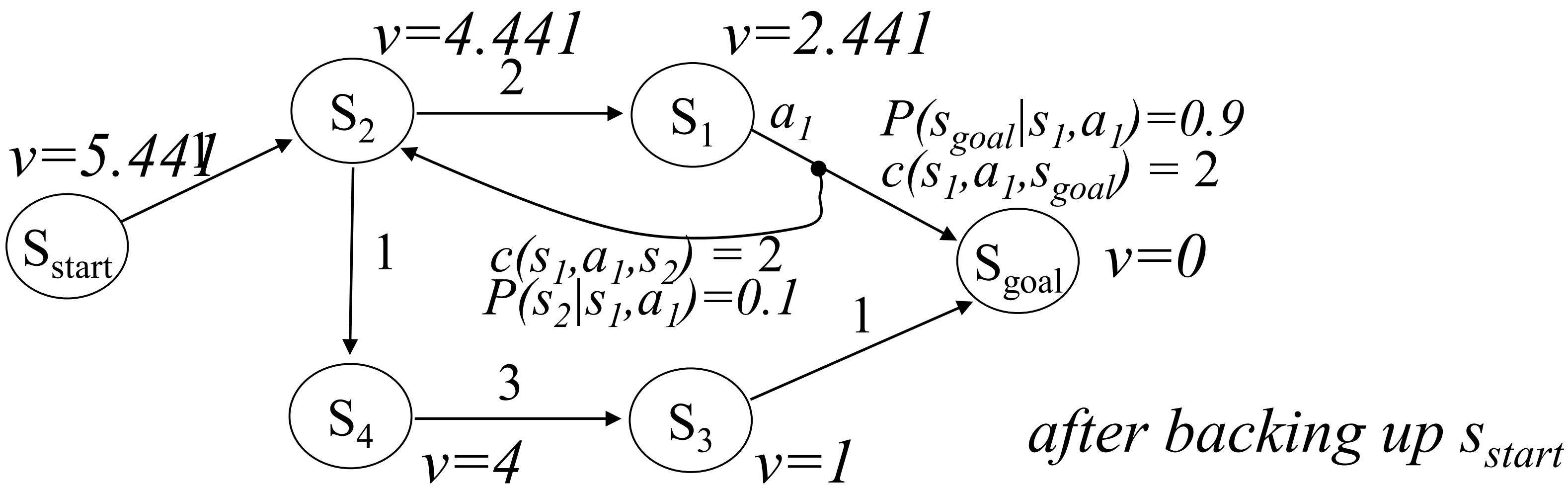
  $$v(s_{goal}) = 0$$

  $$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

  Usual convergence condition: Bellman error over all states $< \Delta$

  Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

  Initialize v-values of all states to finite values;

  Iterate over all s in MDP and re-compute until convergence:

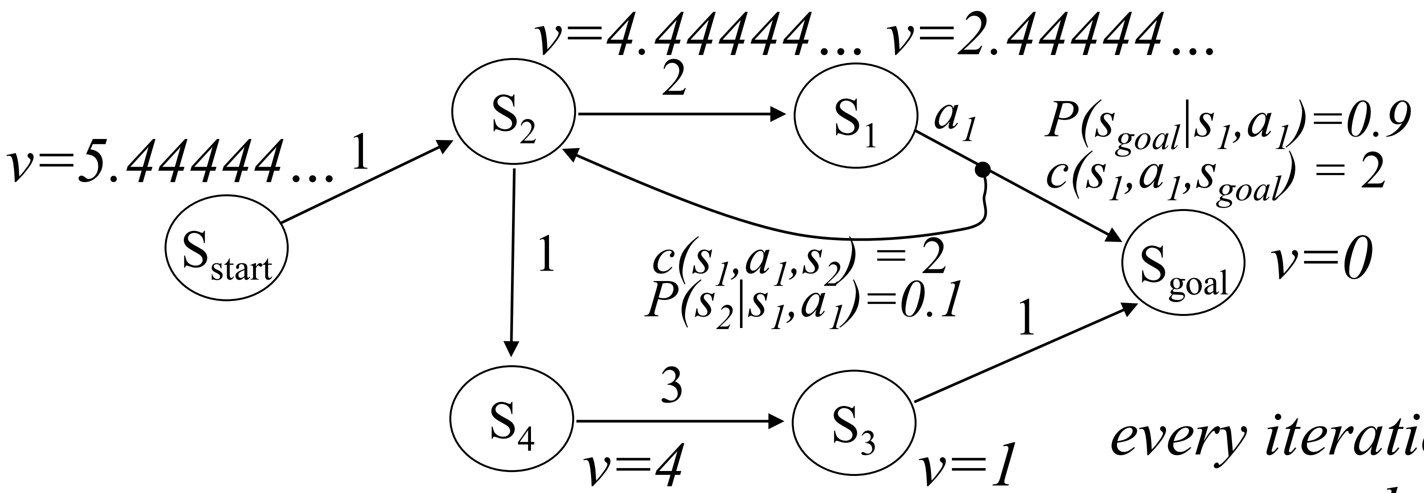
  $$v(s_{goal}) = 0$$

  $$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

  Usual convergence condition: Bellman error over all states $< \Delta$

  Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



every iteration computes one more decimal point

At convergence…

- Value Iteration (VI):

  Initialize v-values of all states to finite values;

  Iterate over all s in MDP and re-compute until convergence:

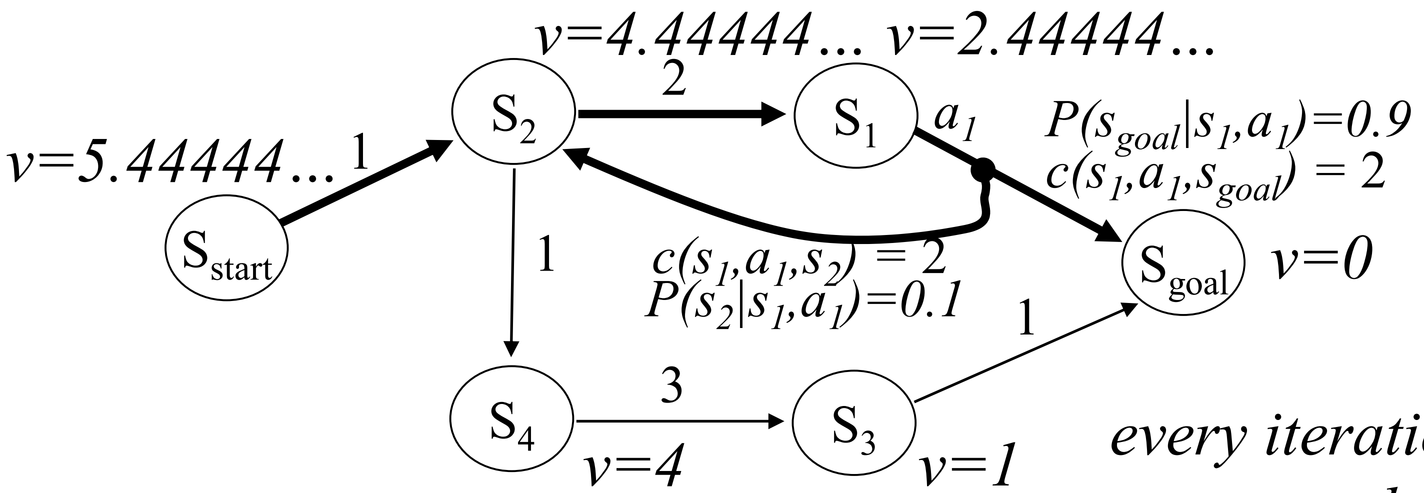
  $$v(s_{goal}) = 0$$

  $$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

  Usual convergence condition: Bellman error over all states < Δ

  Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ *for any* $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



every iteration computes one more decimal point

At convergence…

optimal policy is given by greedy policy: always select an action that minimizes $E\{c(s,a,s')+v(s')\}$

- Va…

Initialize … states to finite values;

Iterate over … and re-compute until convergence:

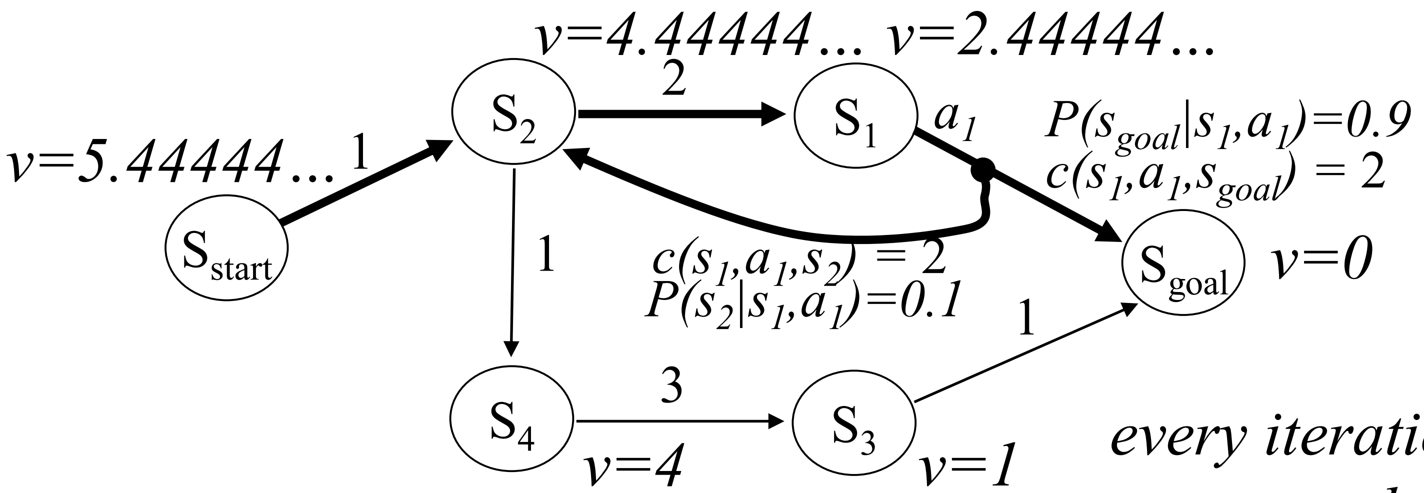
expected cost of executing greedy policy is at most: $v^*(s_{start})c_{min}/(c_{min}-\Delta)$ where c_{min} is minimum edge cost

for any $s \neq s_{goal}$

Usual co… …ellman error over all states $< \Delta$

Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ for any $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



every iteration computes one more decimal point

VI converges in finite number of iterations (assuming goal is reachable from every state)

Why condition?

- Value Iteration (VI):

  Initialize v-values of all states to fi

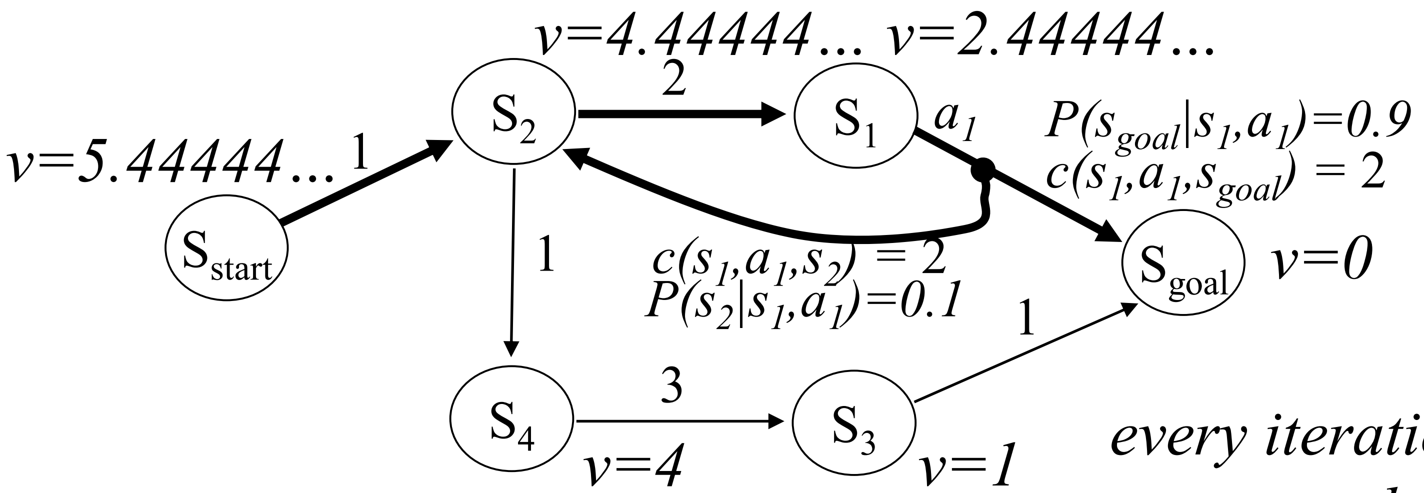
  Iterate over all s in MDP and re-compute until convergence:

  $$v(s_{goal}) = 0$$
  $$v(s) = \min_a E\{c(s,a,s')+v(s')\} \text{ for any } s \neq s_{goal}$$

  Usual convergence condition: Bellman error over all states < Δ

  Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ for any $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



every iteration computes one more decimal point

VI converges in finite number of iterations (assuming goal is reachable from every state)

- Value Iteration (VI):

  Initialize v-values of all states to fin...

  Iterate over all s in MDP and r...

  $v(s_{goal}) = 0$

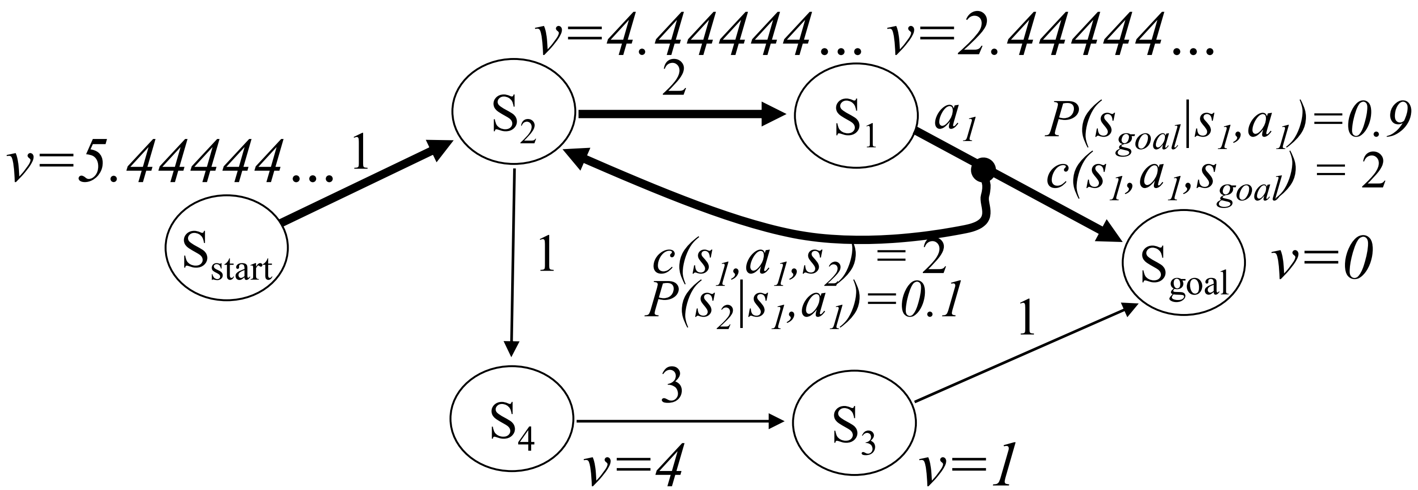
  $v(s) = \min_a E\{c(s,a,s')+v(s')\}$ for ...

How many backups required in a graph with no stochastic actions?

Usual convergence condition: Bellman error over all states < Δ

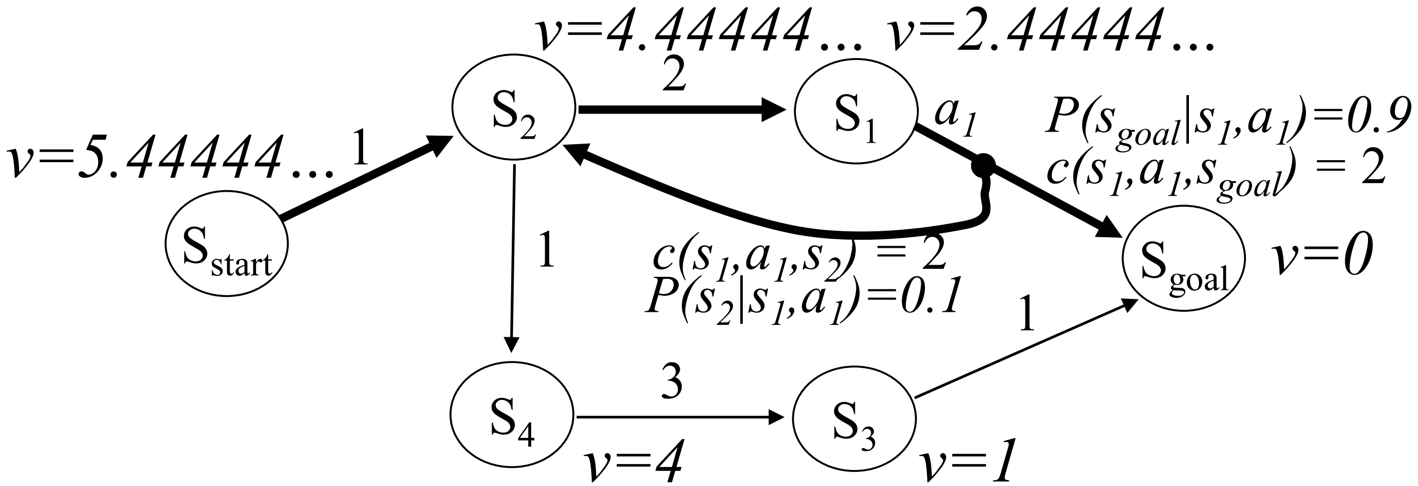
Bellman error: $|v(s) - \min_a E\{c(s,a,s')+v(s')\}|$ for any $s \neq s_{goal}$

Computing Expected Cost Minimal Plans with RTDP



- Real-time Dynamic Programming (RTDP)
  - very popular alternative to Value Iteration
  - does NOT compute values of all states
  - focusses computations on states that are relevant
  - typically, **much more efficient than Value Iteration**

Computing Expected Cost Minimal Plans with RTDP



- RTDP:

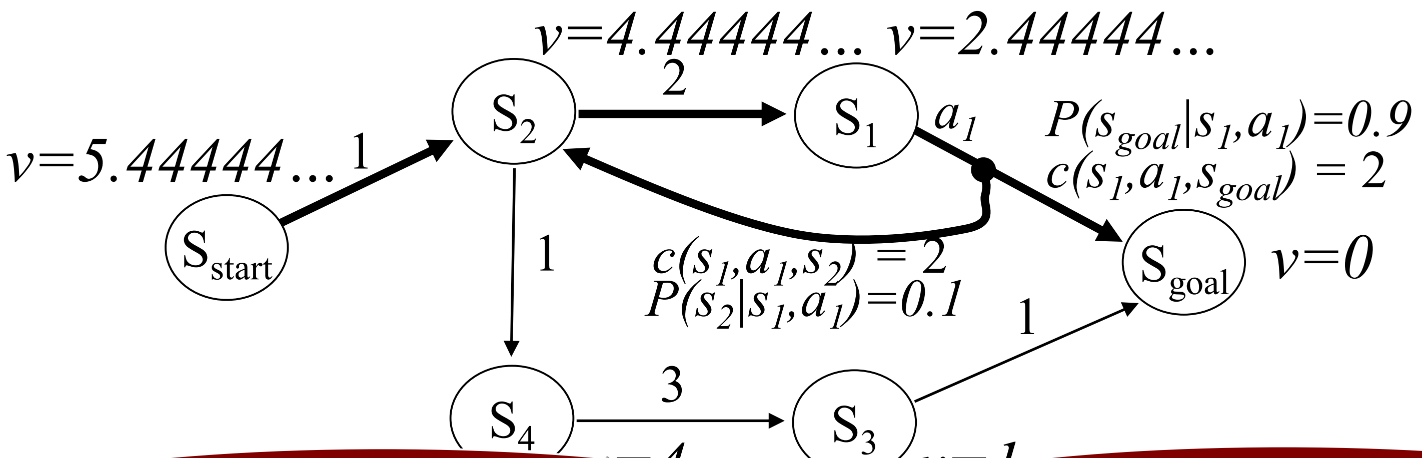
  Initialize v-values of all states to admissible values;

  1. Follow greedy policy picking outcomes at random until goal is reached;

  2. Backup all states visited on the way;

  3. Reset to s_{start} and repeat 1-3 until all states on the current greedy policy have Bellman errors < Δ;

Computing Expected Cost Minimal Plans with RTDP



For any state s, picking action a that minimizes E{c(s,a,s')+v(s')}

Picking successor state s' at random according to probability P(s'|a,s)

- RTDP:

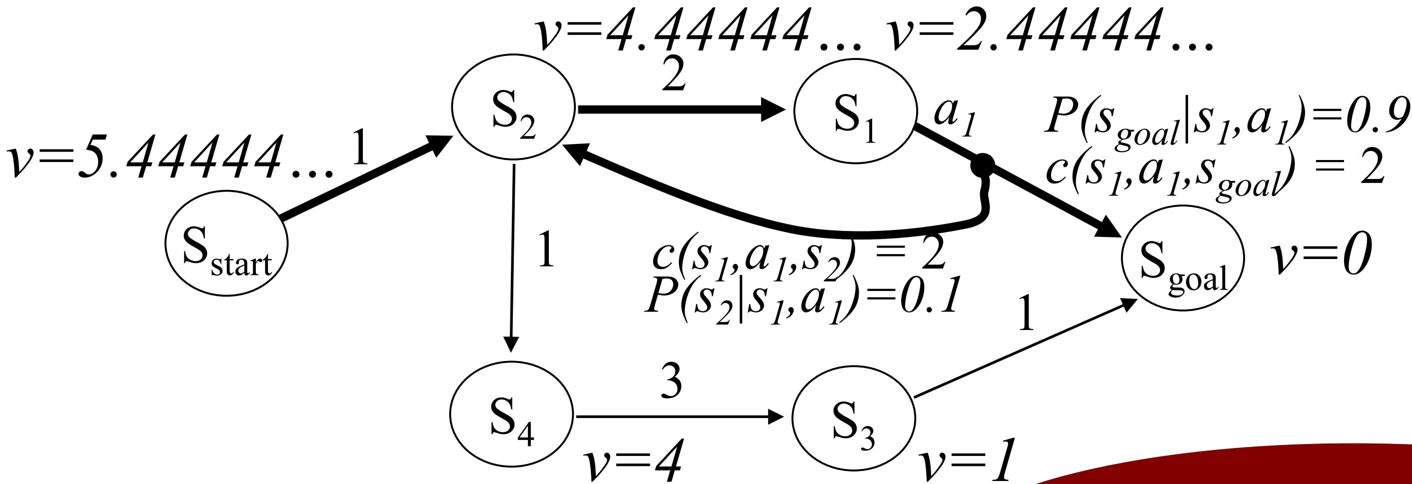
  Initialize v-values of all states to admissible values;

  1. Follow greedy policy picking outcomes at random until goal is reached;

  Updating $v(s) = \min_a E\{c(s,a,s')+v(s')\}$

  2. Backup all states visited on the way;
  3. Reset to s_{start} and repeat 1-3 until all states on the current greedy policy have Bellman errors < Δ;

Computing Expected Cost Minimal Plans with RTDP



*RTDP **focusses** its backups on what is relevant to the optimal plan rather than computing ALL state values (what VI does)*

- RTDP:

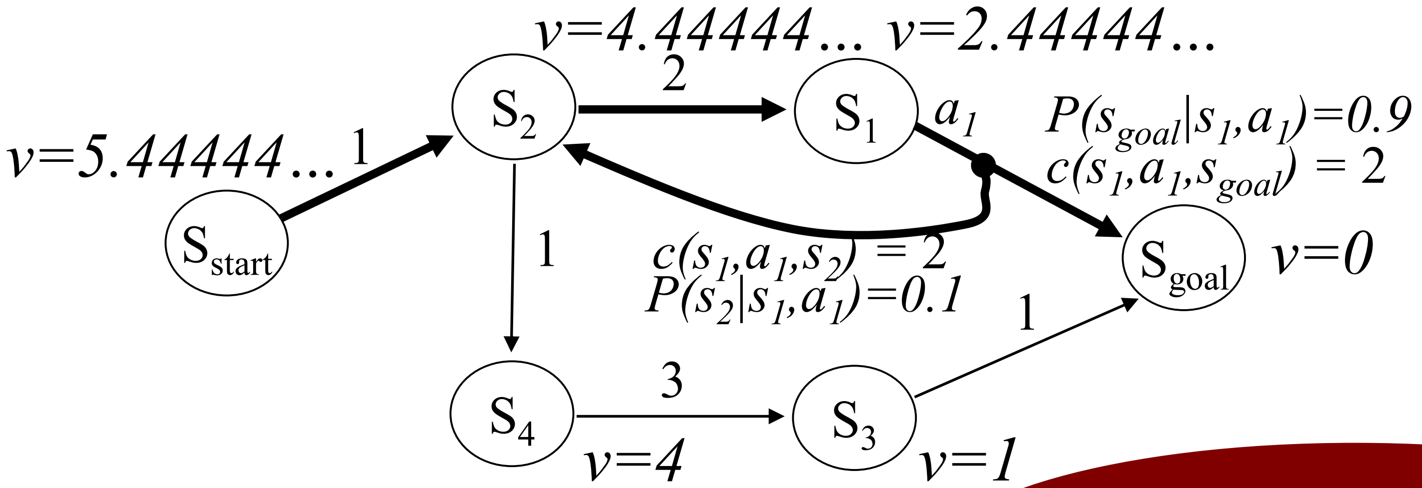
  Initialize *v*-values of all states to admissible values;

  1. Follow greedy policy picking outcomes at random until goal is reached;

  2. Backup all states visited on the way;

  3. Reset to s_{start} and repeat 1-3 until all states on the current greedy policy have Bellman errors < Δ;

Computing Expected Cost Minimal Plans with RTDP



RTDP converges in finite number of iterations (assuming goal is reachable from every state)

- RTDP:

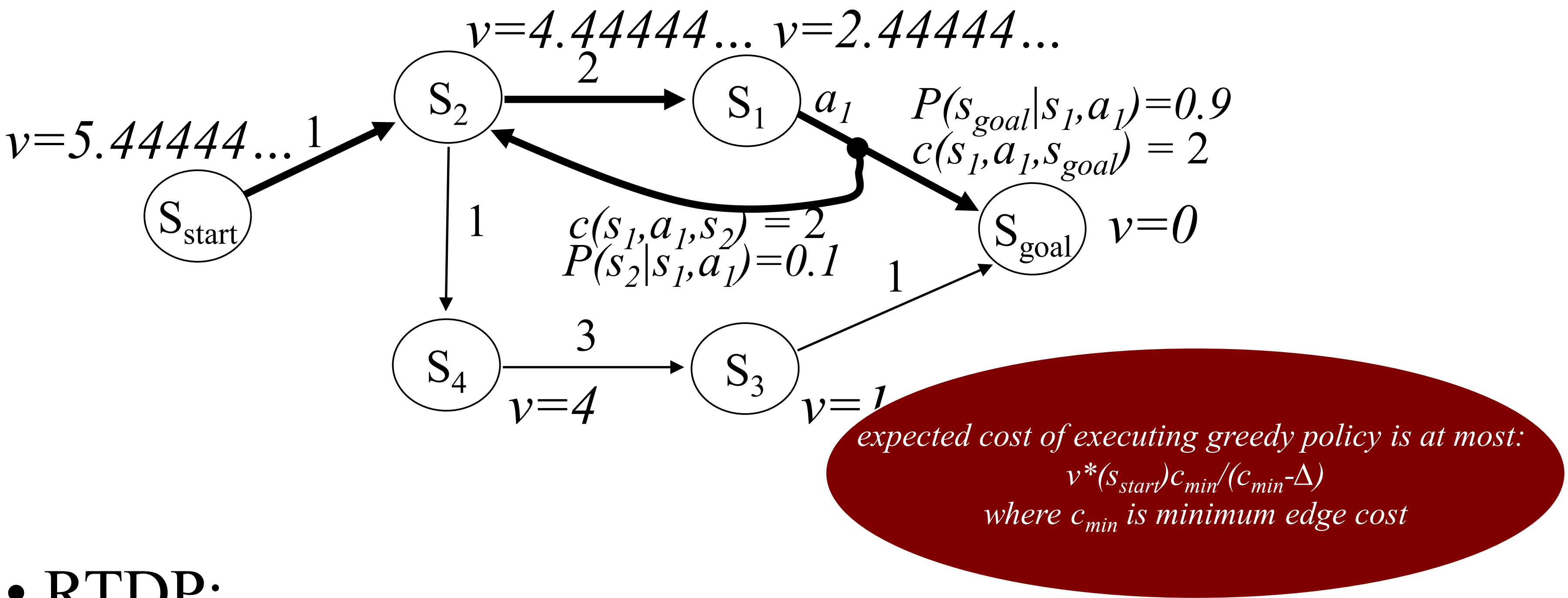
  Initialize *v*-values of all states to admissible values;

  1. Follow greedy policy picking outcomes at random until goal is reached;

  2. Backup all states visited on the way;

  3. Reset to s_{start} and repeat 1-3 until all states on the current greedy policy have Bellman errors < Δ;

Computing Expected Cost Minimal Plans with RTDP



- RTDP:

Initialize *v*-values of all states to admissible values;

1. Follow greedy policy picking outcomes at random until goal is reached;

2. Backup all states visited on the way;

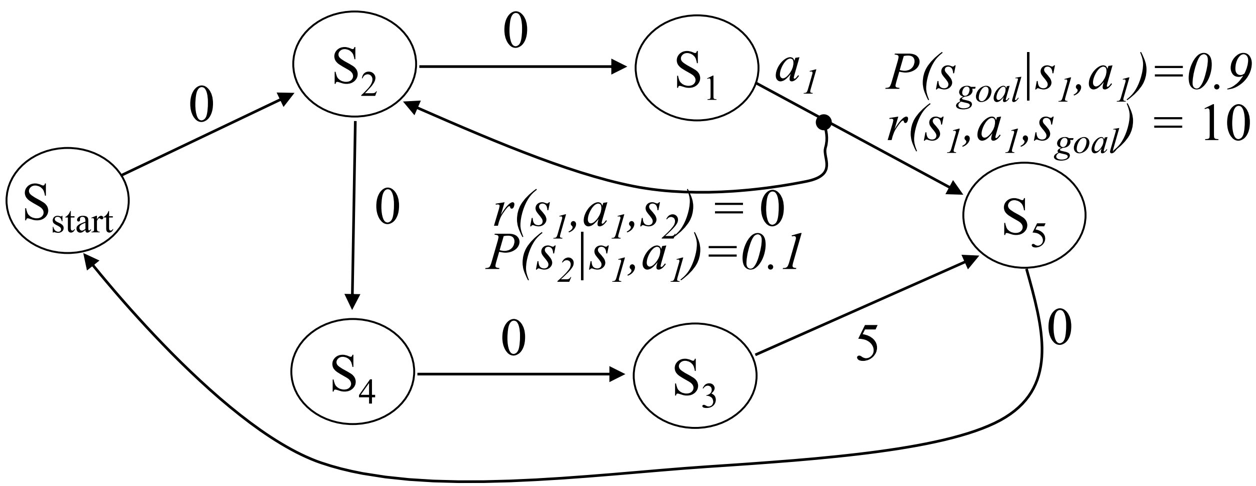
3. Reset to s_{start} and repeat 1-3 until all states on the current greedy policy have Bellman errors < Δ;

Rewards version of MDPs

- Suppose we have a Trash Collecting robot
  - its task is to go around the room and pick-up trash
  - if battery is dead, it can't move anymore
  - available actions:
    - Look for trash (takes 1 min) and discovers trash with probability 0.4
    - Pick-up trash (takes 1 min), <u>and receive reward of 100 units</u>
    - Re-charge (takes 1 min). Battery level goes back to full 3 mins if successful with probability 0.9 (there is a chance that re-charge is not successful)

Example on the board

Markov Decision Processes, REWARDS version



- Optimal expected reward values v^* satisfy:

  $v^*(s) = \boldsymbol{max}_a\, E\{r(s,a,s') + \gamma v^*(s')\}$ *for all s*

  (expectation over outcomes s' of action a executed at state s)

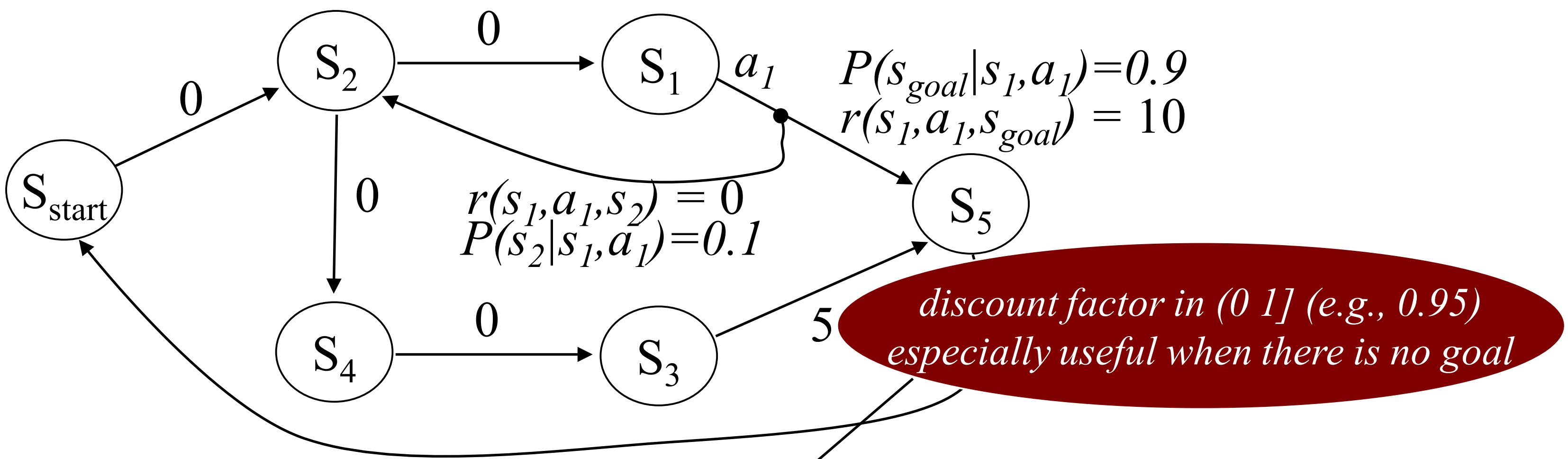
- Optimal policy π^*:

  $\pi^*(s) = arg\boldsymbol{max}_a\, E\{r(s,a,s') + \gamma v^*(s')\}$

- Computing optimal v^*-values via value iteration (VI):

  re-compute $v(s) = \boldsymbol{max}_a\, E\{r(s,a,s') + \gamma v(s')\}$ *until convergence*

Markov Decision Processes, REWARDS version



- Optimal expected reward values v^* satisfy:

  $v^*(s) = \boldsymbol{max}_a\, E\{r(s,a,s')+\gamma v^*(s')\}$ *for all s*

  (expectation over outcomes s' of action a executed at state s)

- Optimal policy π^*:

  $\pi^*(s) = arg\boldsymbol{max}_a\, E\{r(s,a,s')+ \gamma v^*(s')\}$

- Computing optimal v^*-values via value iteration (VI):

  re-compute $v(s) = \boldsymbol{max}_a\, E\{r(s,a,s')+ \gamma v(s')\}$ *until convergence*

What You Should Know…

- Pros and Cons of solving Expected Cost formulation (rather than Minimax formulation)
- The operation of Value Iteration
- The operation of RTDP
- Rewards formulation of MDPs and when it should be used