Intelligence II:
Advanced Decision-Making Mechanisms

Advanced Decision-making Mechanisms for this Class

- More on Behavior Trees
- Planning to Achieve the Goal
- Planning with Uncertainty to Achieve the Goal

Advanced Decision-making Mechanisms for this Class

- More on Behavior Trees

- Planning to Achieve the Goal

- Planning with Uncertainty to Achieve the Goal

More on Behavior Trees

- Additional task: *Decorator*
- Has only one child whose execution it controls in a special way

Example:

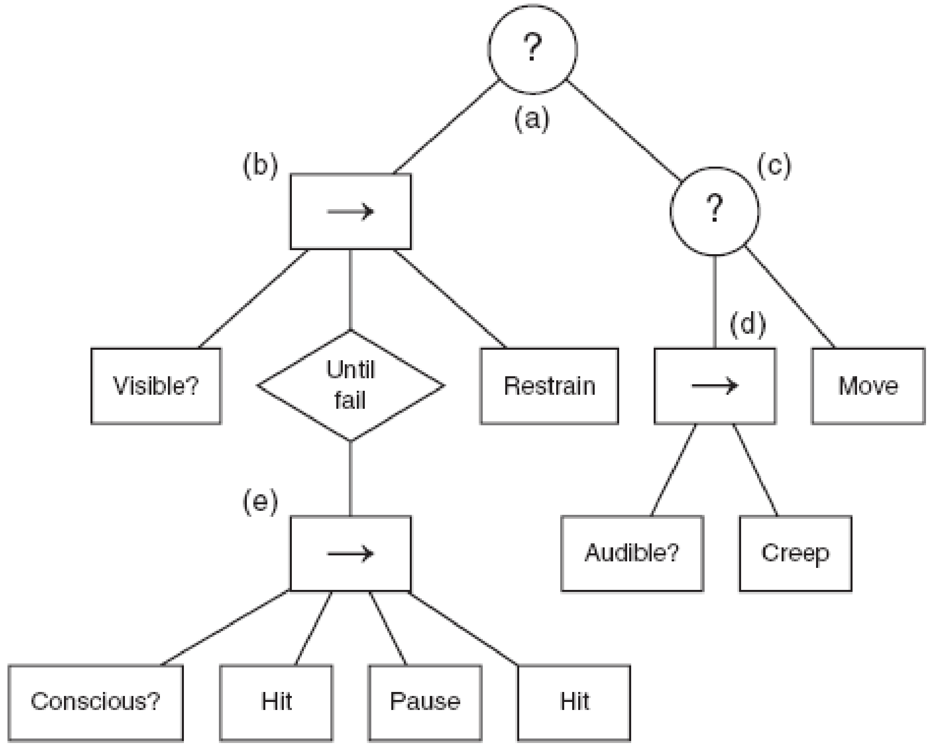
“y decorators” decide whether to execute its children based on some conditions

More on Behavior Trees

- Additional task: *Decorator*
- Has only one child whose execution it controls in a special way

Example:

"Loop decorators" loop until failure



from "Artificial Intelligence for Games" by I. Millington & J. y ge

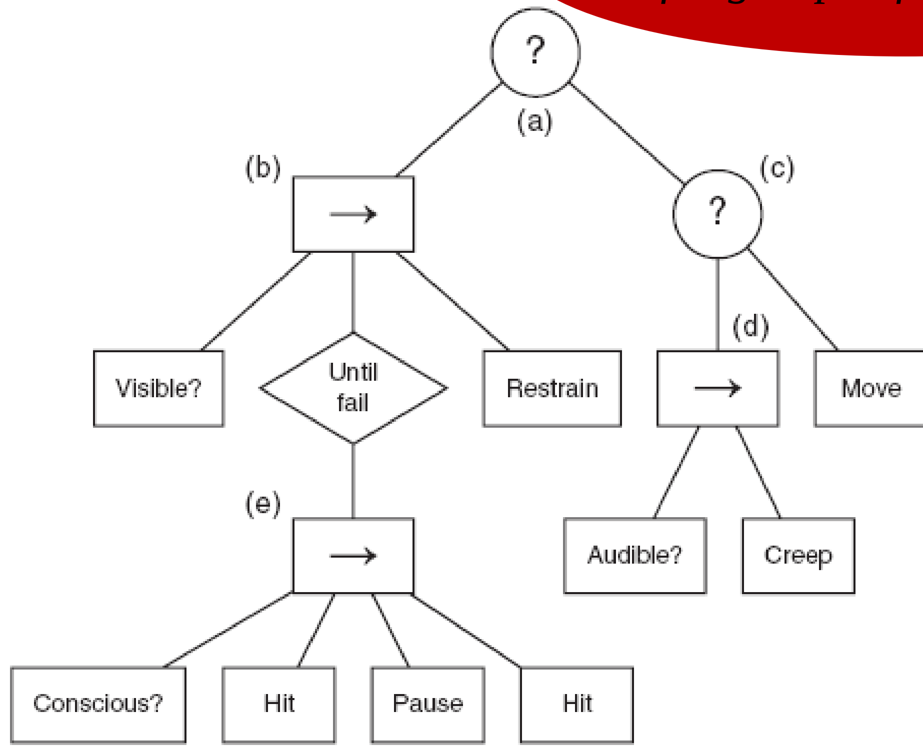
More on Behavior Trees

- Additional task: *Decorator*
- Has only one child whose execution it controls in a special way

Example:

“Loop decorators” loop until fai

What about behavior trees for groups of characters?

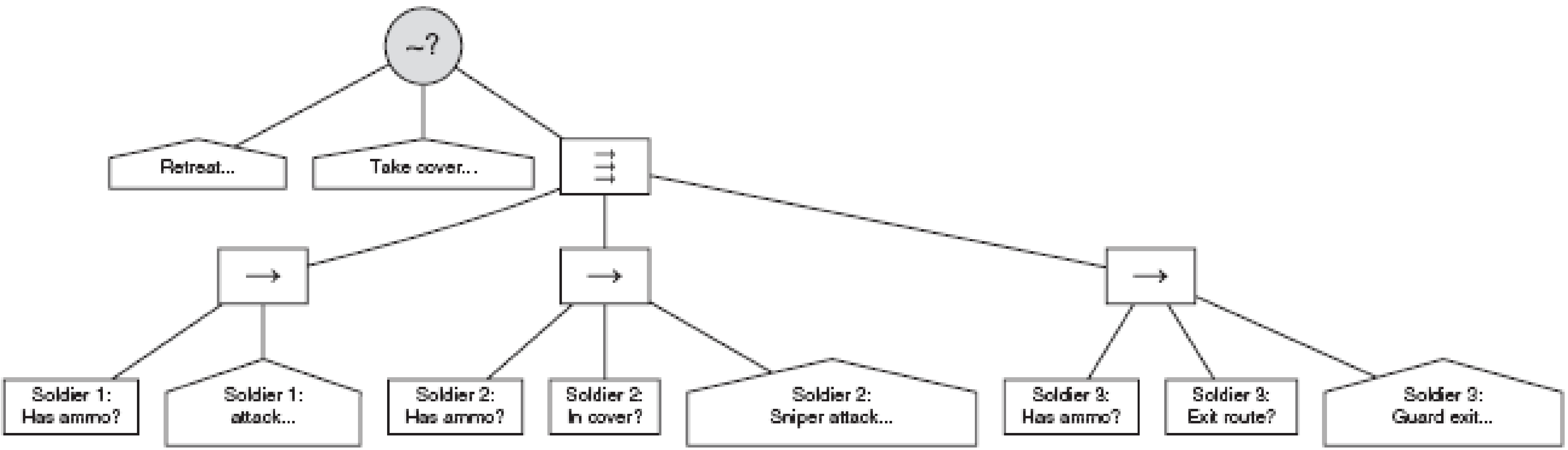


from “Artificial Intelligence for Games” by I. Millington & J. y ge

More on Behavior Trees

- Additional task: *Parallel*
- Executes tasks in parallel until first fails or all succeed

Example:

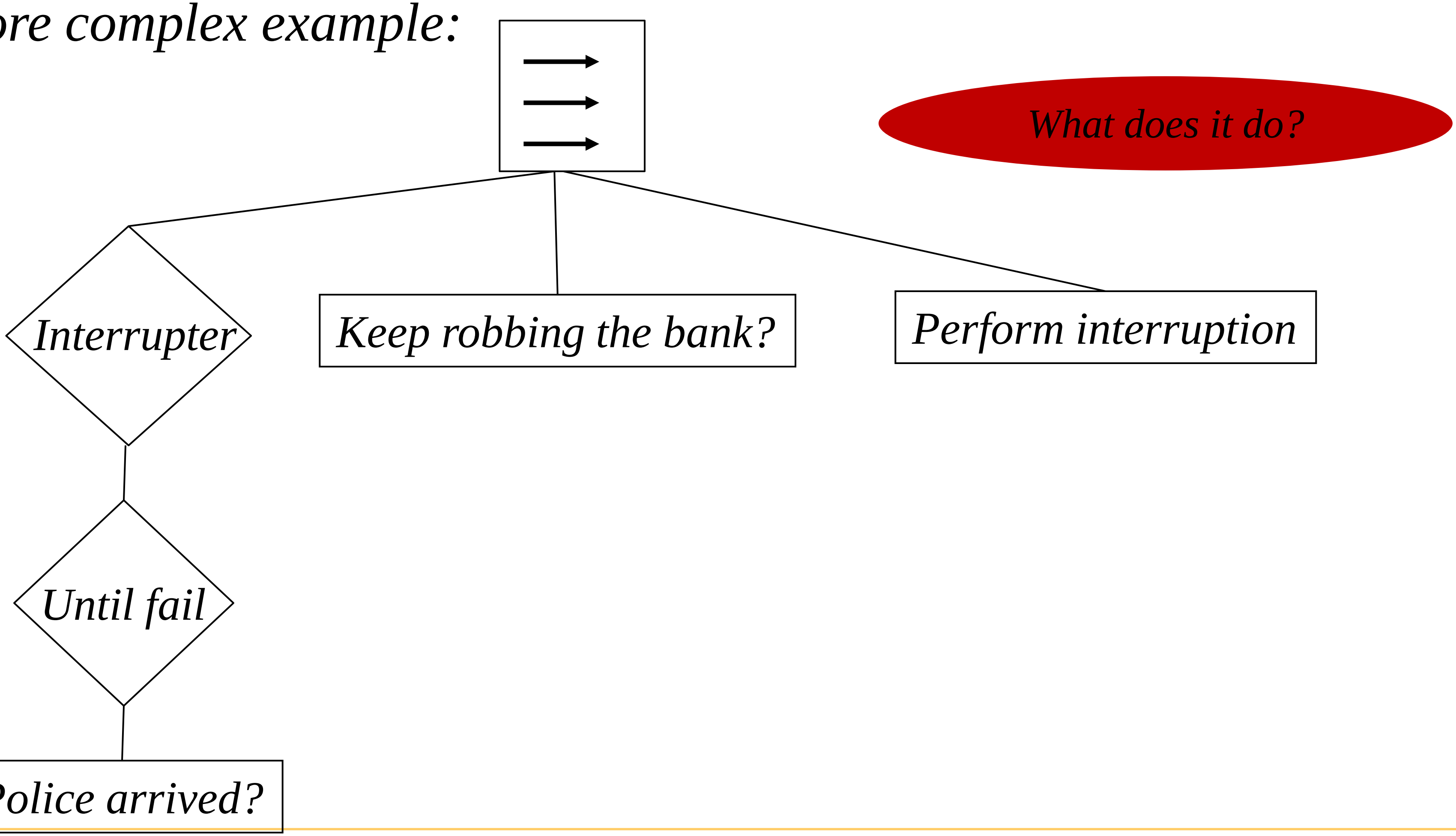


from "Artificial Intelligence for Games" by I. Millington & J. y ge

More on Behavior Trees

- Additional task: *Parallel*
- Executes tasks in parallel until first fails or all succeed

More complex example:



Advanced Decision-making Mechanisms for this Class

- More on Behavior Trees
- Planning to Achieve the Goal
- Planning with Uncertainty to Achieve the Goal

Goal-Oriented Behavior

- Beyond hard-coding stimulus-response pairs
- Seeks to satisfy internal goals (e.g., hunger, threat, gold, etc.)

Goal-Oriented Behavior

- Beyond hard-coding stimulus-response pairs
- Seeks to satisfy internal goals (e.g., hunger, threat, gold, etc.)

Three components:

Goals (motives),

How Pressing each Goal is (insistence)

Actions with Expected Impact on the Insistence of Each Goal

Goal-Oriented Behavior

- Beyond hard-coding stimulus-response pairs
- Seeks to satisfy internal goals (e.g., hunger, threat, gold, etc.)

Example:

Goals with Insistence Values:

Eat = 9, Kill Enemy = 8, Get Healthy = 4

Actions with Impact on Insistence Values of Goals:

Get Food (Eat: -5)

Kill Enemy (Kill Enemy: -8, Get Healthy: +4)

Get Health Pack (Get Healthy: –2)

Goal-Oriented Behavior

- Beyond hard-coding stimulus-response pairs
- Seeks to satisfy internal goals (e.g., hunger, threat, gold, etc.)

Example:

Goals with Insistence Values:

Eat = 9, Kill Enemy = 8, Get Healthy = 4

Actions with Impact on Insistence Values of Goals:

Get Food (Eat: -5)

Kill Enemy (Kill Enemy: -8, Get Healthy: +4)

Get Health Pack (Get Healthy: –2)

Negative side effect of the action

Pick action that has the best net effect (could be weighted)

Goal-Oriented Behavior

- Beyond hard-coding stimulus-response pairs
- Seeks to satisfy internal goals (e.g., hunger, threat, gold, etc.)

Example:

Potential problems?

Goals with Insistence Values:

Eat = 9, Kill Enemy = 8, Get Healthy = 4

Actions with Impact on Insistence Values of Go

Negative side effect of the action

Get Food (Eat: -5)

Kill Enemy (Kill Enemy: -8, Get Healthy: +4)

Get Health Pack (Get Healthy: –2)

Pick action that has the best net effect (could be weighted)

Goal-Oriented Behavior

- Beyond hard-coding stimulus-response pairs
- Seeks to satisfy internal goals (e.g., hunger, threat, gold, etc.)

Optimal sequence?

Example:

Suppose the character is under attack and can pick a weapon that allows it to more effectively shoot the enemies

Goals with Insistence Values:

Eat = 7, Kill Enemy = 8, Get Healthy = 4, NewWeapons=1

Actions with Impact on Insistence Values of Goals:

Get Food (Eat: -5, Get Healthy: +2)

Kill Enemy (Kill Enemy: -8, Get Healthy: +7)

Get Health Pack (Get Healthy: –2, Eat: +2)

Get Weapon (NewWeapons: -1, Get Healthy: +8

plus Kill Enemy action will have no impact on Health)

Goal-Oriented Behavior

- Beyond hard-coding stimulus-response pairs
- Seeks to satis

Optimal sequence:
Get Health Pack, Get Weapon, Kill Enemy

How to compute it?

Example:
Suppose the character is under attack and can pick a weapon that allows it to more effectively shoot the enemies

Goals with Insistence Values:

Eat = 7, Kill Enemy = 8, Get Healthy = 4, NewWeapons=1

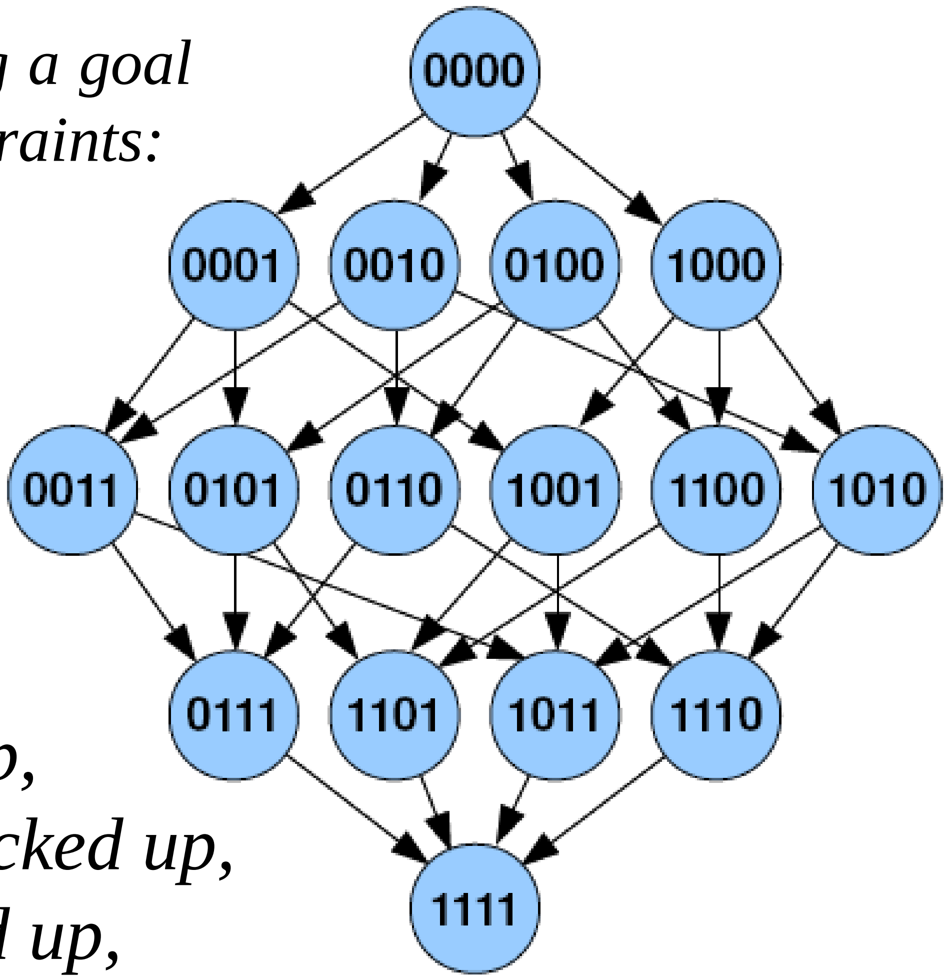
Actions with Impact on Insistence Values of Goals:

Get Food (Eat: -5, Get Healthy: +2)
Kill Enemy (Kill Enemy: -8, Get Healthy: +7)
Get Health Pack (Get Healthy: –2, Eat: +2)
Get Weapon (NewWeapons: -1, Get Healthy: +8
plus Kill Enemy action will have no impact on Health)

Goal-Oriented Planning

- Beyond single-step decisions

Search-space (graph) for finding a goal that satisfies the necessary constraints:



Example:

initial state is 0000

goal state is 1111

(e.g., bit 0: food is picked up,

bit 1: health pack is picked up,

bit 2: weapon is picked up,

bit 3: enemy is shot)

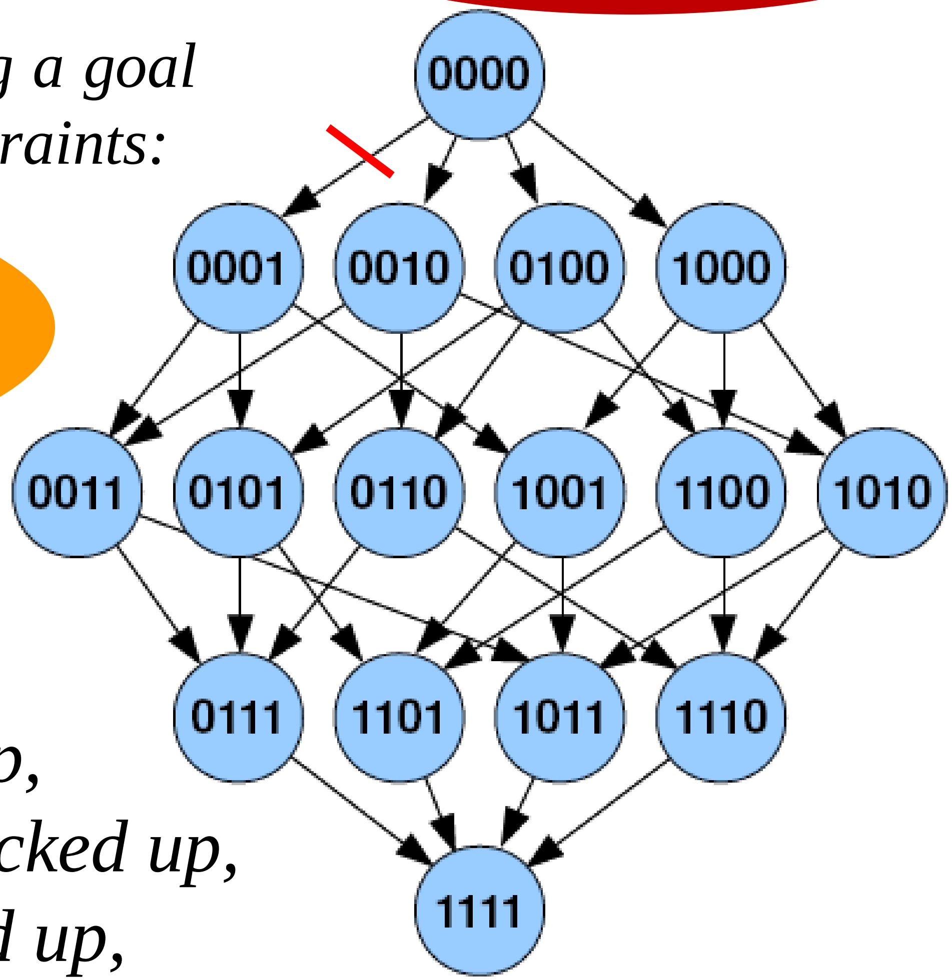
Goal-Oriented Planning

- Beyond single-step decisions

What could be the transition costs?

Search-space (graph) for finding a goal that satisfies the necessary constraints:

The validity of a transition depends on the source state (e.g., can't shoot enemy weapon was picked up)



Example:

initial state is 0000

goal state is 1111

(e.g., bit 0: food is picked up,

bit 1: health pack is picked up,

bit 2: weapon is picked up,

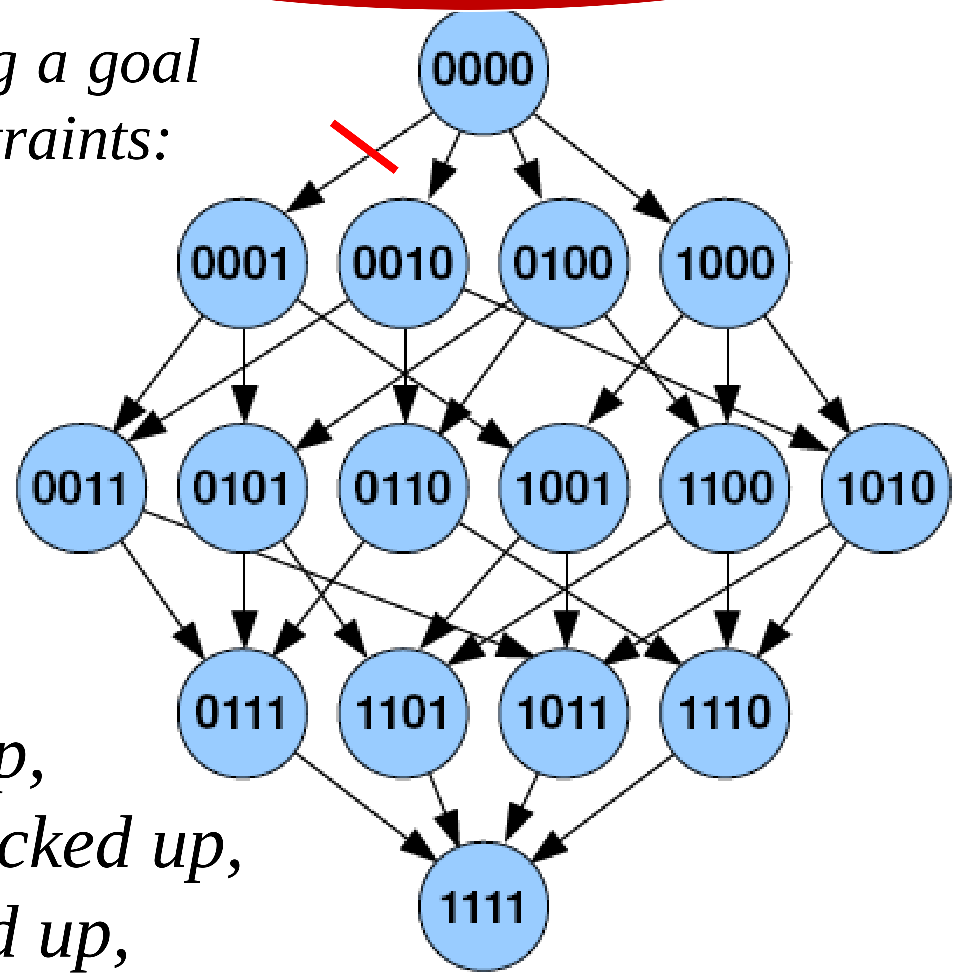
bit 3: enemy is shot)

Goal-Oriented Planning

- Beyond single-step decisions

What if the goal is to shoot enemy (whether health pack and food are picked up or not)?

Search-space (graph) for finding a goal that satisfies the necessary constraints:



Example:

initial state is 0000

goal state is 1111

(e.g., bit 0: food is picked up,

bit 1: health pack is picked up,

bit 2: weapon is picked up,

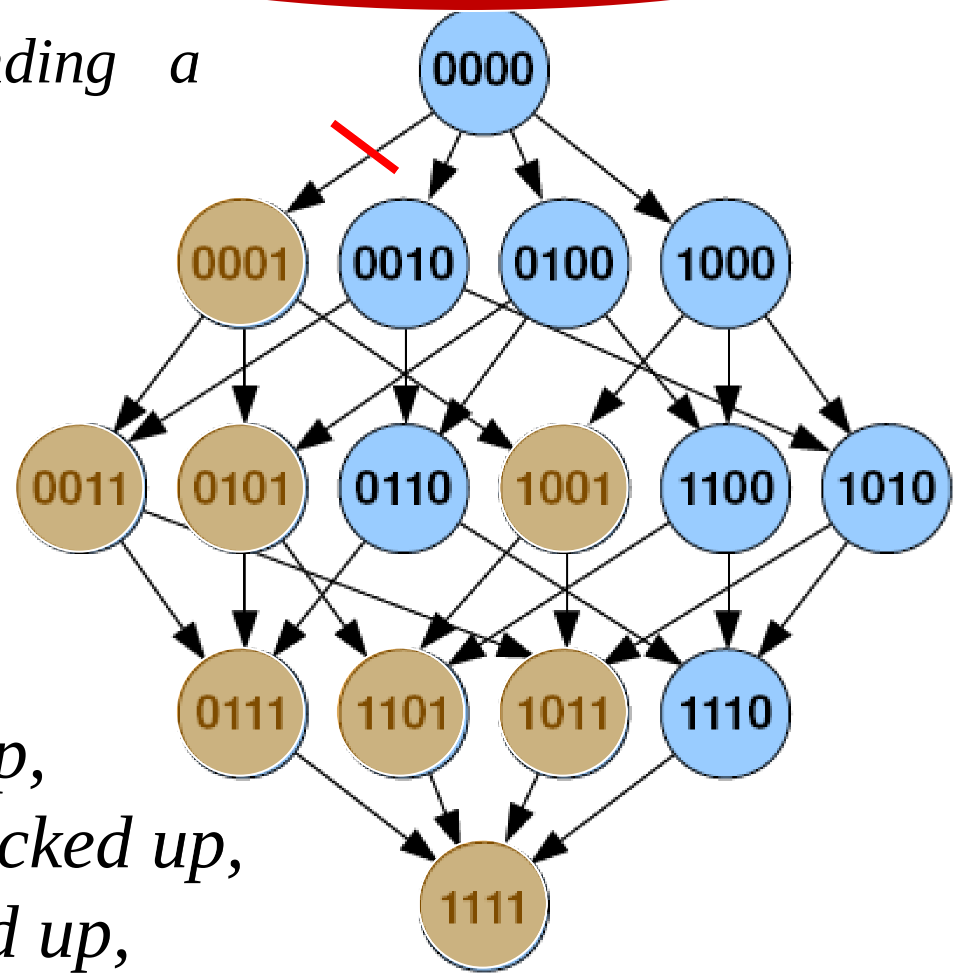
bit 3: enemy is shot)

Goal-Oriented Planning

- Beyond single-step decisions

What if the goal is to shoot enemy (whether health pack and food are picked up or not)?

Search-space (graph) for finding a partially-defined goal:



Example:

initial state is 0000

goal state is 1111

(e.g., bit 0: food is picked up,

bit 1: health pack is picked up,

bit 2: weapon is picked up,

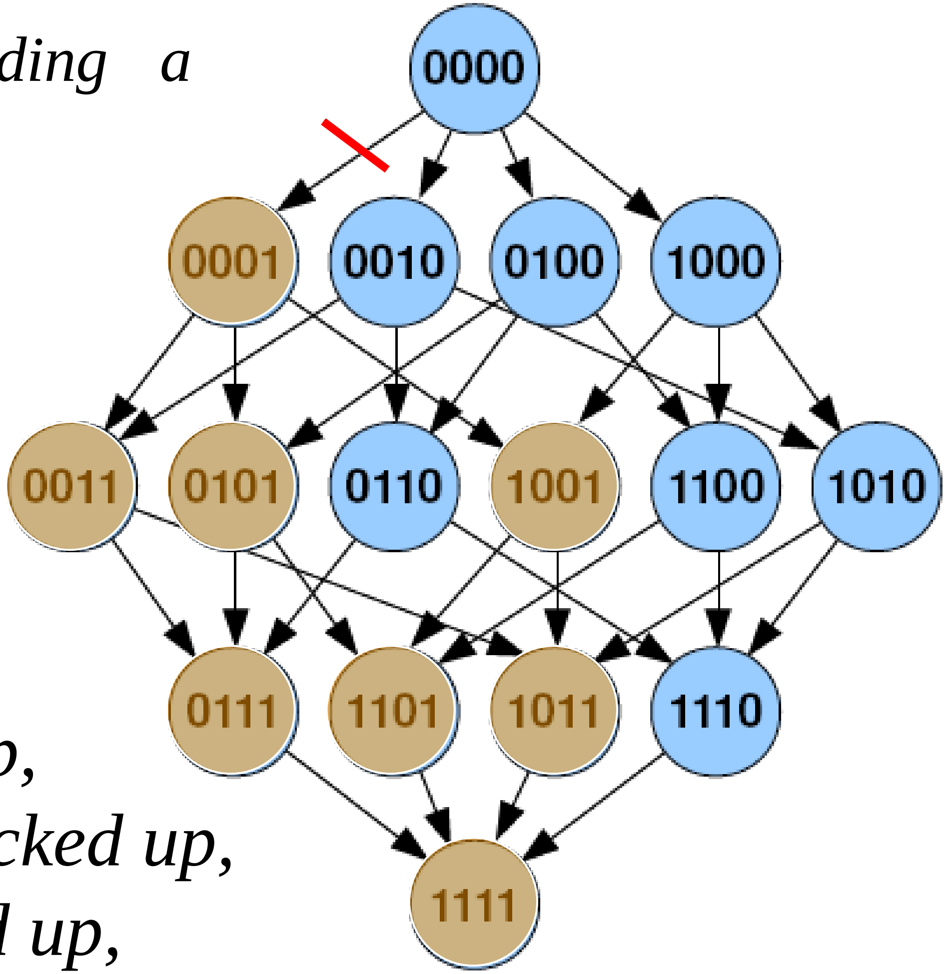
bit 3: enemy is shot)

Goal-Oriented Planning

What are the efficient way to represent the state vectors?

- Beyond single-step decisions

Search-space (graph) for finding a partially-defined goal:



Example:

initial state is 0000

goal state is 1111

(e.g., bit 0: food is picked up,

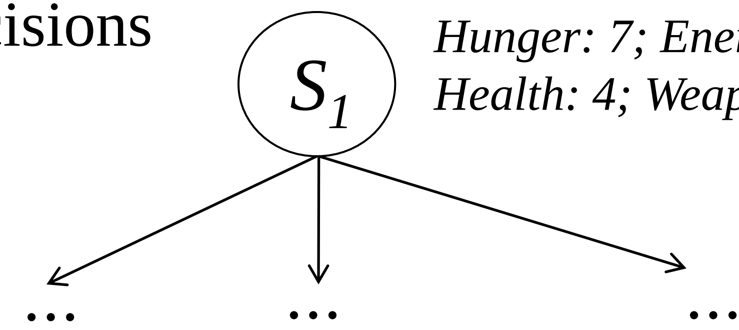
bit 1: health pack is picked up,

bit 2: weapon is picked up,

bit 3: enemy is shot)

Goal-Oriented Planning

- Beyond single-step decisions



Hunger: 7; Enemy shot: 0;
Health: 4; Weapon: 1

Working out Example with some non-binary variables:

Goals with Insistence Values:

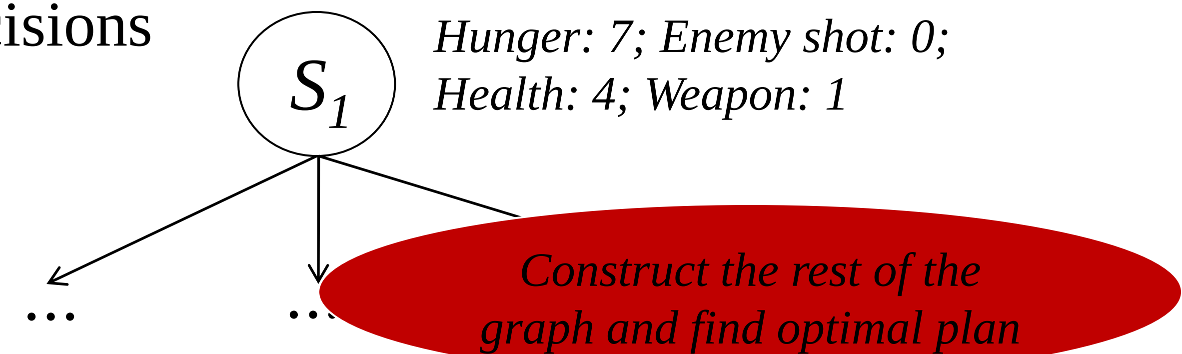
Eat = 7, Kill Enemy = 8, Get Healthy = 4, NewWeapons=1

Actions with Impact on Insistence Values of Goals:

Get Food (Eat: -5, Get Healthy: +2)
Kill Enemy (Kill Enemy: -8, Get Healthy: +7)
Get Health Pack (Get Healthy: –2, Eat: +2)
Get Weapon (NewWeapons: -1, Get Healthy: +8
plus Kill Enemy action will have no impact on Health)

Goal-Oriented Planning

- Beyond single-step decisions



Working out Example with some non-binary variables:

Goals with Insistence Values:

Eat = 7, Kill Enemy = 8, Get Healthy = 4, NewWeapons=1

Actions with Impact on Insistence Values of Goals:

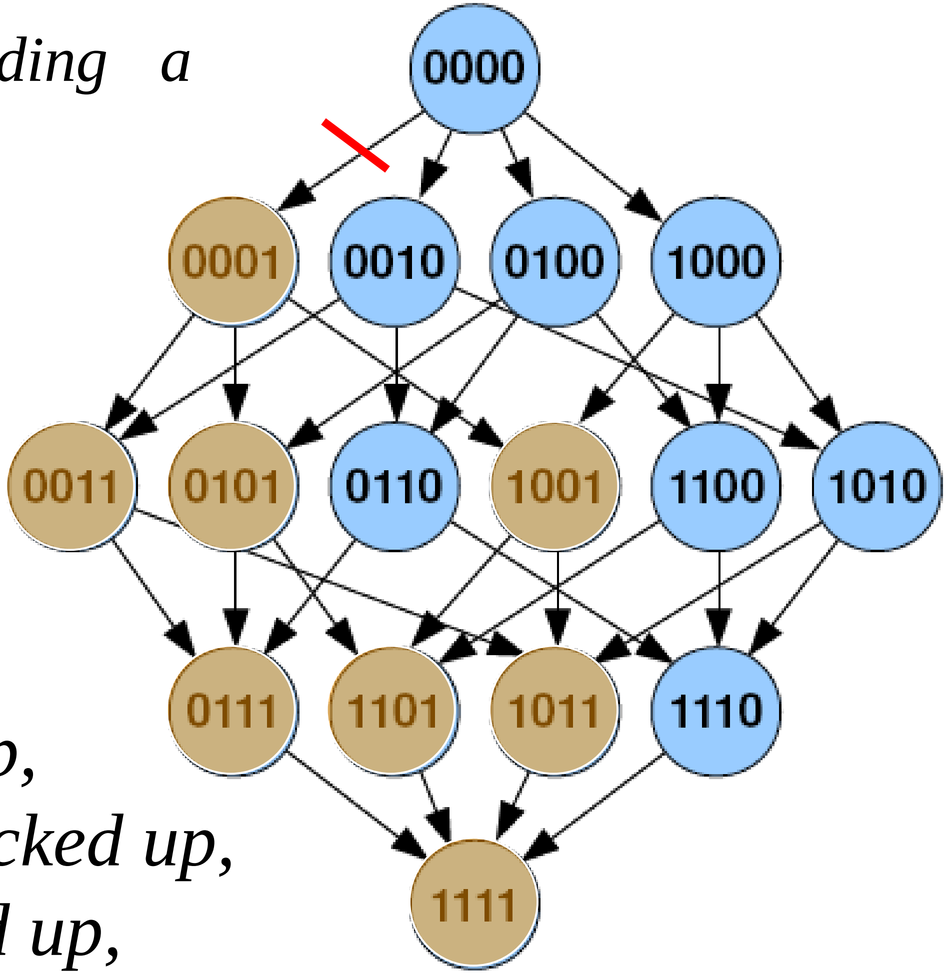
Get Food (Eat: -5, Get Healthy: +2)
Kill Enemy (Kill Enemy: -8, Get Healthy: +7)
Get Health Pack (Get Healthy: –2, Eat: +2)
Get Weapon (NewWeapons: -1, Get Healthy: +8
plus Kill Enemy action will have no impact on Health)

Goal-Oriented Planning

- Beyond single-step decisions

How to search the graph?

Search-space (graph) for finding a partially-defined goal:



Example:

initial state is 0000

goal state is 1111

(e.g., bit 0: food is picked up,

bit 1: health pack is picked up,

bit 2: weapon is picked up,

bit 3: enemy is shot)

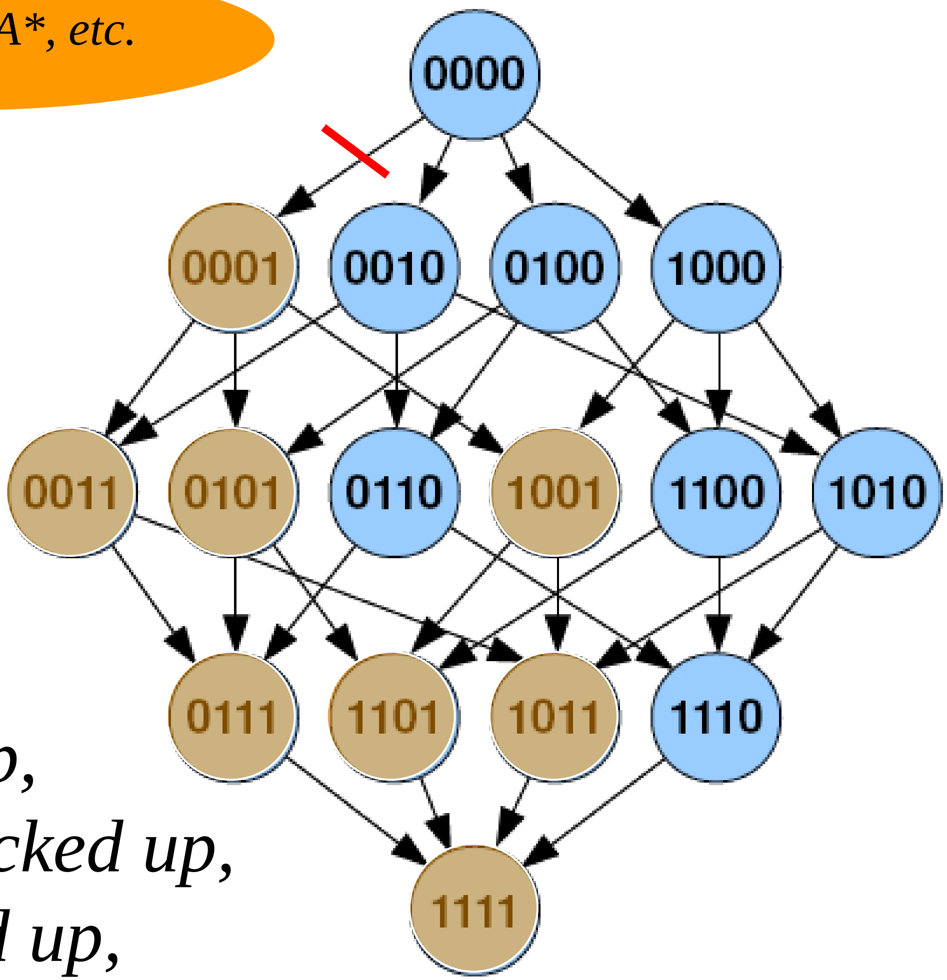
Goal-Oriented Planning

- Beyond single-step decisions

How to search the graph?

DFS, BFS, A, etc.*

Search-space (grap
partially-defined goal:



Example:

initial state is 0000

goal state is 1111

(e.g., bit 0: food is picked up,

bit 1: health pack is picked up,

bit 2: weapon is picked up,

bit 3: enemy is shot)

Goal-Oriented Planning

- Beyond single-step decisions

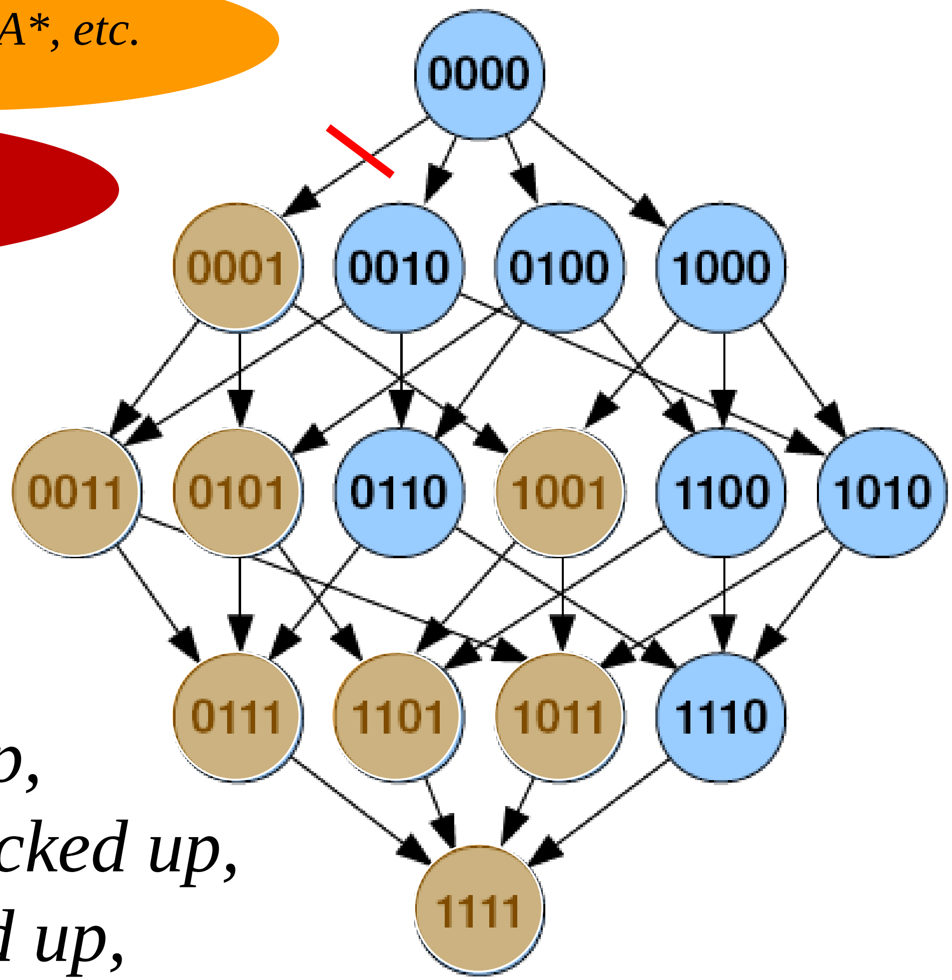
How to search the graph?

DFS, BFS, A, etc.*

Search-space (grap

partially

*Problems with BFS and A**
for large state vectors?



Example:

initial state is 0000

goal state is 1111

(e.g., bit 0: food is picked up,

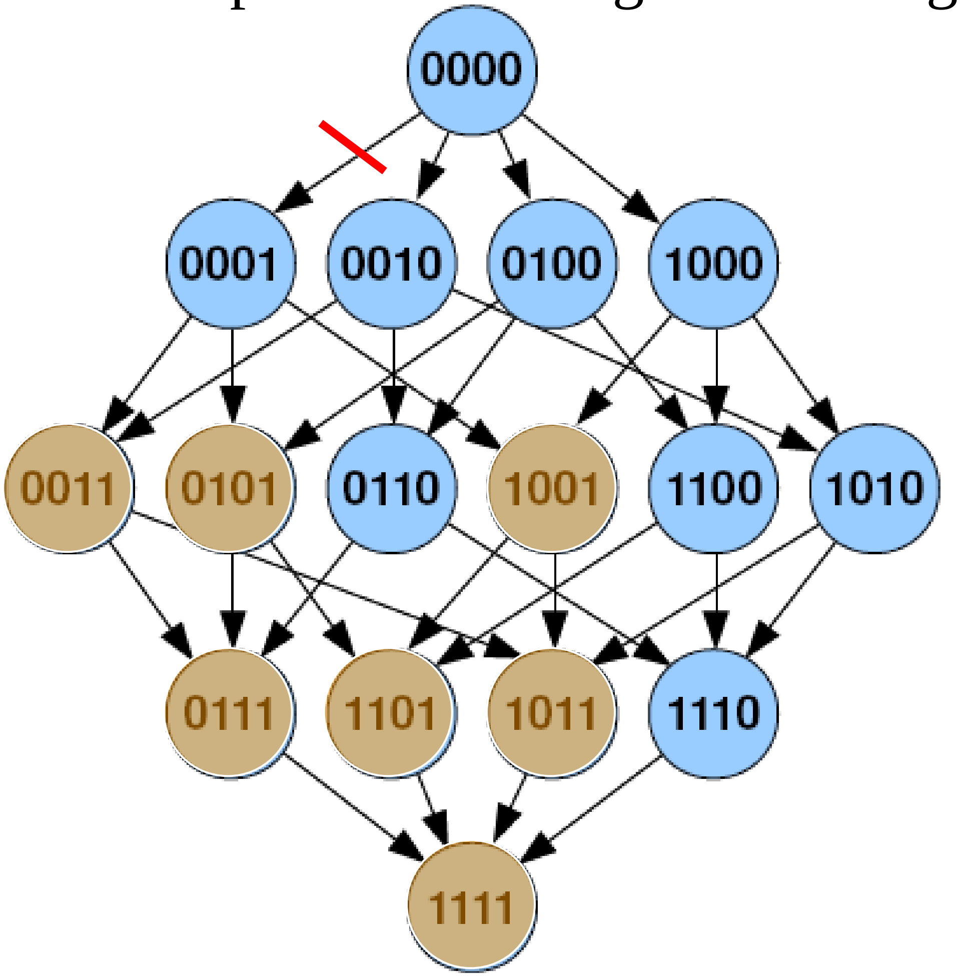
bit 1: health pack is picked up,

bit 2: weapon is picked up,

bit 3: enemy is shot)

Goal-Oriented Planning

- Beyond single-step decisions
- **IDA***: Very popular search for state-spaces with large branching factors and shallow goals



Goal-Oriented Planning

- Beyond single-step decisions
- **IDA***: Very popular search for state-spaces with large branching factors and shallow goals

IDA* (Iterative Deepening A*)

1. set $f_{max} = 1$ (or some other small value)
2. traverse the graph in DFS fashion without expanding states with $f > f_{max}$
3. If path to the goal found, return the best path it finds
4. Otherwise $f_{max} = f_{max} + 1$ and go to step 2

Goal-Oriented Planning

- Beyond single-step decisions
- **IDA***: Very popular search for state-spaces with large branching factors and shallow goals

IDA* (Iterative Deepening A*)

1. set $f_{max} = 1$ (or some other small value)
2. traverse the graph in DFS fashion without expanding states with $f > f_{max}$
3. If path to the goal found, return the best path it finds
4. Otherwise $f_{max} = f_{max} + 1$ and go to step 2

Proof?

- Complete and optimal in any state-space (with positive costs)
- Memory: $O(bl)$, where b – max. branching factor, l – length of optimal path
- Complexity: $O(kb^l)$, where k is the number of times DFS is called

Advanced Decision-making Mechanisms for this Class

- More on Behavior Trees
- Planning to Achieve the Goal
- Planning with Uncertainty to Achieve the Goal

Goal-Oriented Planning Under Uncertainty

- Dealing with uncertainty in outcomes

How do we represent this?

Example:

Suppose the character is under attack and can pick a weapon that allows it to more effectively shoot the enemies

Suppose also there is 50% chance of getting health pack at each attempt

Goals with Insistence Values:

Eat = 7, Kill Enemy = 8, Get Healthy = 4, NewWeapons=1

Actions with Impact on Insistence Values of Goals:

Get Food (Eat: -5, Get Healthy: +2)

Kill Enemy (Kill Enemy: -8, Get Healthy: +7)

Get Health Pack (Get Healthy: –2, Eat: +2)

Get Weapon (NewWeapons: -1, Get Healthy: +8 plus Kill Enemy action will have no impact on Health)

Goal-Oriented Planning Under Uncertainty

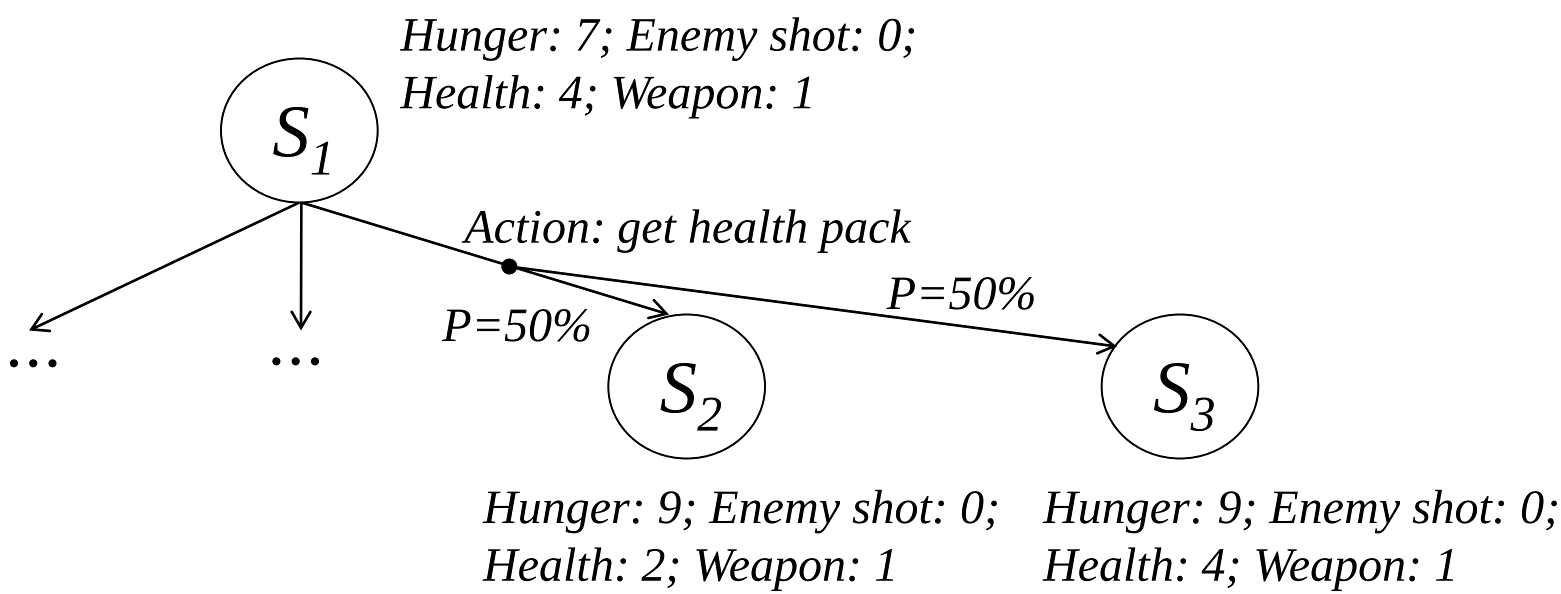
- Dealing with uncertainty in outcomes

Markov Decision Processes (MDP)

Example:

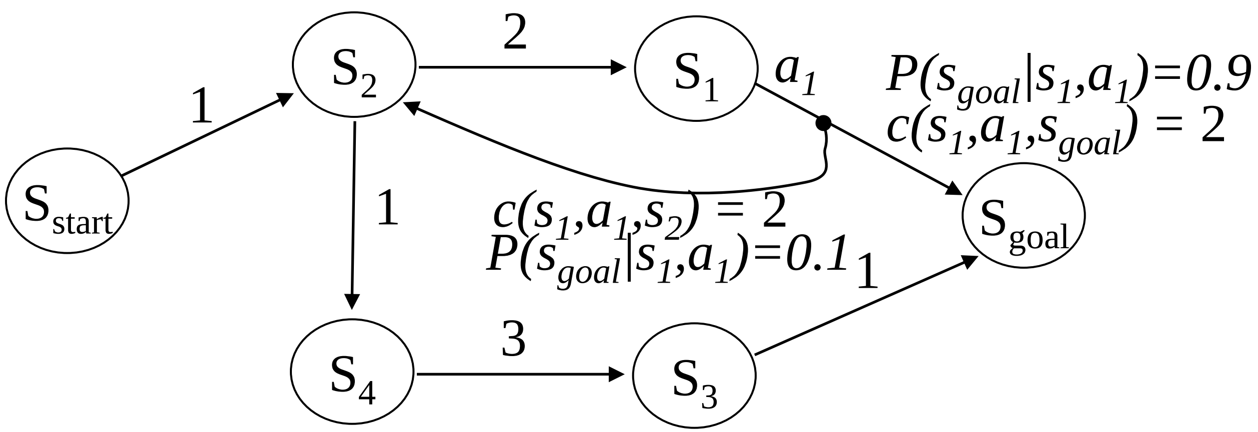
Suppose the character is under attack and can pick a weapon that allows it to more effectively shoot the enemies

Suppose also there is 50% chance of getting health pack at each attempt



Planning in MDPs

- What plan to compute?
  - Plan that minimizes the worst-case scenario (minimax plan)
  - Plan that minimizes the expected cost
  - Plan that minimizes cost while guaranteeing *P(goal reached) > t*

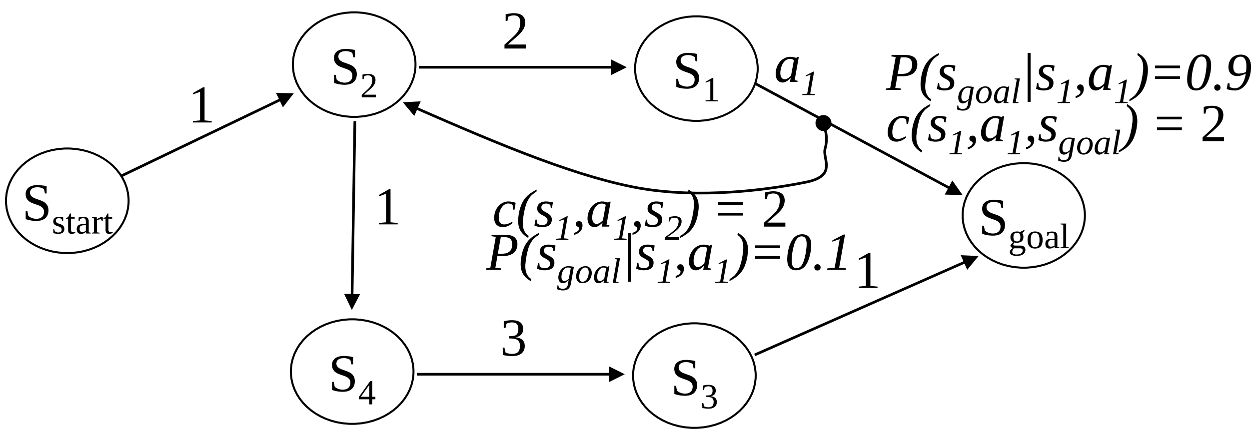


- Without uncertainty, plan is a single path:
  a sequence of states (a sequence of actions)
- In MDPs, plan is a policy π:
  mapping from a state onto an action

Planning in MDPs

- What plan to compute?
  - Plan that minimizes the worst-case scenario (minimax plan)
  - Plan that minimizes the expected cost
  - Plan that minimizes cost while guaranteeing *P(goal reached) > t*

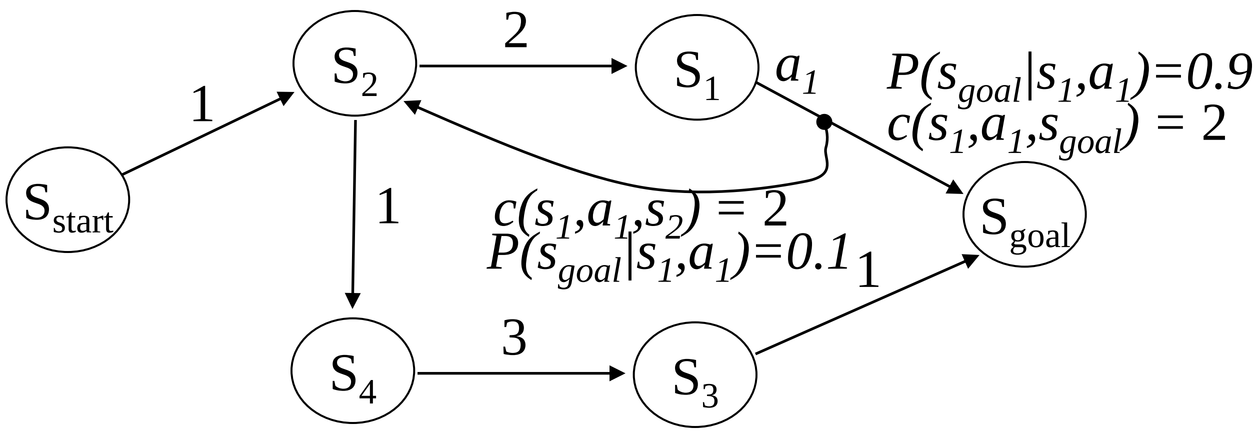
Which ones are policies?



- Without uncertainty, plan is a single path:
  a sequence of states (a sequence of actions)
- In MDPs, plan is a policy π:
  mapping from a state onto an action

Why?

Minimax Formulation



- Optimal policy π^*:

  minimizes the *worst* cost-to-goal

  $\pi^* = argmin_{\pi}\ max_{outcomes\ of\ \pi}\{cost\text{-}to\text{-}goal\}$

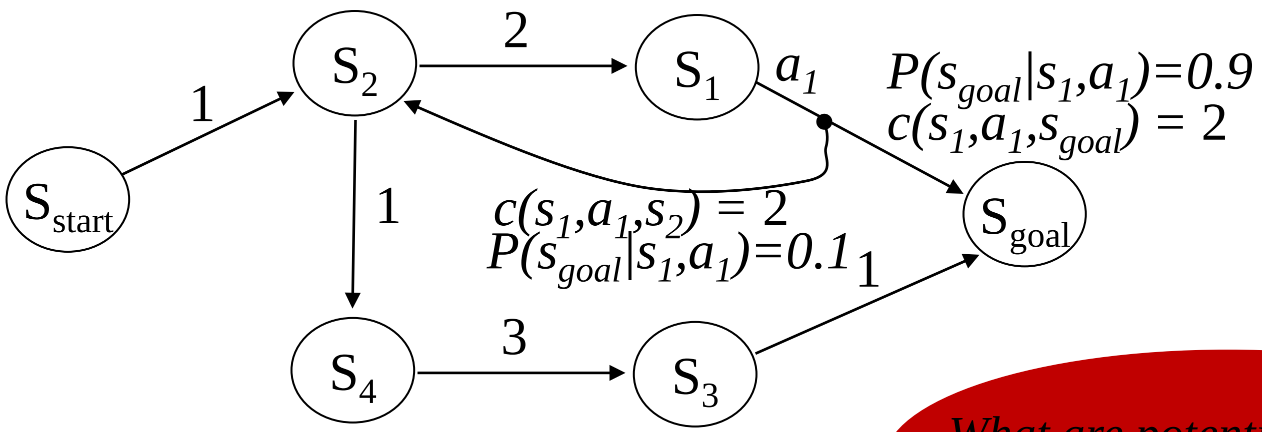
- worst cost-to-goal for $\pi_1=(s_{start},s_2,s_4,s_3,s_{goal})$ is:

  1+1+3+1 = 6

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

  1+2+2+2+2+2+2 + … = ∞

Minimax Formulation

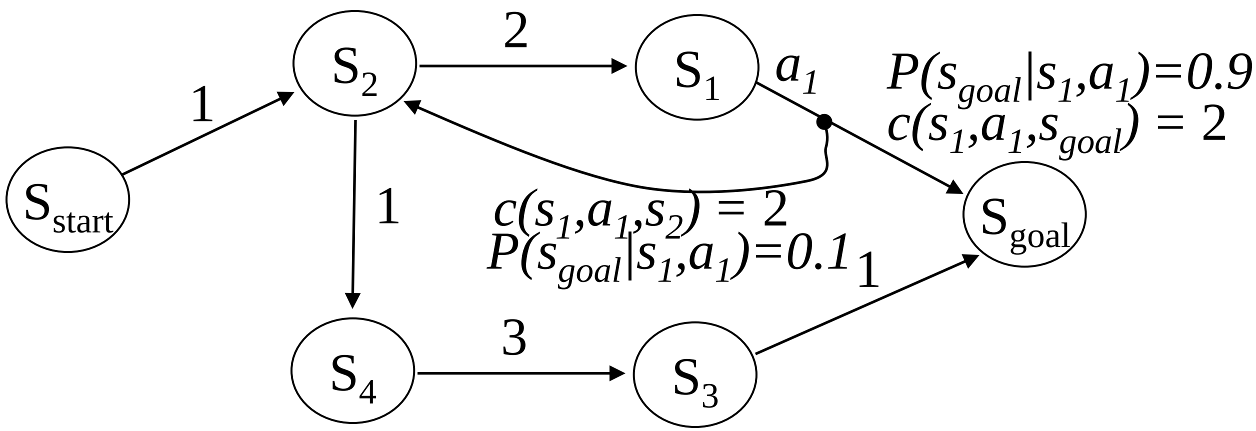


What are potential problems with minimax approaches?

- Optimal policy π^*:
  minimizes the *worst* cost-to-goal
  $\pi^* = argmin_{\pi}\ max_{outcomes\ of\ \pi}\{cost\text{-}to\text{-}goal\}$

- Optimal minimax policy $\pi^* = \pi_1 = (s_{start}, s_2, s_4, s_3, s_{goal})$

Expected Cost Formulation



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

expectation over outcomes

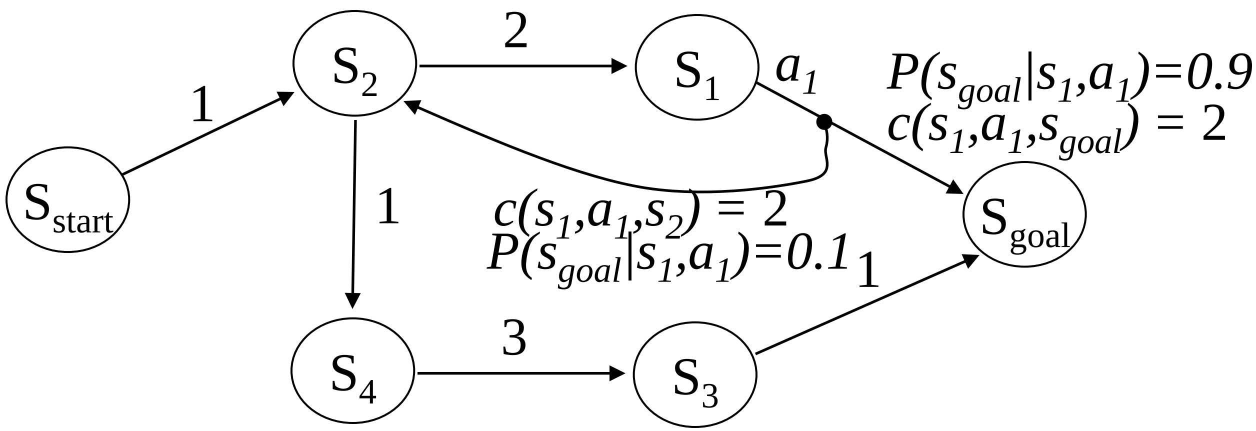
- expected cost-to-goal for $\pi_1=(s_{start},s_2,s_4,s_3,s_{goal})$ is

  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

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

expectation over outcomes

- expected cost-to-goal for $\pi_1=(s_{start},s_2,s_4,s_3,s_{goal})$ is

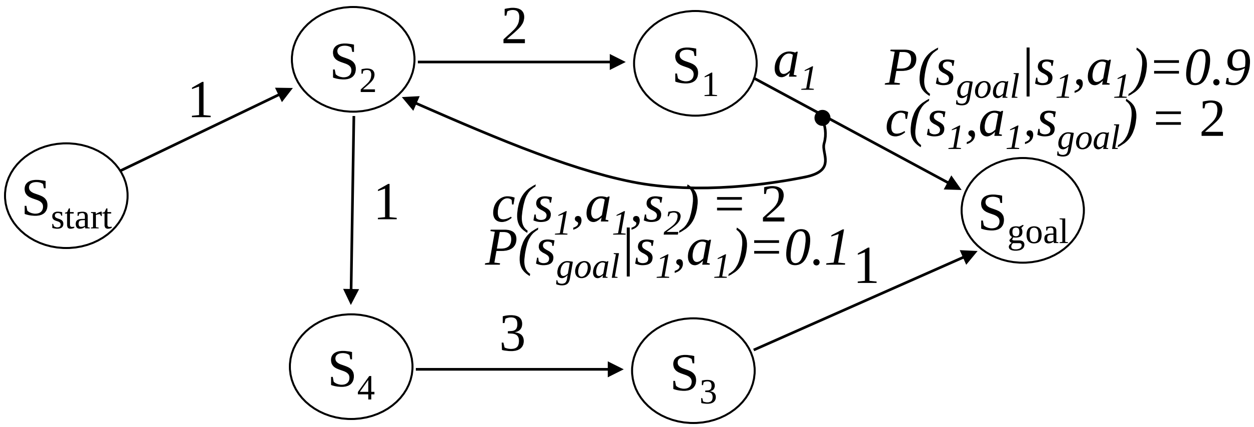
How to compute it?

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) + …=$5.4\overline{4}$

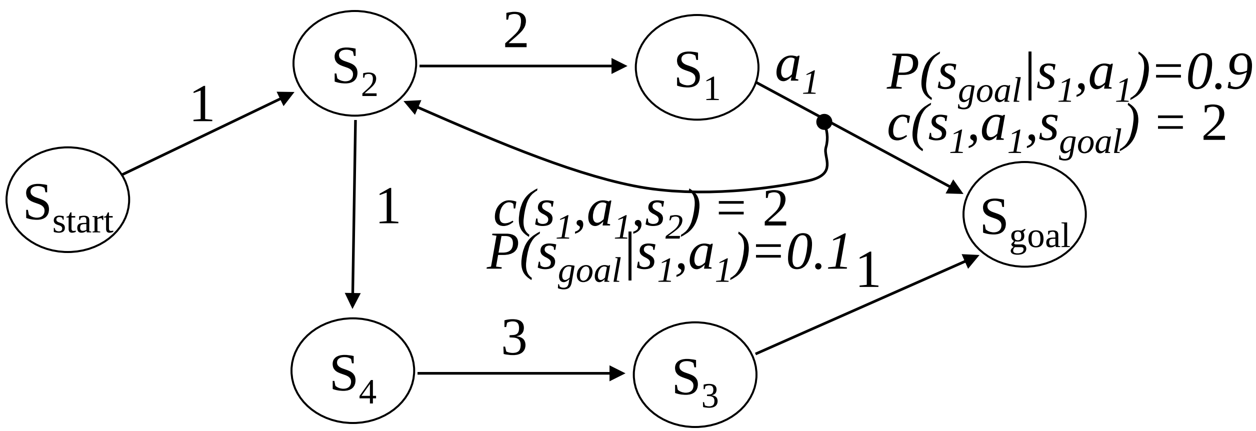
Expected Cost Formulation



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

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

Minimizing Cost with *P(success)>t* constraint



- Optimal path π^* is a path that

  minimizes the cost-to-goal assuming some outcomes

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

  s.t. $P(assumed\ outcomes) > t$

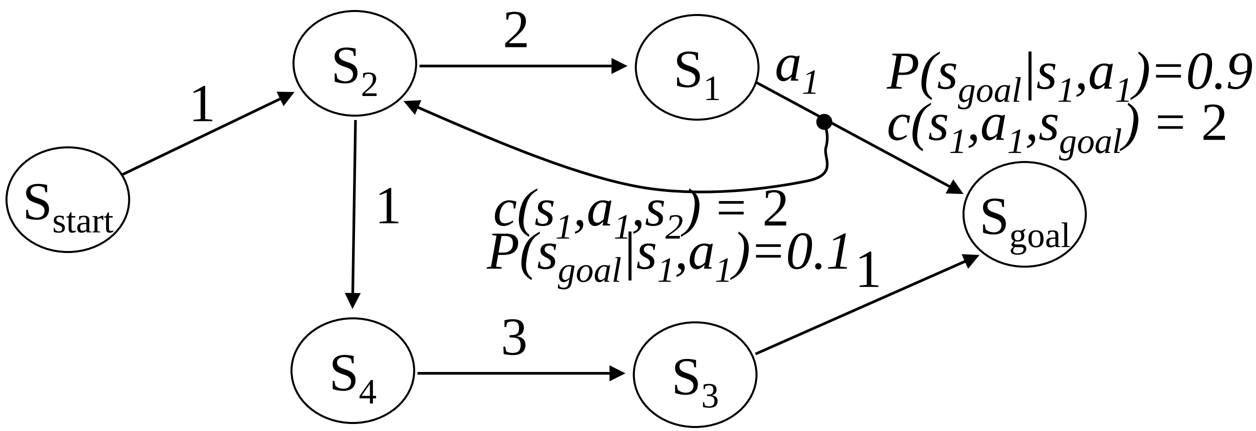
- for $\pi_1=(s_{start},s_2,s_4,s_3,s_{goal})$:

  cost-to-goal = 1+1+3+1 = 6, $P(success) = 1$

- for π_2=(try to go through s_1) is:

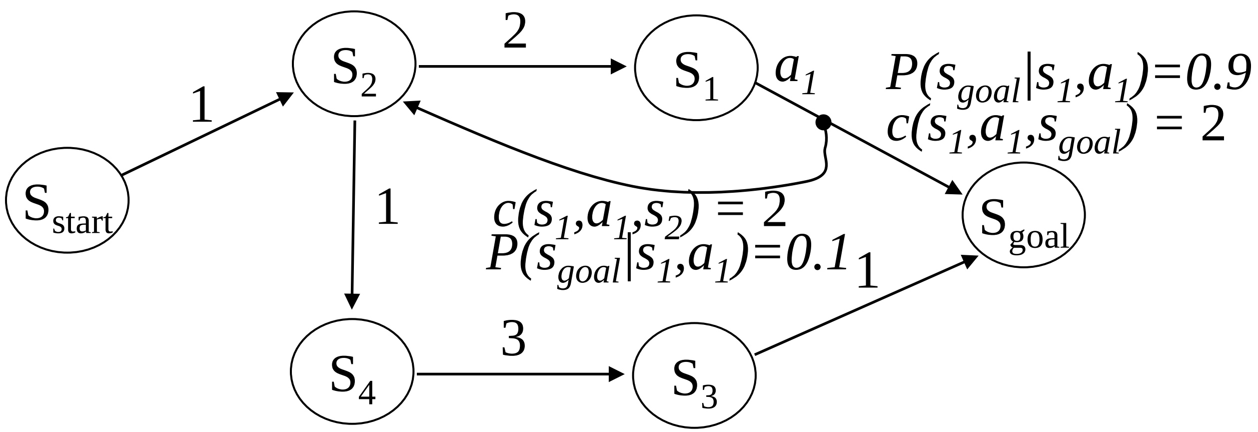
  cost-to-goal = 1+2+2 = 5, $P(success) = 0.9$

Minimizing Cost with *P(success)>t* constraint



- Optimal path π^* is a path that
  minimizes the cost-to-goal assuming some outcomes
  $\pi^* = argmin_\pi \{cost\text{-}to\text{-}goal\}$
  s.t. $P(assumed\ outcomes) > t$
- for $t > 0.9$, $\pi^* = \pi_1 = (s_{start}, s_2, s_4, s_3, s_{goal})$
- for $t \leq 0.9$, $\pi^* = \pi_2 =$ (try to go through s_1)

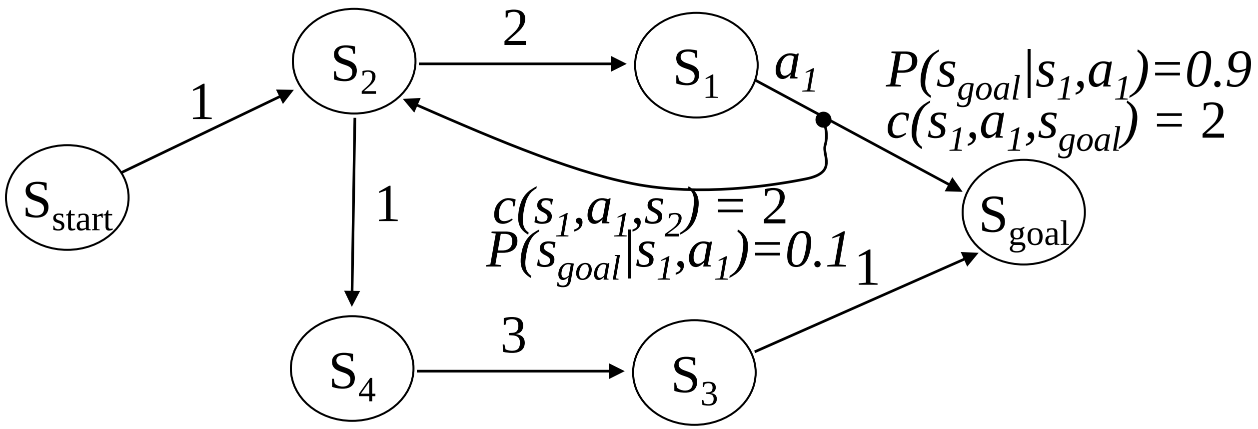
Computing Expected Cost Minimal Plans



- 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



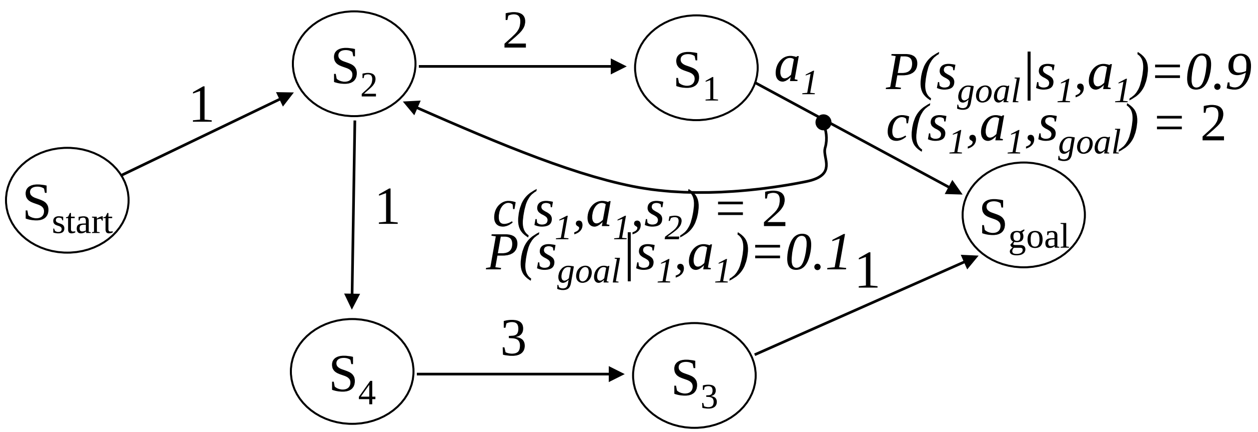
- Optimal policy π^*:

  $\pi^*(s) = argmin_a\ y \qquad (s,a,s')+v^*(s')\}$

  (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:

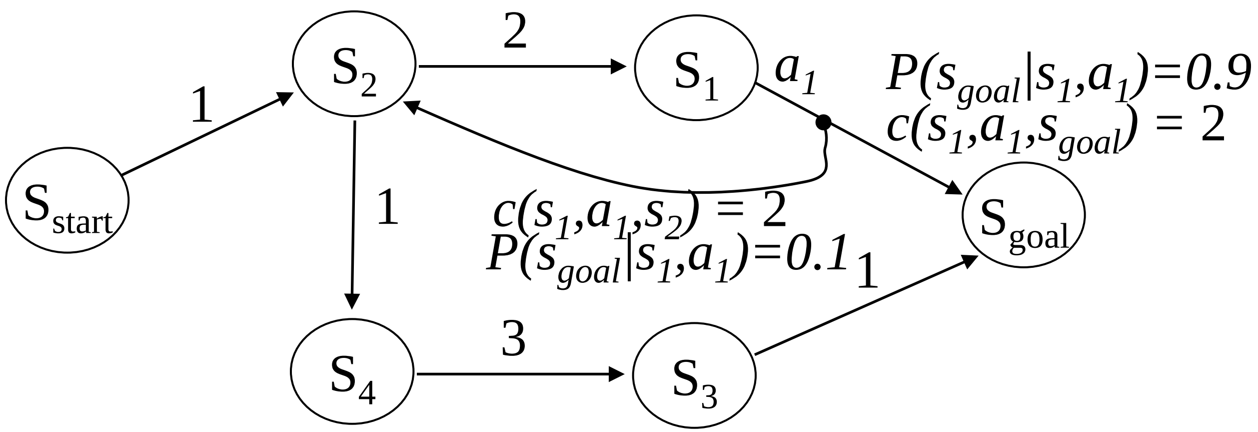
$v^*(s_{goal})=0$

$v^*(s) = \min_a y \quad (s,a,s')+v^*(s'))$ 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):

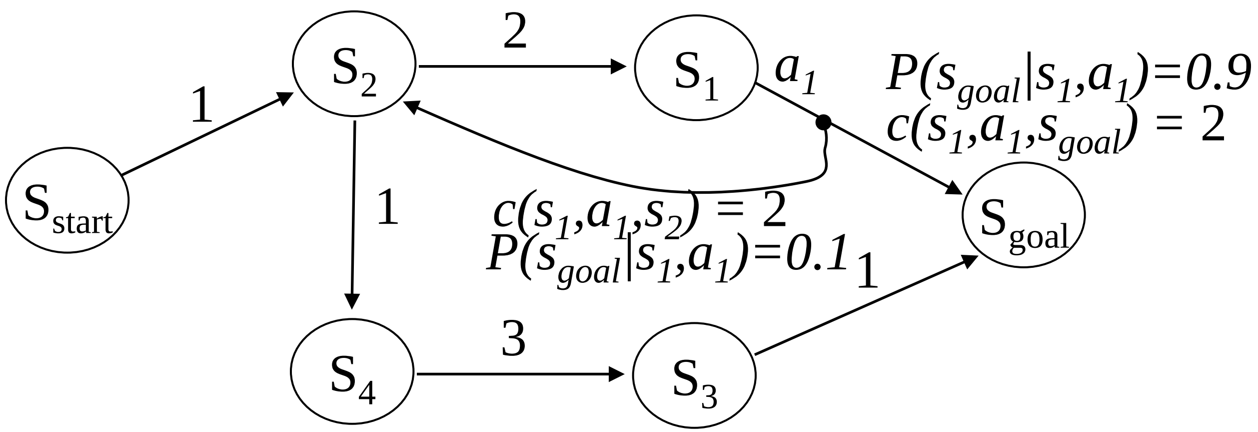
  Initialize *v*-values of all states to finite values;

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

  $v(s_{goal}) = 0$

  $v(s) = min_a\, y \quad s,a,s')+v(s'))$ for any $s \neq s_{goal}$

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:

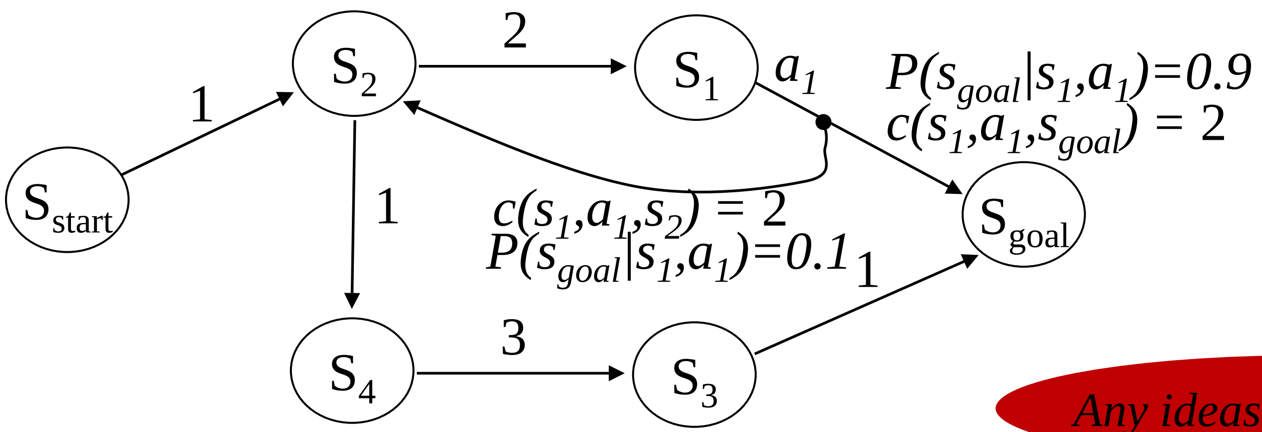
$v(s_{goal}) = 0$

$v(s) = min_a y \qquad s,a,s')+v(s'))$ for any $s \neq s_{goal}$

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

convergence time does depend a lot on iteration order

Computing Expected Cost Minimal Plans



Any ideas for the order?

- 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:

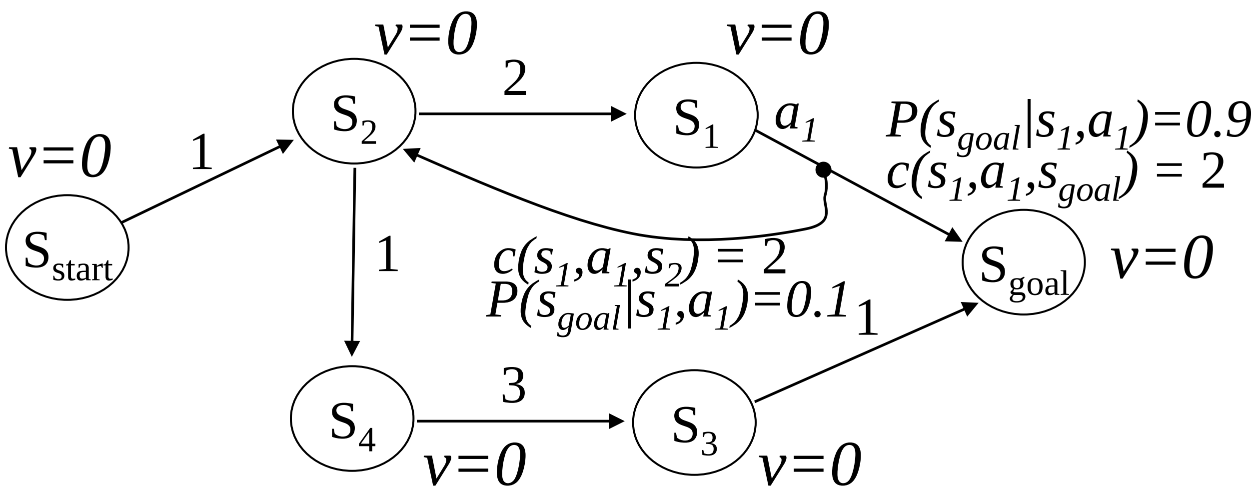
$v(s_{goal}) = 0$

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

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

convergence time does depend a lot on iteration order

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

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

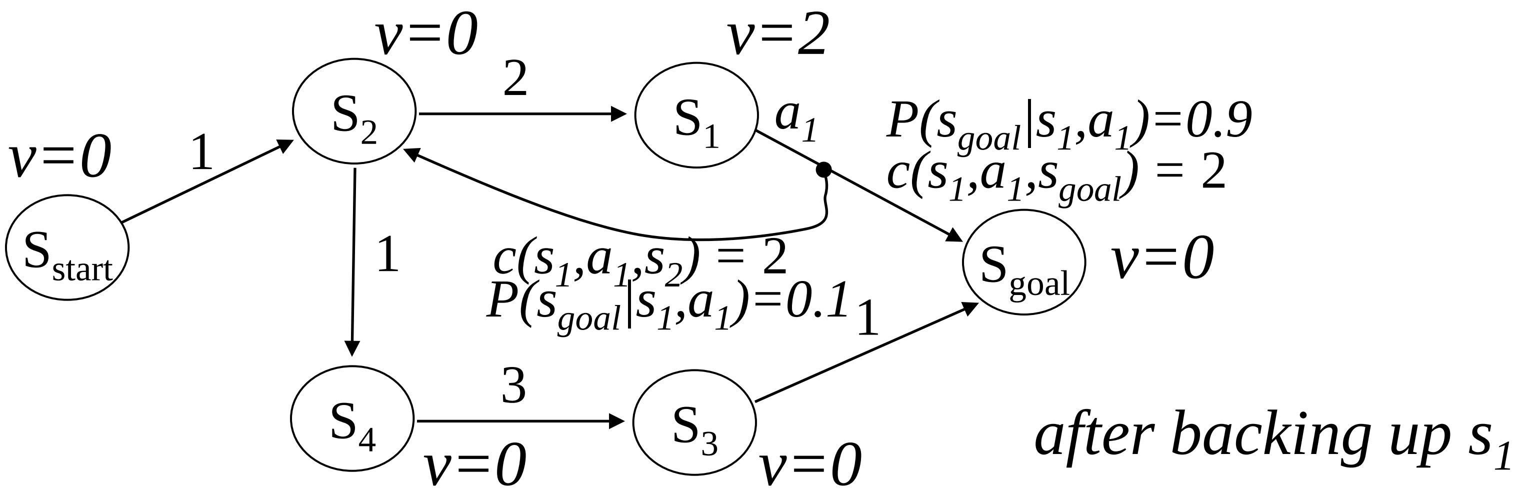
Iterate over all *s* in MDP and re-compute until convergence:

$v(s_{goal}) = 0$

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

Bellman update equation
(or backup)

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

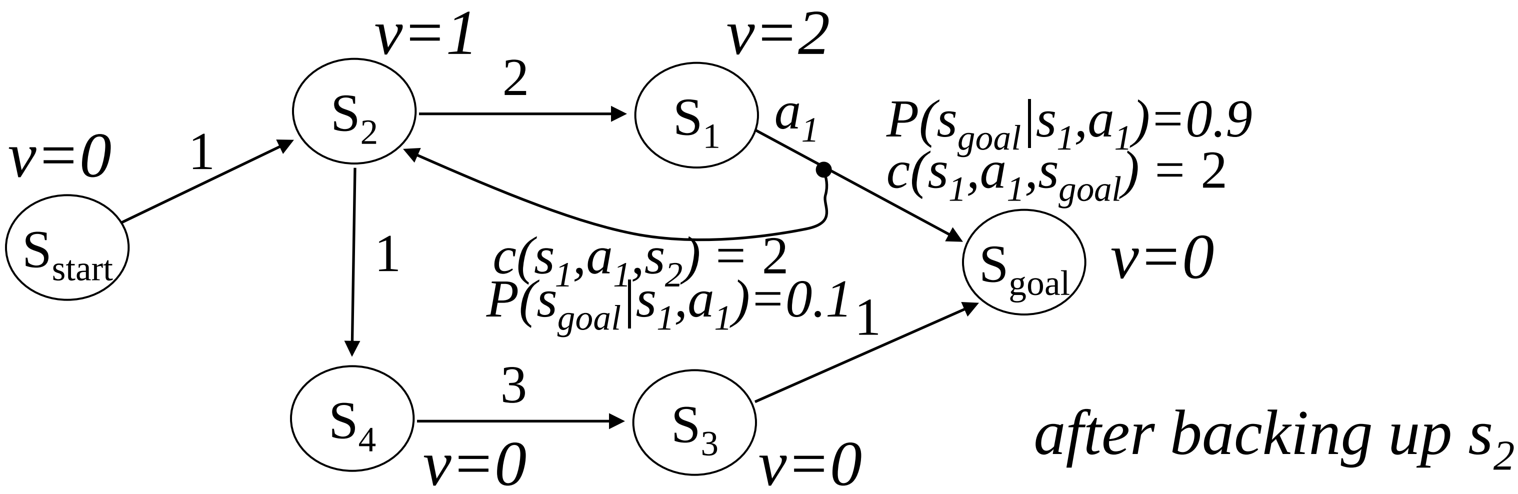
 Initialize *v*-values of all states to finite values;

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

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

$$v(s) = min_a\ y \quad s,a,s')+v(s'))$$ for any $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

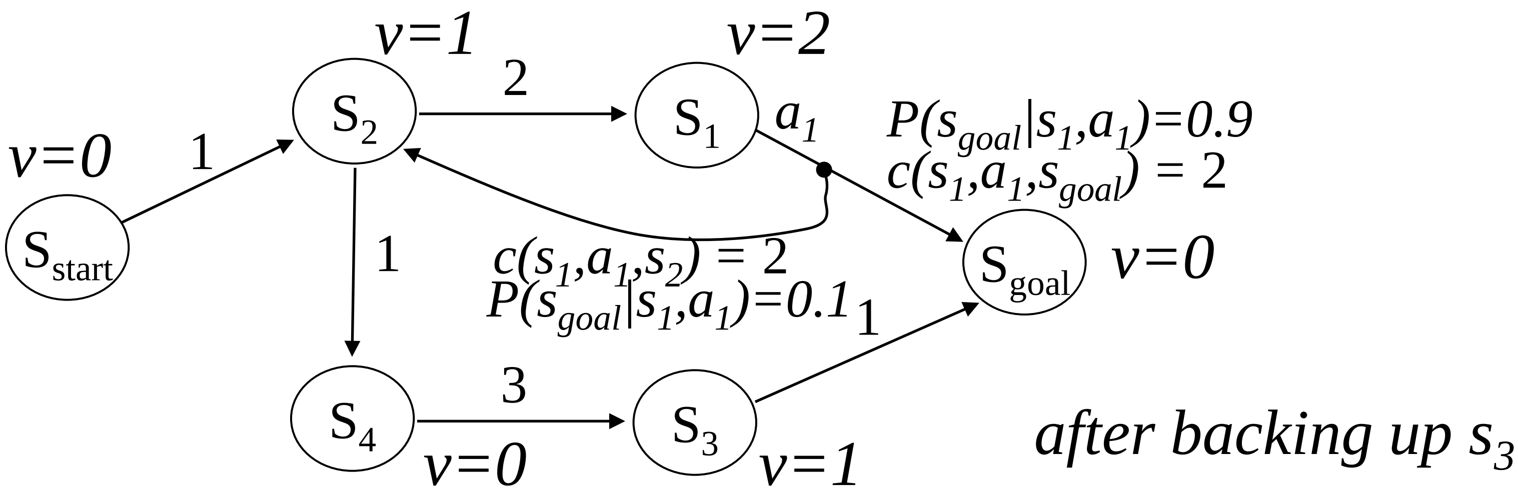
  Initialize *v*-values of all states to finite values;

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

  $v(s_{goal}) = 0$

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

Computing Expected Cost Minimal Plans



- Value Iteration (VI):

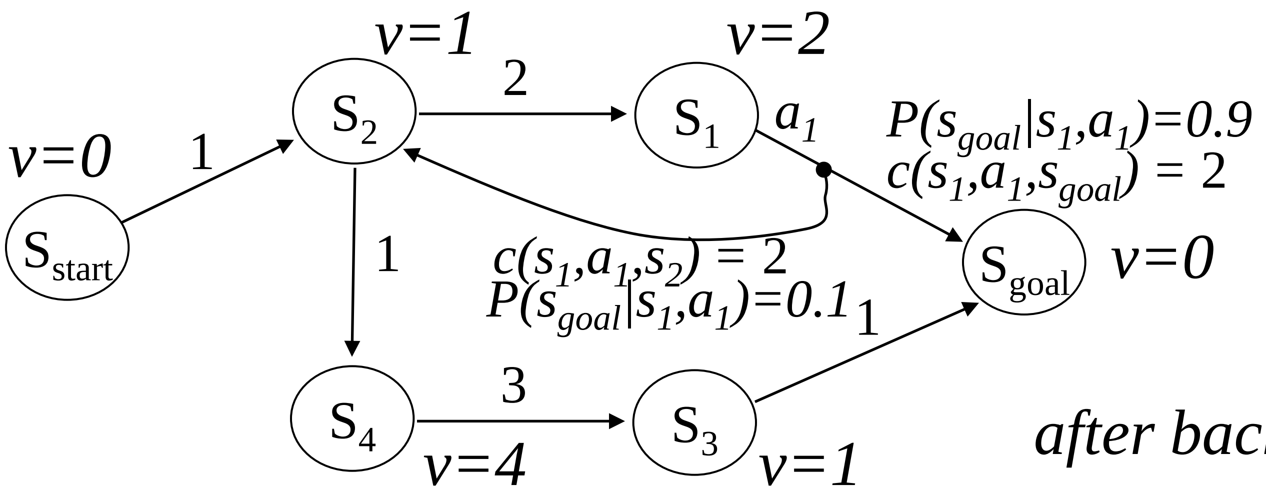
  Initialize *v*-values of all states to finite values;

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

  $v(s_{goal}) = 0$

  $v(s) = min_a\ y \quad s,a,s')+v(s'))$ for any $s \neq s_{goal}$

Computing Expected Cost Minimal Plans



after backing up s_4

- Value Iteration (VI):

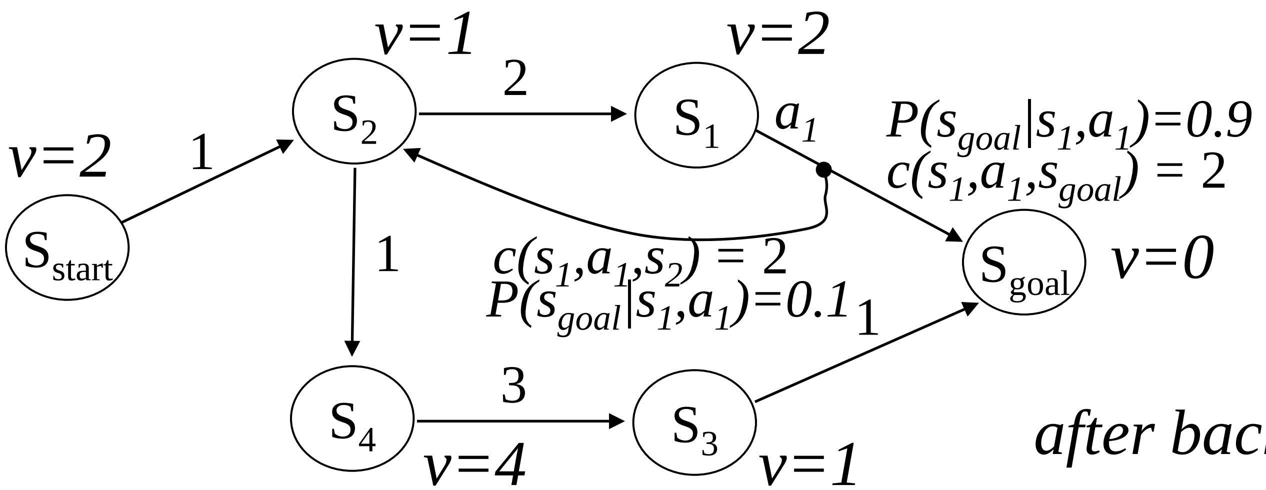
  Initialize *v*-values of all states to finite values;

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

  $v(s_{goal}) = 0$

  $v(s) = min_a\ y$ $s,a,s')+v(s'))$ *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:

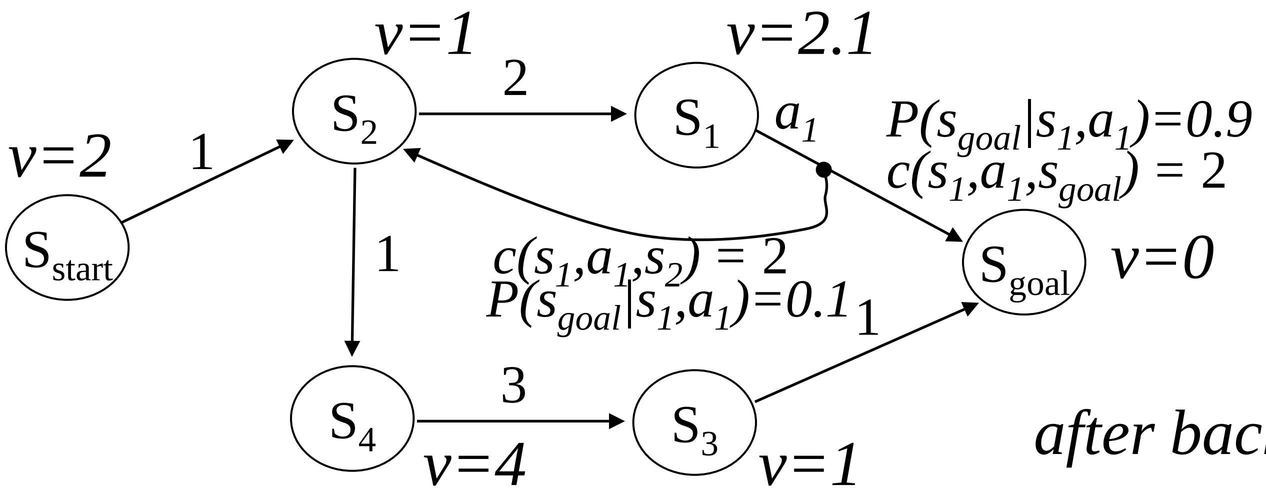
$v(s_{goal}) = 0$

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

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

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

Computing Expected Cost Minimal Plans



after backing up s_1

- Value Iteration (VI):

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

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

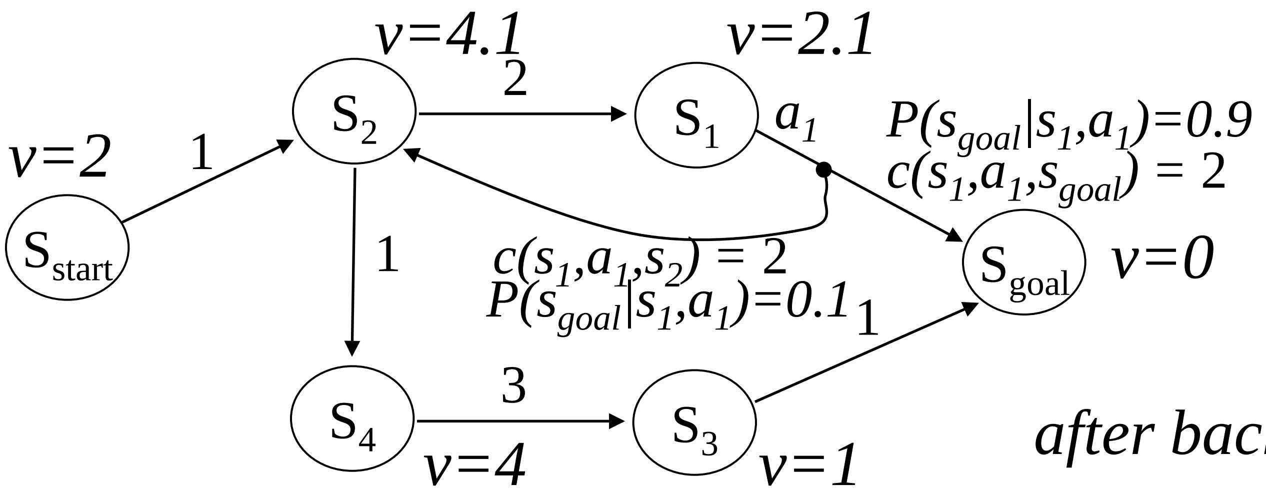
  $v(s_{goal}) = 0$

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

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

  Bellman error: $|v(s) - min_a\ y \quad s,a,s')+v(s')\}|$ *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:

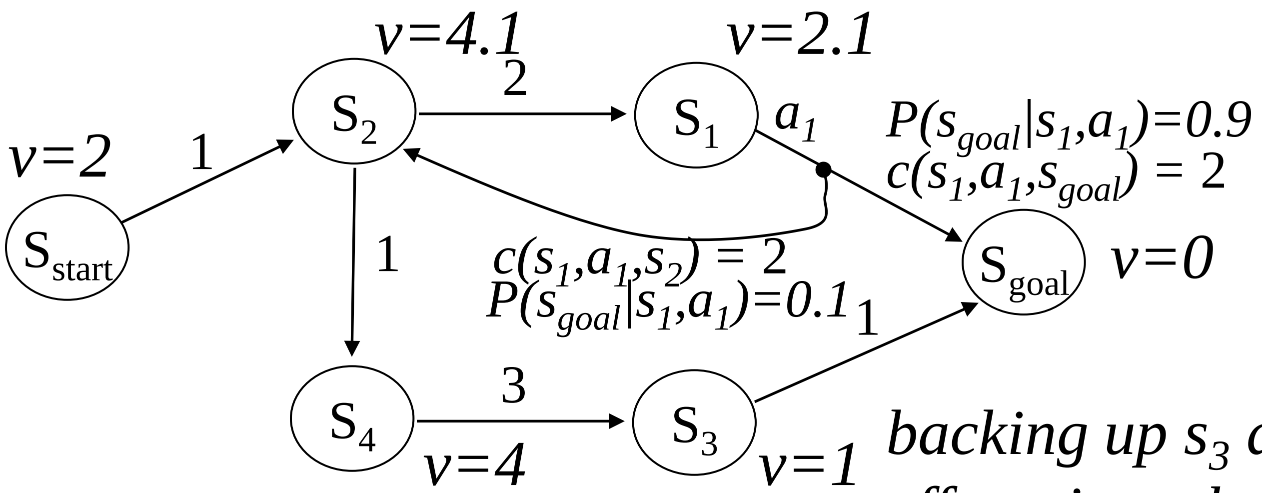
  $v(s_{goal}) = 0$

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

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

  Bellman error: $|v(s) - min_a\ y \quad s,a,s')+v(s'))|$ *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;

  How to select backups more effectively?

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

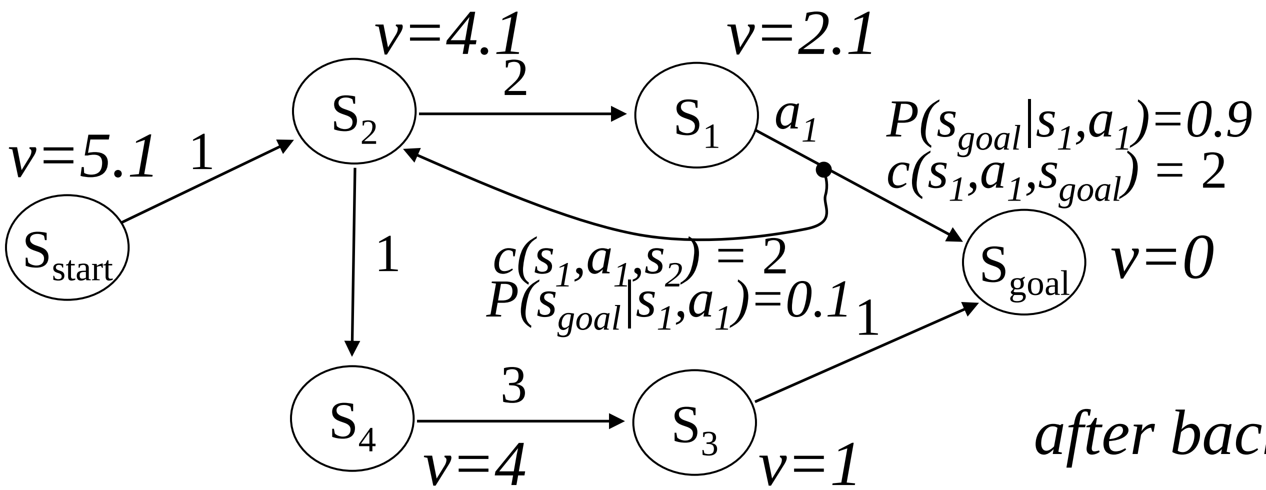
  $v(s_{goal}) = 0$

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

Usual convergence condition: Bellman error over all states < Δ

Bellman error: $|v(s) - min_a\ y \quad s,a,s')+v(s')\}|$ *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:

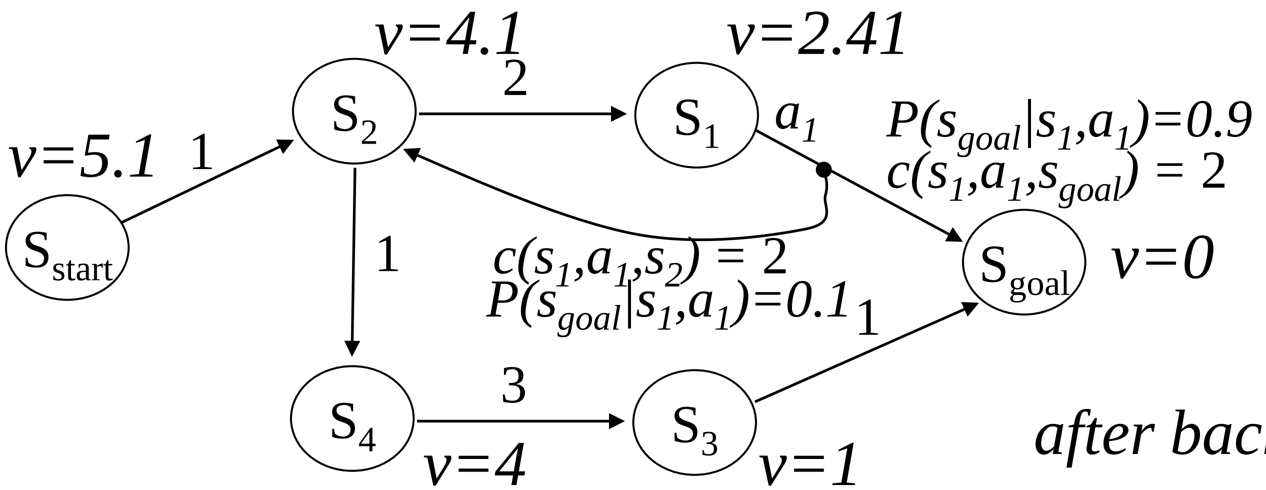
  $v(s_{goal}) = 0$

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

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

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

Computing Expected Cost Minimal Plans



after backing up s_1

- Value Iteration (VI):

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

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

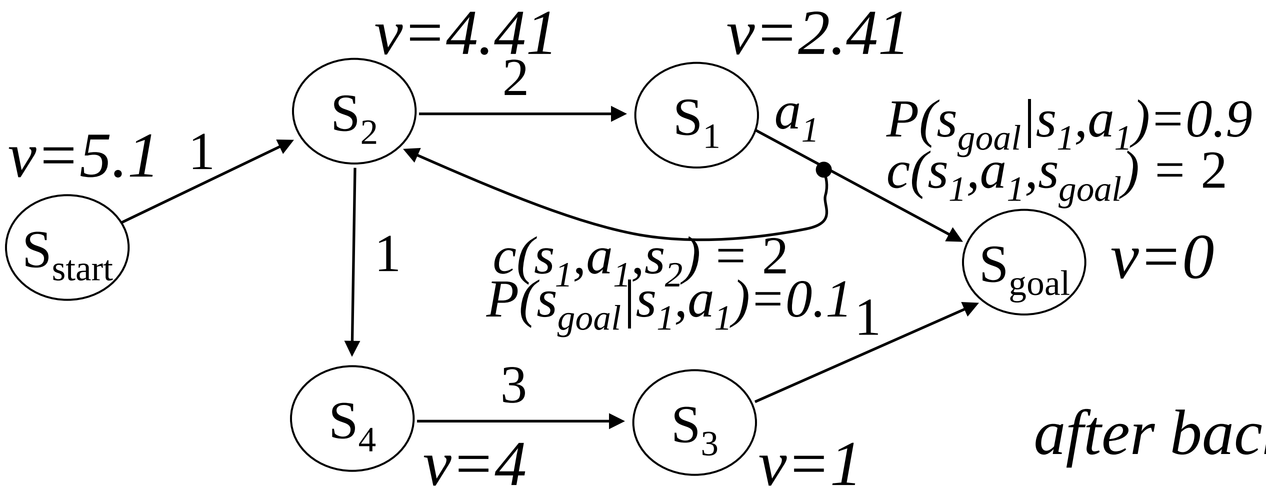
  $v(s_{goal}) = 0$

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

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

  Bellman error: $|v(s) - min_a\ y \qquad s,a,s')+v(s'))|$ *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:

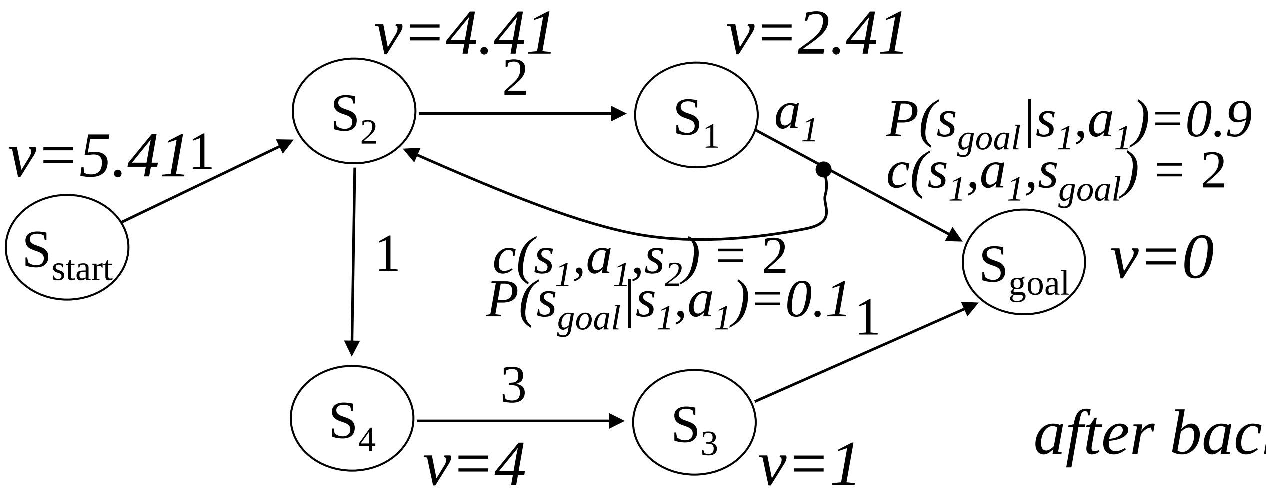
  $v(s_{goal}) = 0$

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

Usual convergence condition: Bellman error over all states < Δ

Bellman error: $|v(s)$ - $min_a\ y \qquad s,a,s')+v(s')\}|$ *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:

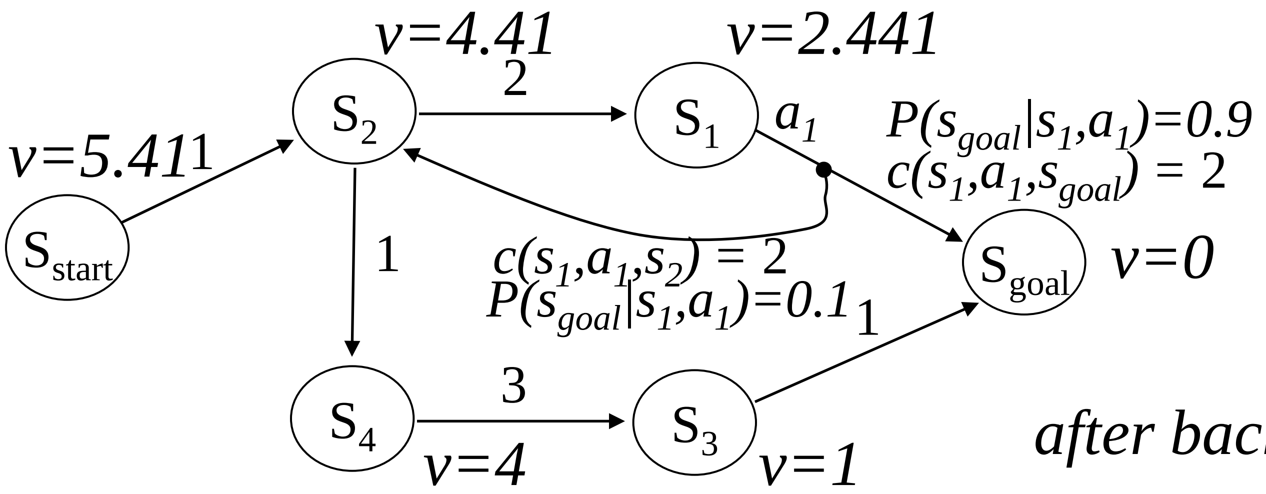
$v(s_{goal}) = 0$

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

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

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

Computing Expected Cost Minimal Plans



after backing up s_1

- Value Iteration (VI):

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

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

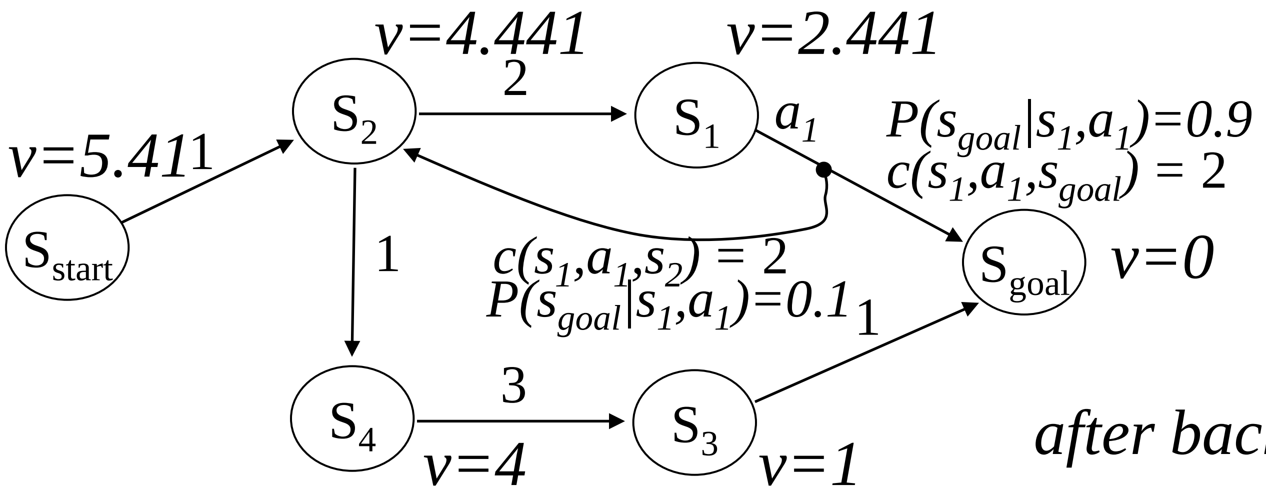
  $v(s_{goal}) = 0$

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

  Usual convergence condition: Bellman error over all states < Δ

  Bellman error: $|v(s) - min_a\ y$ $s,a,s')+v(s')\}|$ *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:

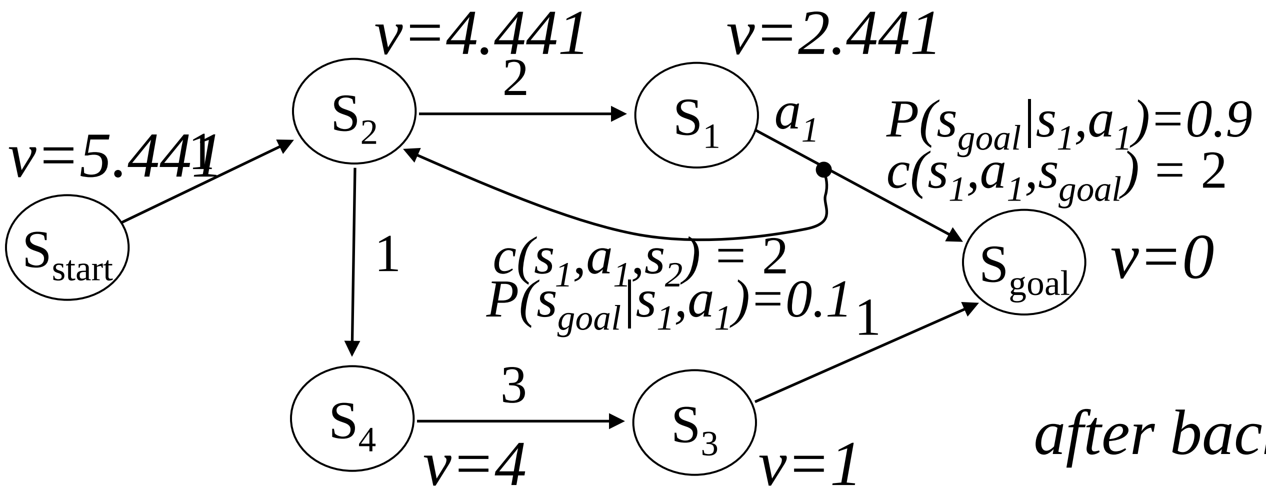
  $v(s_{goal}) = 0$

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

  Usual convergence condition: Bellman error over all states < Δ

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

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:

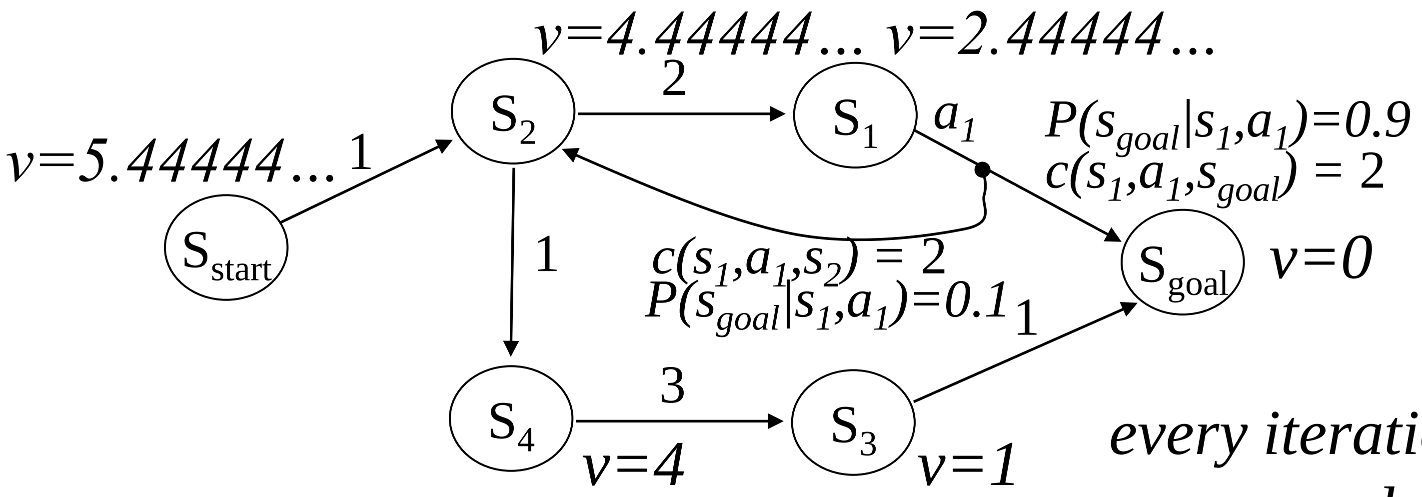
  $v(s_{goal}) = 0$

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

  Usual convergence condition: Bellman error over all states < Δ

  Bellman error: $|v(s) - min_a$ y 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:

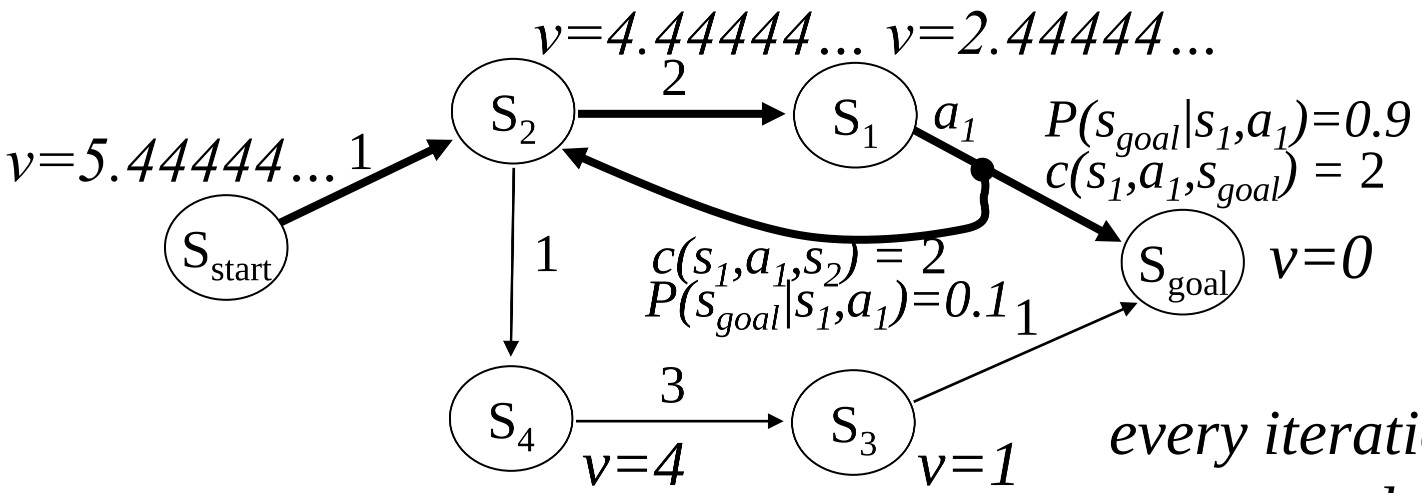
  $v(s_{goal}) = 0$

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

Usual convergence condition: Bellman error over all states < Δ

Bellman error: $|v(s) - \min_a y$ $s,a,s')+v(s')\}|$ *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
y (s,a,s')+v(s')}

Initialize v-values of all states to finite values;

re-compute until convergence:

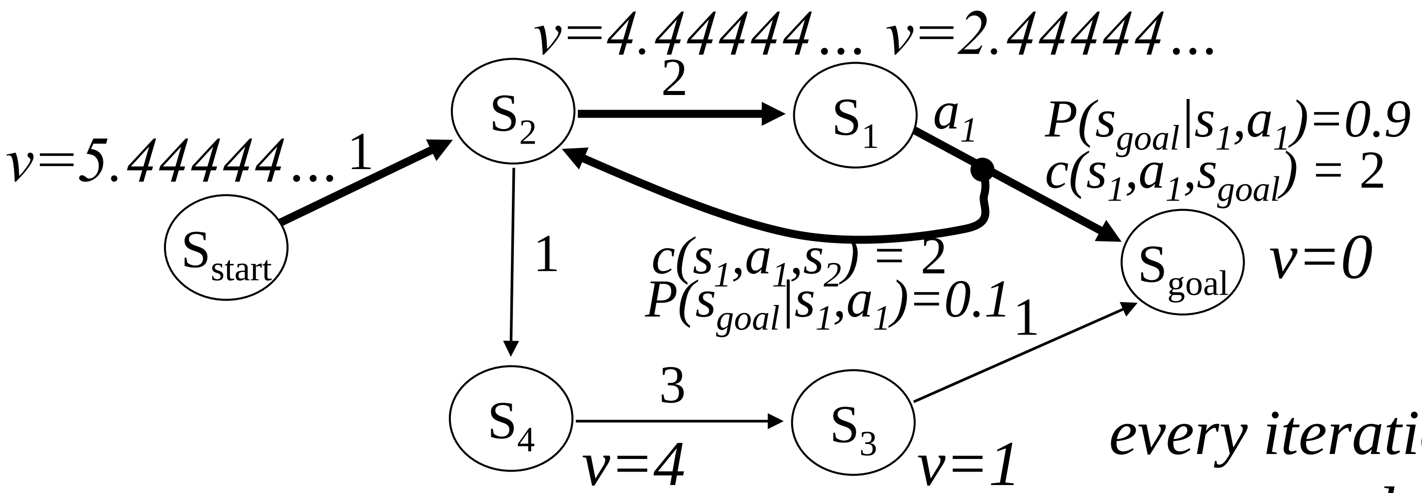
expected cost of executing greedy policy is at most:
$v^*(s_{start})c_{min}/(c_{min}-\Delta)$
where c_{min} is minimum edge cost

for any $s \neq s_{goal}$

Usual convergence condition: Bellman error over all states < Δ

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

Computing Expected Cost Minimal Plans



every iteration computes

…imal 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 finite values;

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

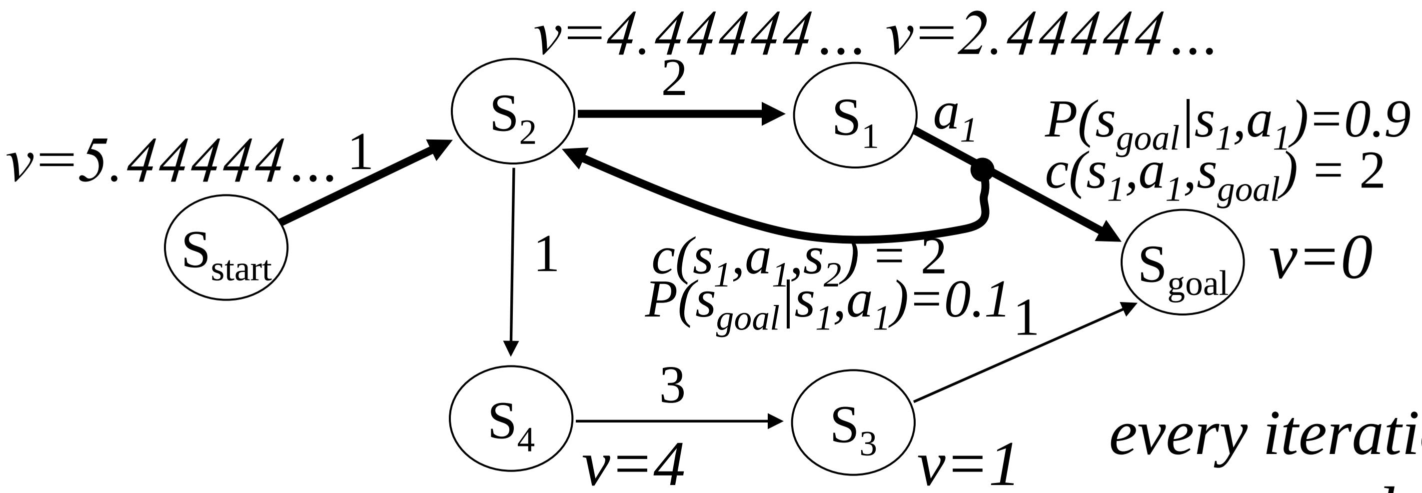
$v(s_{goal}) = 0$

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

Usual convergence condition: Bellman error over all states < Δ

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

Computing Expected Cost Minimal Plans



every iteration computes one more decimal point

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

- Value Iteration (VI):

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

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

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

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

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

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