

16-350

Planning Techniques for Robotics

*Planning under Uncertainty:
Minimax Formulation*

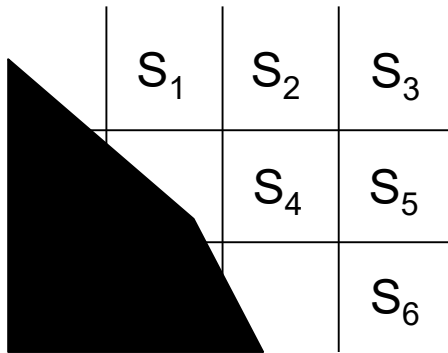
Maxim Likhachev

Robotics Institute

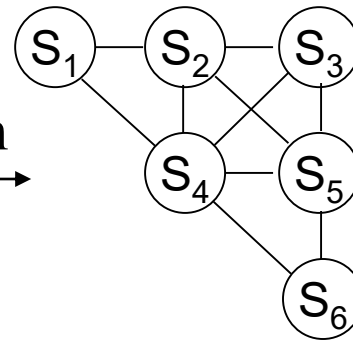
Carnegie Mellon University

Uncertainty in Robotics

- So far our planners assumed no uncertainty
 - execution is perfect



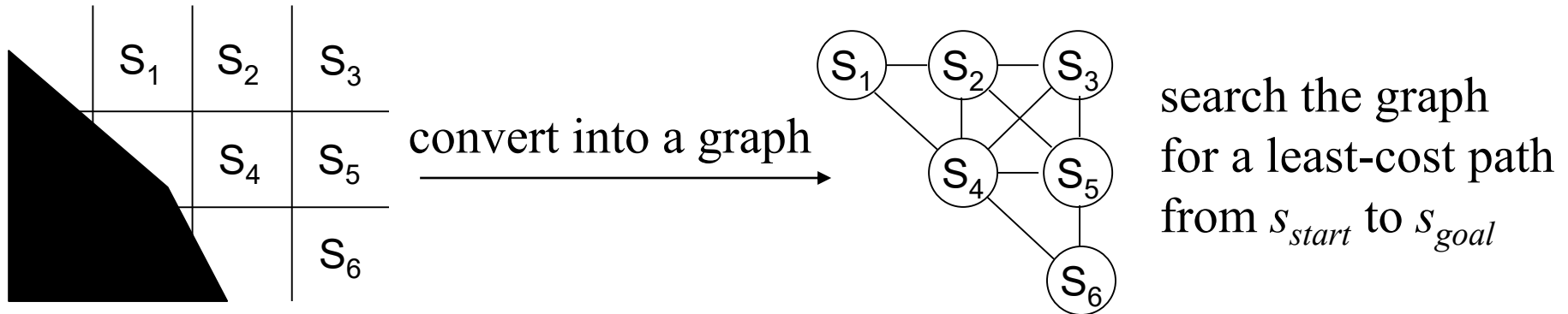
convert into a graph



search the graph
for a least-cost path
from s_{start} to s_{goal}

Uncertainty in Robotics

- So far our planners assumed no uncertainty
 - execution is perfect

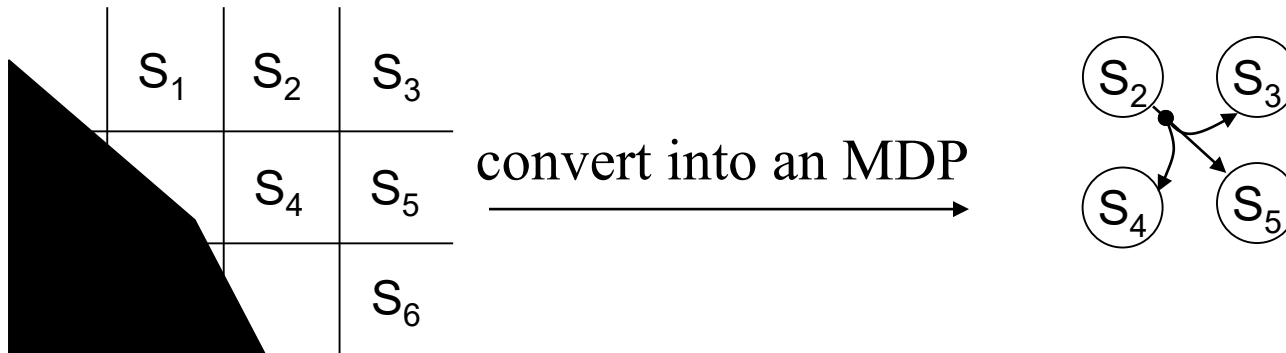


- Any deviations from the plan are dealt by re-planning
- Could be quite suboptimal and sometimes dangerous
 - planning a path along cliff does not take into account slippage
 - others examples???

Uncertainty in Robotics

- Modeling uncertainty in execution during planning

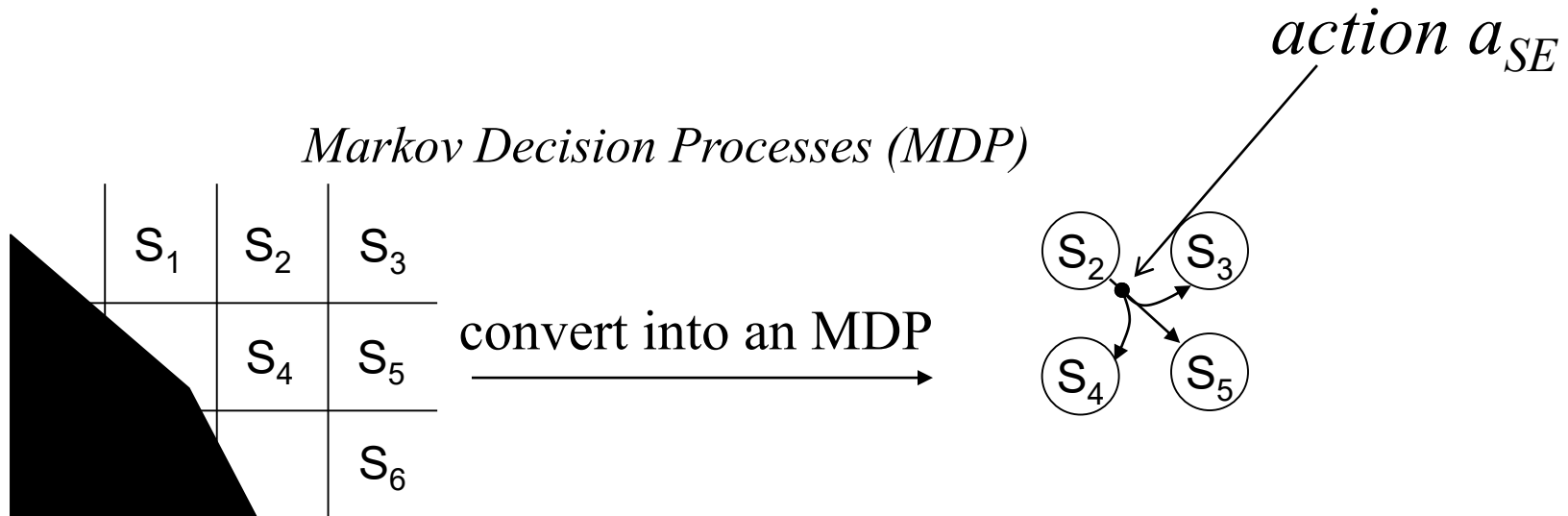
Markov Decision Processes (MDP)



- at least one action in the graph has more than one outcome
- each outcome is associated with probability and cost

Uncertainty in Robotics

- Modeling uncertainty in execution during planning



- at least one action in the graph has more than one outcome
- each outcome is associated with probability and cost

example: $s_3, s_4, s_5 \in \text{succ}(s_2, a_{SE})$,

$$P(s_5|a_{se},s_2) = 0.9, \quad c(s_2,a_{se},s_5) = 1.4$$

$$P(s_3|a_{se},s_2) = 0.05, \quad c(s_2,a_{se},s_3) = 1.0$$

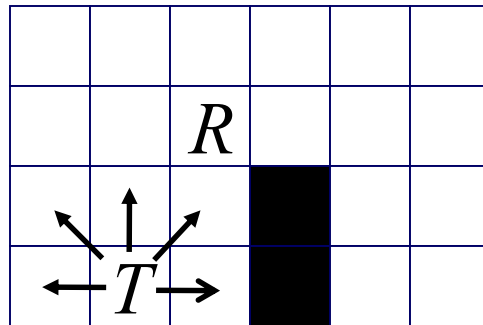
$$P(s_4|a_{se},s_2) = 0.05, \quad c(s_2,a_{se},s_4) = 1.0$$

Moving along Cliff Example

- Example on the board

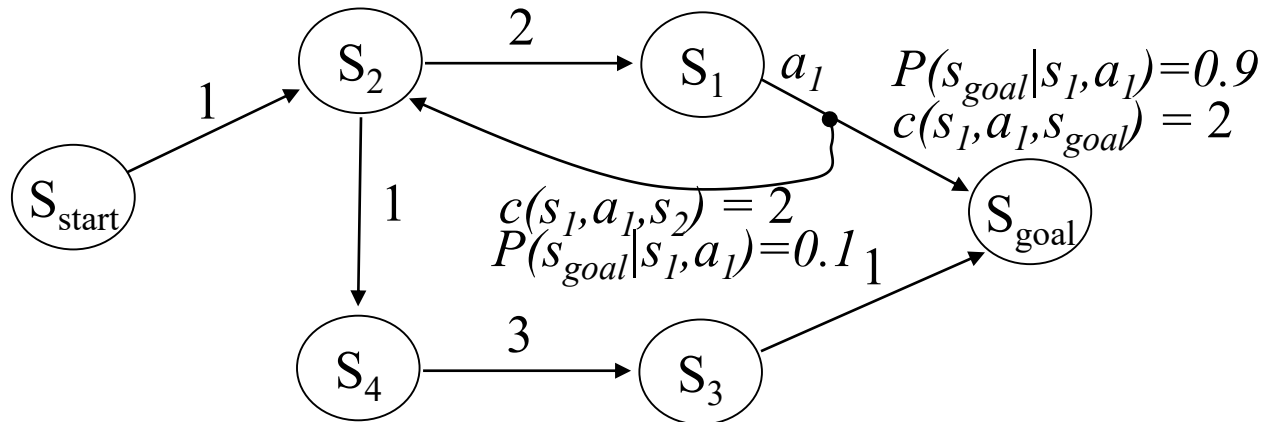
Moving-Target Search Example

- Uncertainty in the target moves
- What is a state-space and action space?



Planning in MDPs

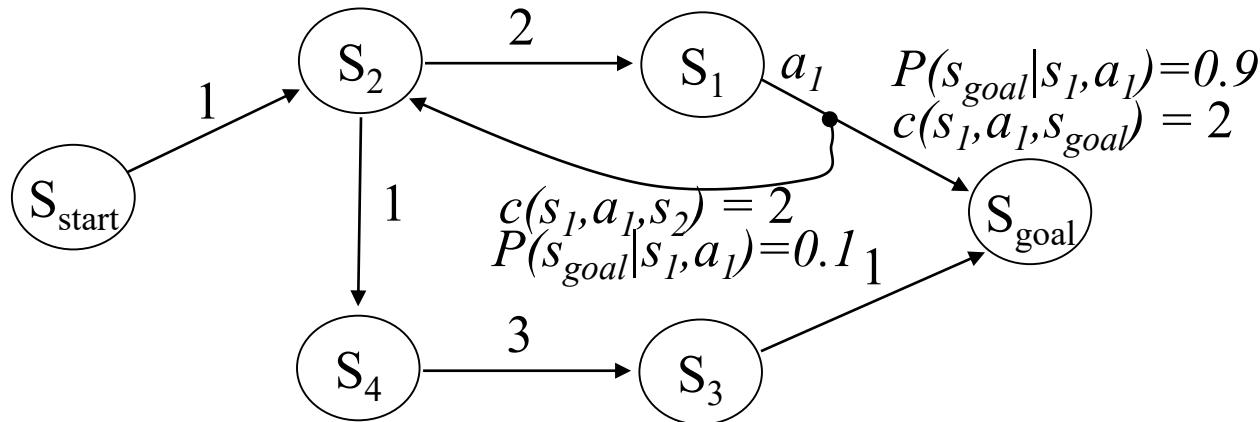
- What plan to compute?
 - Plan that minimizes the worst-case scenario (minimax plan)
 - Plan that minimizes the expected cost



- 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

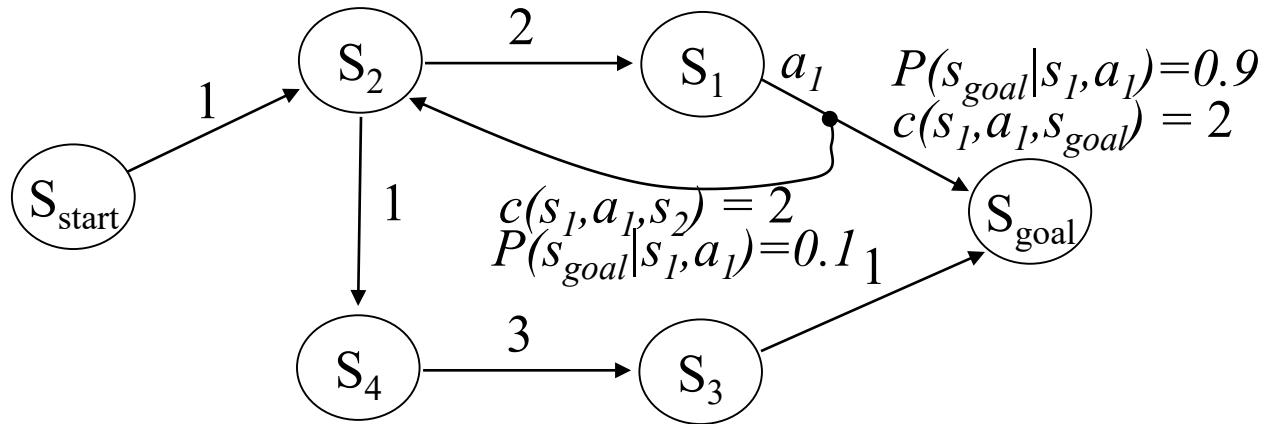
- What plan to compute?
 - Plan that minimizes the worst-case scenario (minimax plan)
 - Plan that minimizes the expected cost



- 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



What is the best plan?

- Optimal policy π^* :

minimizes the *worst* cost-to-goal

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

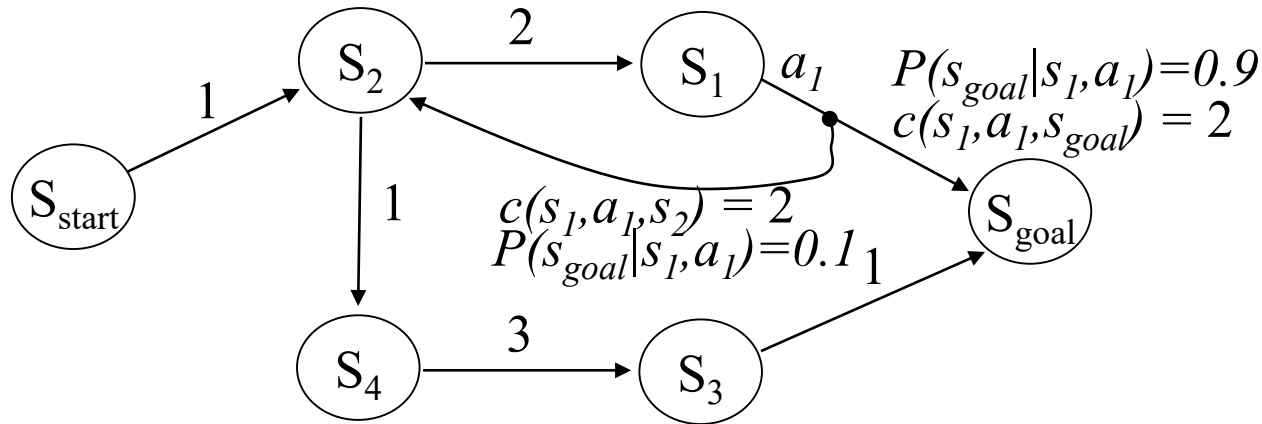
- worst cost-to-goal for $\pi_1 = (\text{go through } s_4)$ is:

$$1 + 1 + 3 + 1 = 6$$

- worst cost-to-goal for $\pi_2 = (\text{try to go through } s_1)$ is:

$$1 + 2 + 2 + 2 + 2 + 2 + 2 + \dots = \infty$$

Minimax Formulation



- Optimal policy π^* :

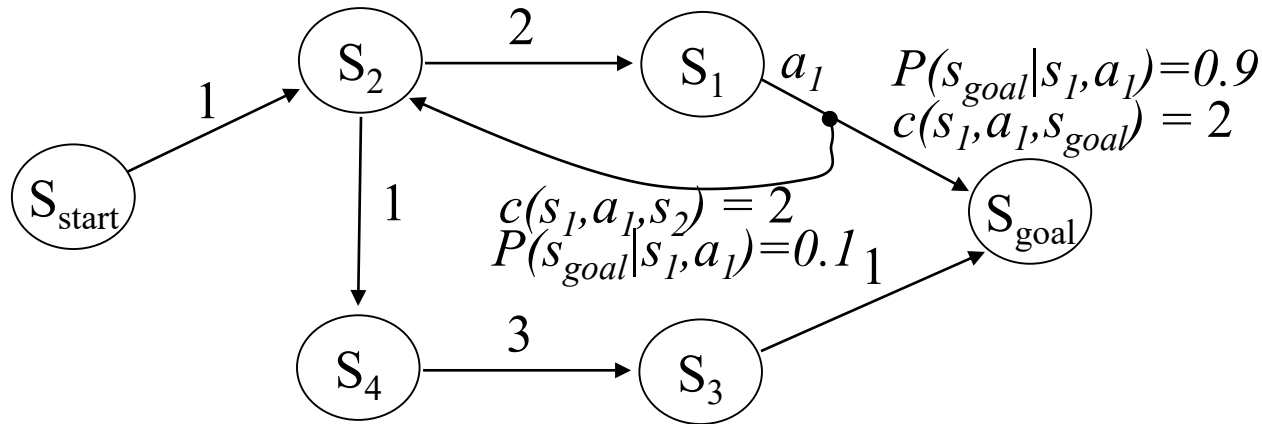
minimizes the *worst* cost-to-goal

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

- Optimal minimax policy $\pi^* = (\text{go through } s_4) =$

$$[\{s_{start}, a_{ne}\}, \{s_2, a_{south}\}, \{s_4, a_{east}\}, \{s_3, a_{ne}\}, \{s_{goal}, null\}]$$

Computing Minimax Plans



- **Minimax backward A*:**

$g(s_{goal}) = 0$; all other g -values are infinite; $OPEN = \{s_{goal}\}$;

while(s_{start} not expanded)

 remove s with the smallest $[f(s) = g(s) + h(s)]$ from $OPEN$;

 insert s into $CLOSED$;

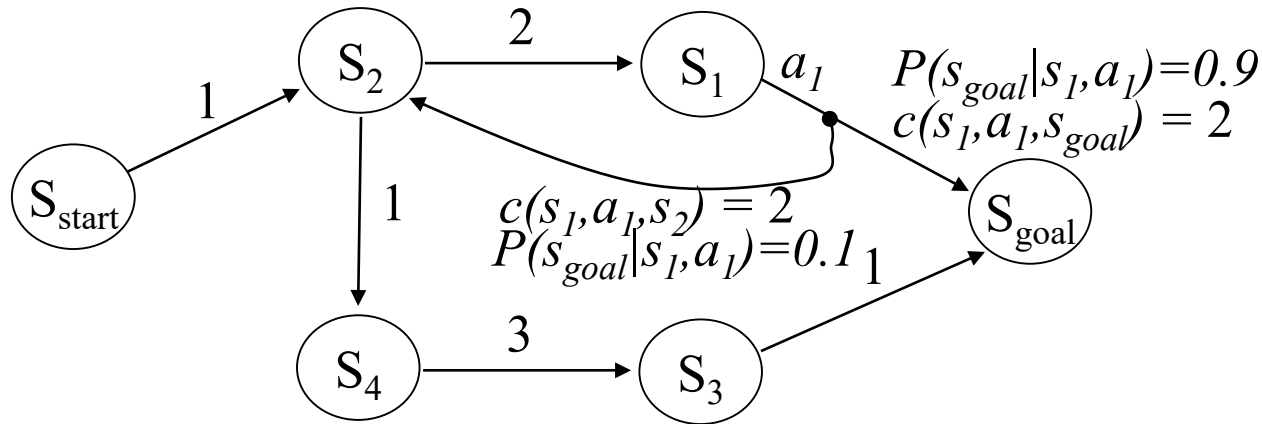
 for every s' s.t $s \in succ(s', a)$ for some a and s' not in $CLOSED$

 if $g(s') > \max_{u \in succ(s', a)} c(s', u) + g(u)$

$g(s') = \max_{u \in succ(s', a)} c(s', u) + g(u)$;

 insert s' into $OPEN$;

Computing Minimax Plans



- Minimax backward A^* :

$g(s_{goal}) = 0$; all other g -values are infinite; $OPEN = \{s_{goal}\}$;

while(s_{start} not expanded)

remove s with the smallest $[f(s) = g(s) + h(s)]$ from $OPEN$;

insert s into $CLOSED$;

for every s' s.t $s \in succ(s', a)$ for some a and s' not in $CLOSED$

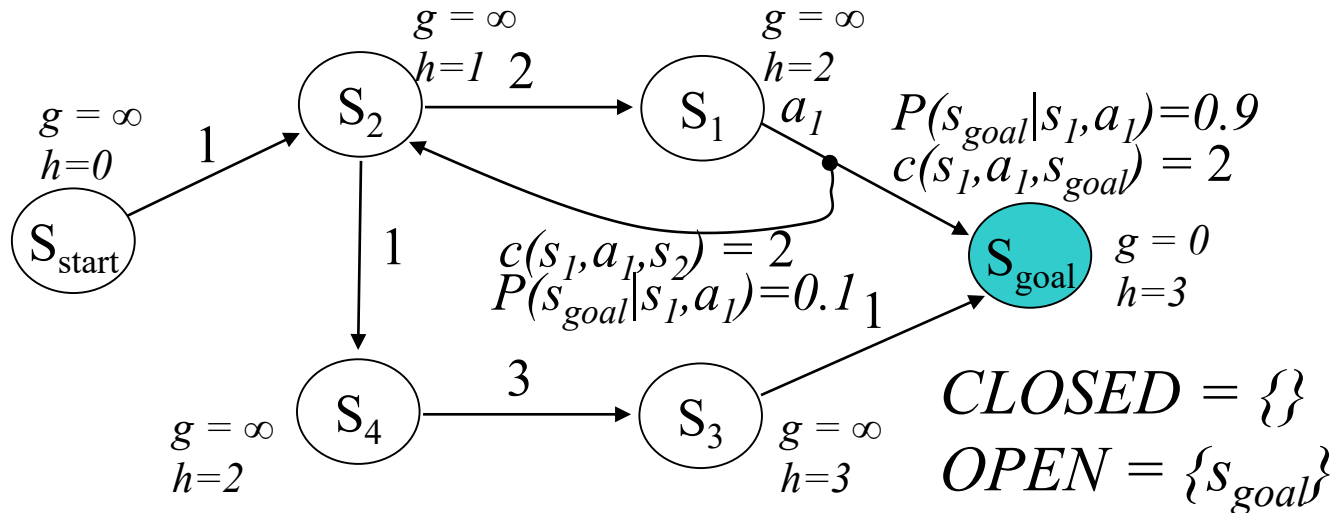
if $g(s') > \max_{u \in succ(s', a)} c(s', u) + g(u)$

$g(s') = \max_{u \in succ(s', a)} c(s', u) + g(u)$;

insert s' into $OPEN$;

reduces to usual backward A^ if
no uncertainty in outcomes*

Computing Minimax Plans



- Minimax backward A*:

$g(s_{goal}) = 0$; all other g -values are infinite; $OPEN = \{s_{goal}\}$;

while(s_{start} not expanded)

remove s with the smallest $[f(s) = g(s) + h(s)]$ from $OPEN$;

insert s into $CLOSED$;

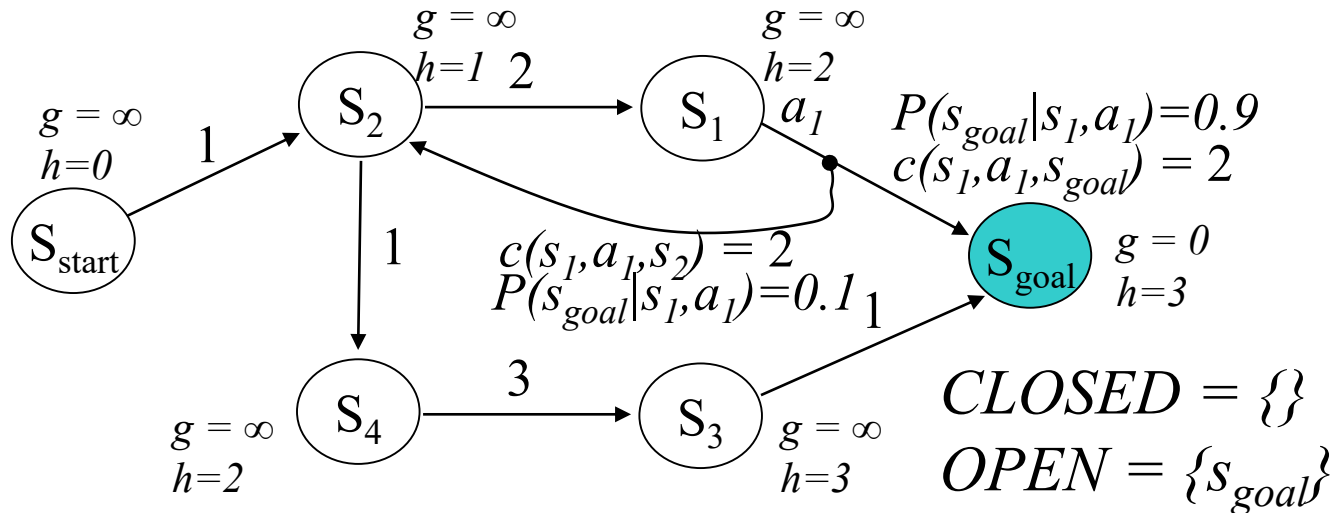
for every s' s.t $s \in succ(s', a)$ for some a and s' not in $CLOSED$

if $g(s') > \max_{u \in succ(s', a)} c(s', u) + g(u)$

$g(s') = \max_{u \in succ(s', a)} c(s', u) + g(u)$;

insert s' into $OPEN$;

Computing Minimax Plans



- Minimax backward A*:

$g(s_{goal}) = 0$; all other g -values are infinite; $OPEN = \{s_{goal}\}$;

while(s_{start} not expanded)

remove s with the smallest $[f(s) = g(s) + h(s)]$ from $OPEN$;

insert s into $CLOSED$;

for every s' s.t $s \in succ(s', a)$ for some a and s' not in $CLOSED$

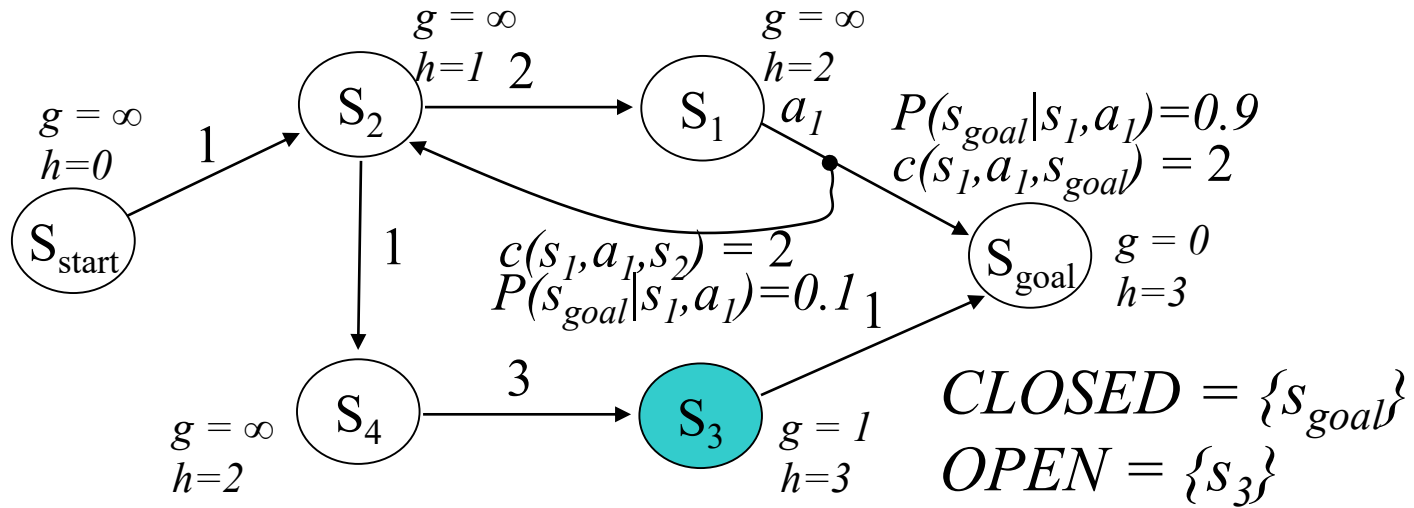
if $g(s') > \max_{u \in succ(s', a)} c(s', u) + g(u)$

$g(s') = \max_{u \in succ(s', a)} c(s', u) + g(u)$;

insert s' into $OPEN$;

*After s_{goal} expanded,
what are $g(s_3)$ and $g(s_1)$?*

Computing Minimax Plans



- Minimax backward A*:

$g(s_{goal}) = 0$; all other g -values are infinite; $OPEN = \{s_{goal}\}$;

while(s_{start} not expanded)

remove s with the smallest $[f(s) = g(s) + h(s)]$ from $OPEN$;

insert s into $CLOSED$;

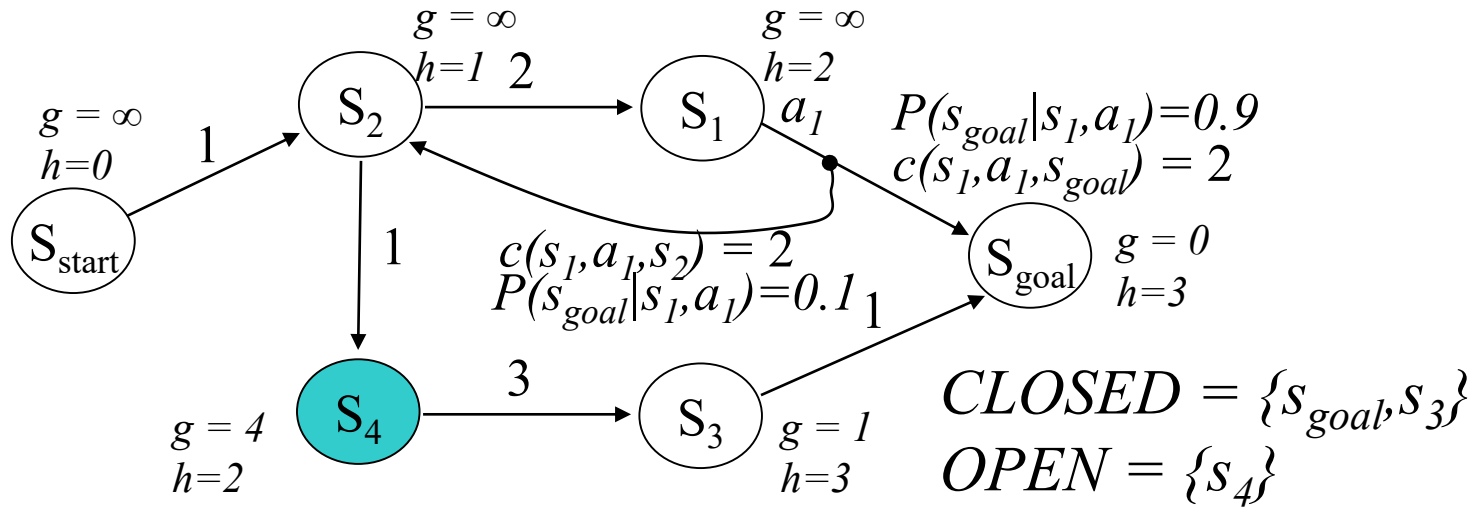
for every s' s.t $s \in succ(s', a)$ for some a and s' not in $CLOSED$

if $g(s') > \max_{u \in succ(s', a)} c(s', u) + g(u)$

$g(s') = \max_{u \in succ(s', a)} c(s', u) + g(u)$;

insert s' into $OPEN$;

Computing Minimax Plans



- Minimax backward A*:

$g(s_{goal}) = 0$; all other g -values are infinite; $OPEN = \{s_{goal}\}$;

while(s_{start} not expanded)

remove s with the smallest $[f(s) = g(s) + h(s)]$ from $OPEN$;

insert s into $CLOSED$;

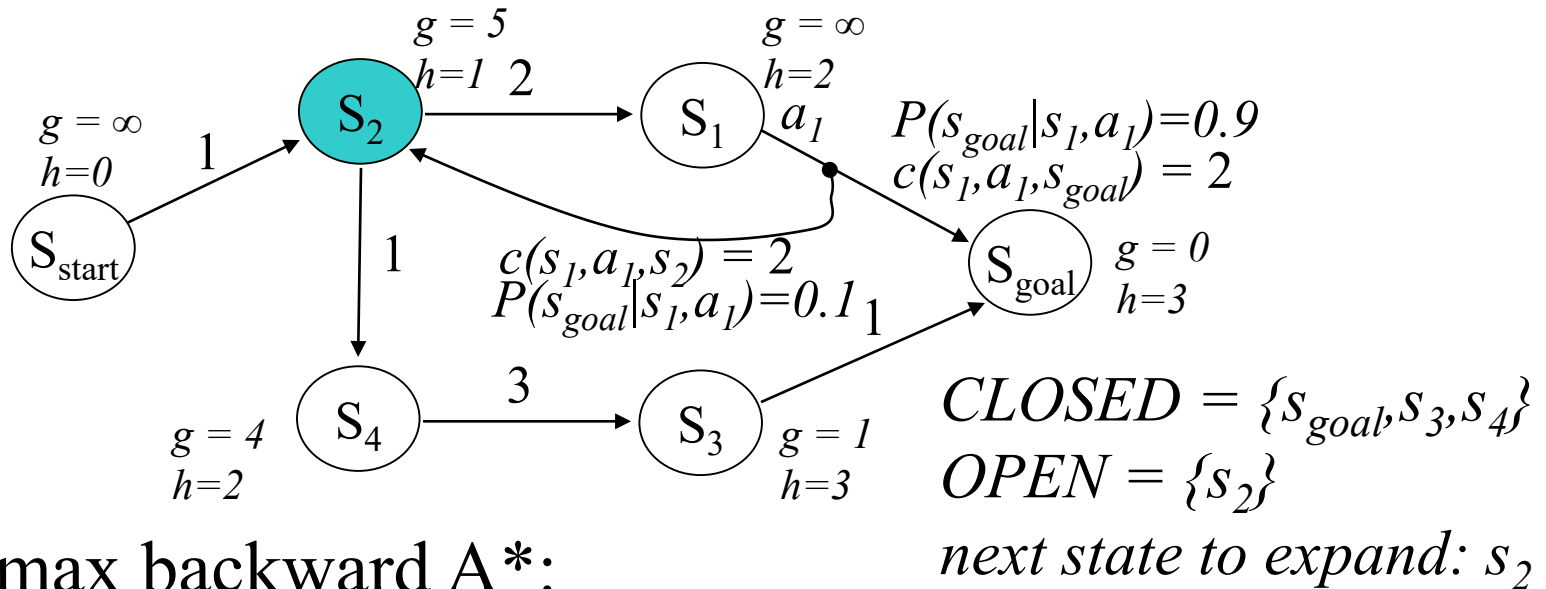
for every s' s.t $s \in succ(s', a)$ for some a and s' not in $CLOSED$

if $g(s') > \max_{u \in succ(s', a)} c(s', u) + g(u)$

$g(s') = \max_{u \in succ(s', a)} c(s', u) + g(u)$;

insert s' into $OPEN$;

Computing Minimax Plans



- Minimax backward A*:

$g(s_{goal}) = 0$; all other g -values are infinite; $OPEN = \{s_{goal}\}$;

while(s_{start} not expanded)

remove s with the smallest $[f(s) = g(s) + h(s)]$ from $OPEN$;

insert s into $CLOSED$;

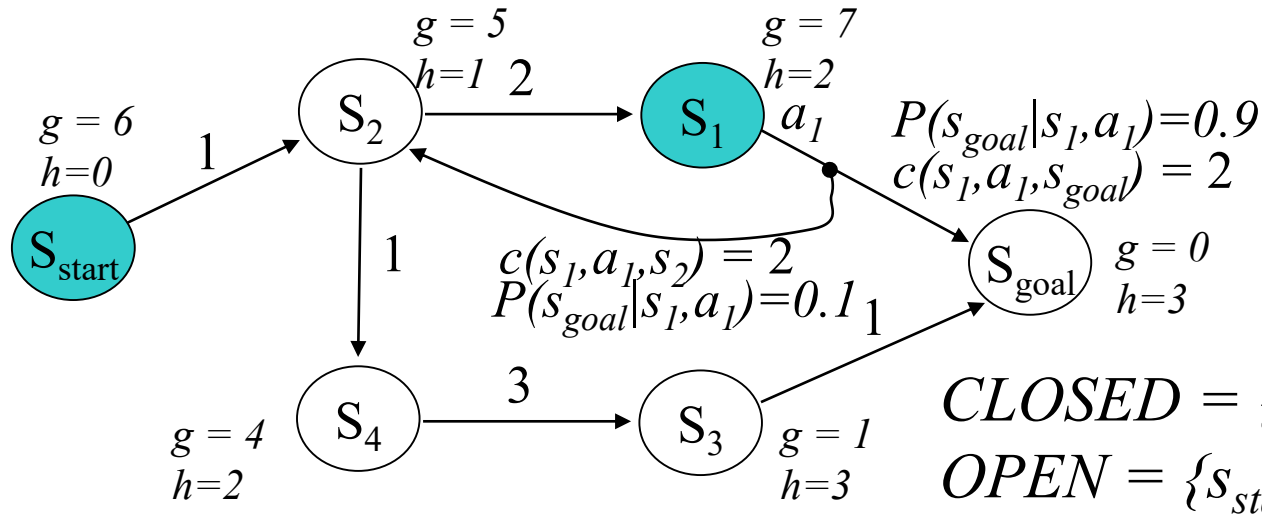
for every s' s.t $s \in succ(s', a)$ for some a and s' not in $CLOSED$

if $g(s') > \max_{u \in succ(s', a)} c(s', u) + g(u)$

$g(s') = \max_{u \in succ(s', a)} c(s', u) + g(u)$;

insert s' into $OPEN$;

Computing Minimax Plans



- Minimax backward A*:

$g(s_{goal}) = 0$; all other g -values are infinite; $OPEN = \{s_{goal}\}$;

while(s_{start} not expanded)

remove s with the smallest $[f(s) = g(s) + h(s)]$ from $OPEN$;

insert s into $CLOSED$;

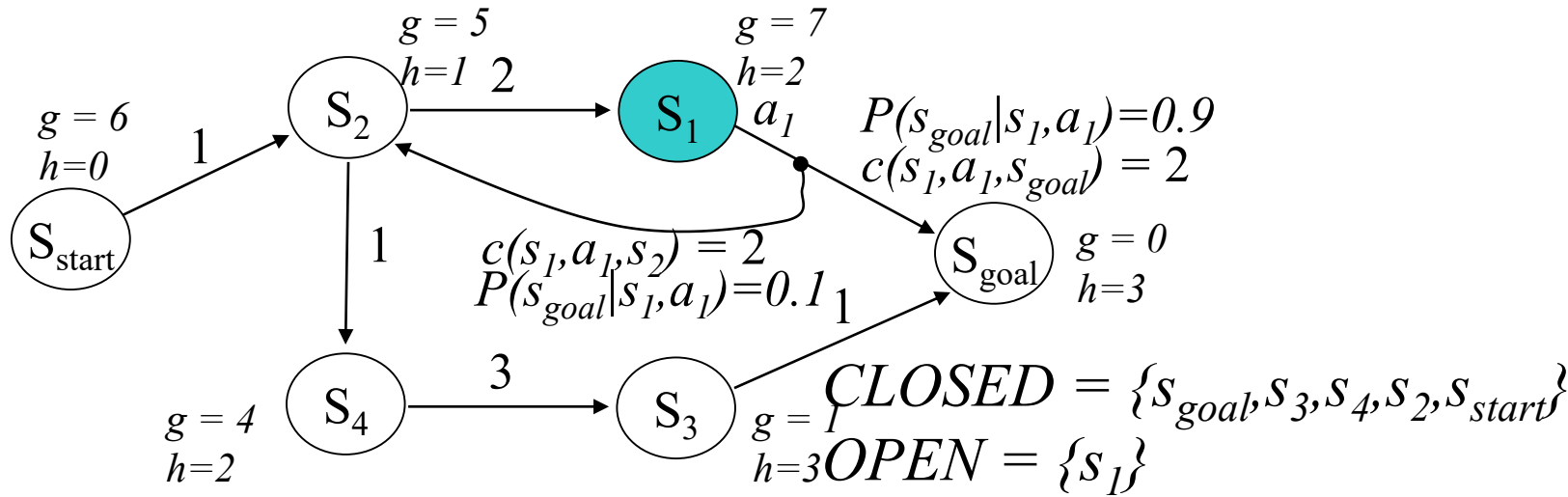
for every s' s.t $s \in succ(s', a)$ for some a and s' not in $CLOSED$

if $g(s') > \max_{u \in succ(s', a)} c(s', u) + g(u)$

$g(s') = \max_{u \in succ(s', a)} c(s', u) + g(u)$;

insert s' into $OPEN$;

Computing Minimax Plans



- Minimax backward A*:

$g(s_{goal}) = 0$; all other g -values are infinite; $OPEN = \{s_{goal}\}$;

while(s_{start} not expanded)

remove s with the smallest $[f(s) = g(s) + h(s)]$ from $OPEN$;

insert s into $CLOSED$;

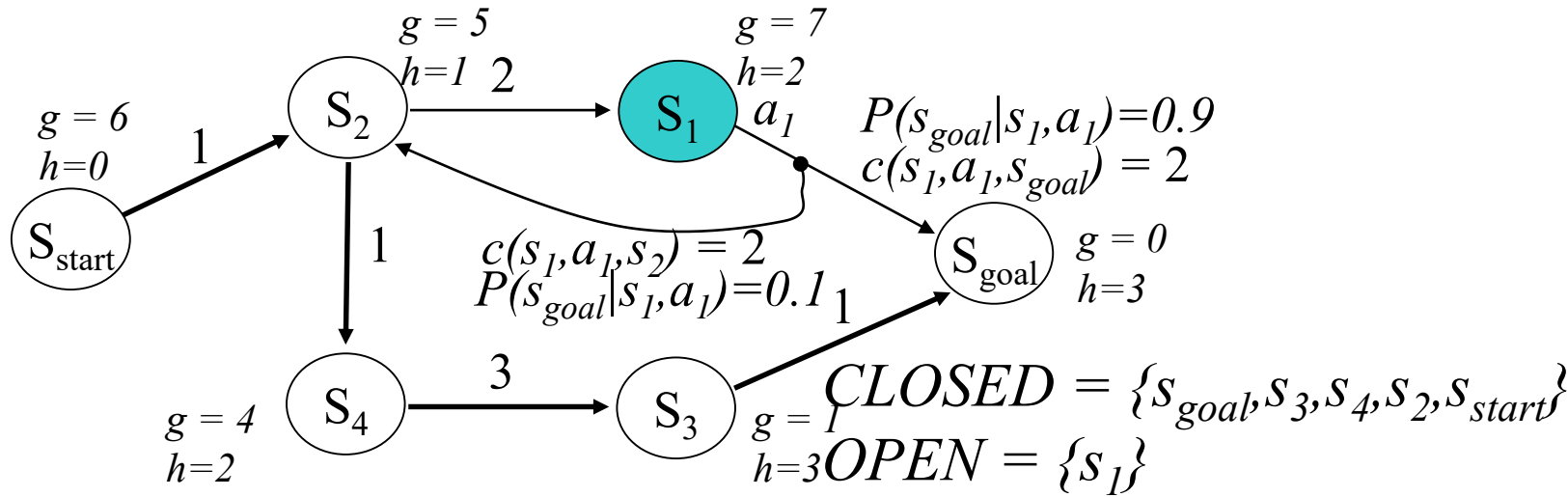
for every s' s.t $s \in succ(s', a)$ for some a and s' not in $CLOSED$

if $g(s') > \max_{u \in succ(s', a)} c(s', u) + g(u)$

$g(s') = \max_{u \in succ(s', a)} c(s', u) + g(u)$;

insert s' into $OPEN$;

Computing Minimax Plans



- Minimax backward A*:

$g(s_{goal}) = 0$; all other g -values are infinite; $OPEN = \{s_{goal}\}$;

while(s_{start} not expanded)

remove s with the smallest $[f(s) = g(s) + h(s)]$ from $OPEN$;

insert s into $CLOSED$;

for every s' s.t $s \in succ(s', a)$ for some a and s' not in $CLOSED$

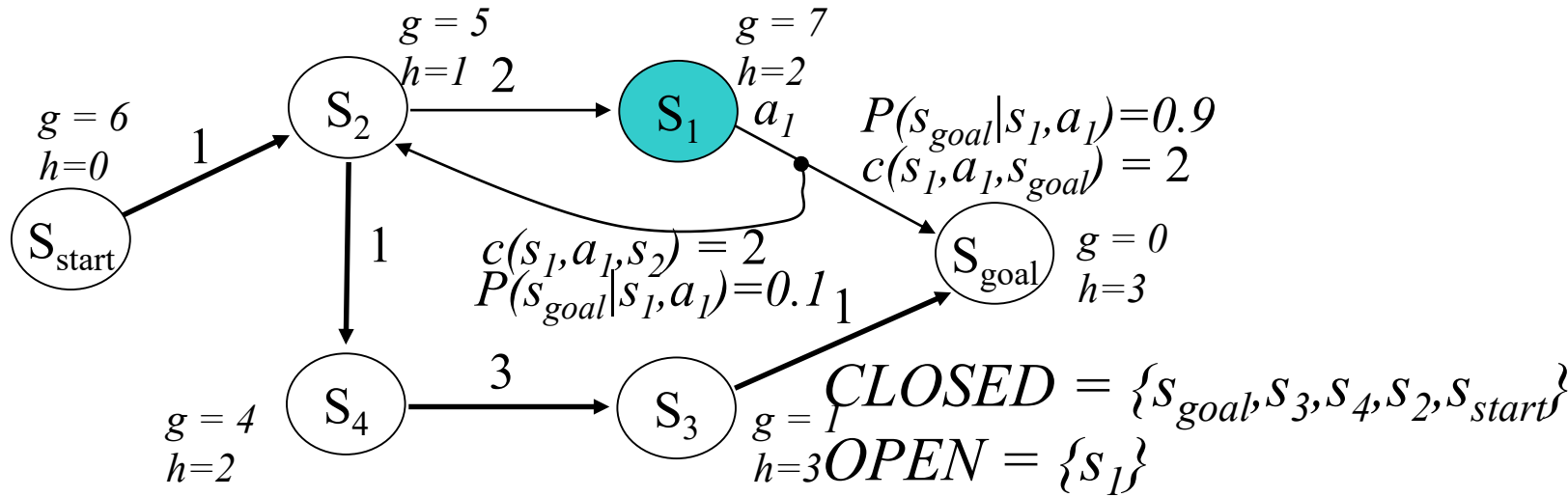
if $g(s') > \max_{u \in succ(s', a)} c(s', u) + g(u)$

$g(s') = \max_{u \in succ(s', a)} c(s', u) + g(u)$;

insert s' into $OPEN$;

*in this example, the computed policy is a path,
but in general it is a tree*

Computing Minimax Plans



- Minimax backward A*:

$g(s_{goal}) = 0$; all other g -values are infinite; $OPEN = \{s_{goal}\}$;

while(s_{start} not expanded)

remove s with the smallest $[f(s) = g(s) + h(s)]$ from $OPEN$;

insert s into $CLOSED$;

for every s' s.t. $s \in succ(s', a)$ for some a

if $g(s') > \max_{u \in succ(s', a)} c(s', u) + g(u)$

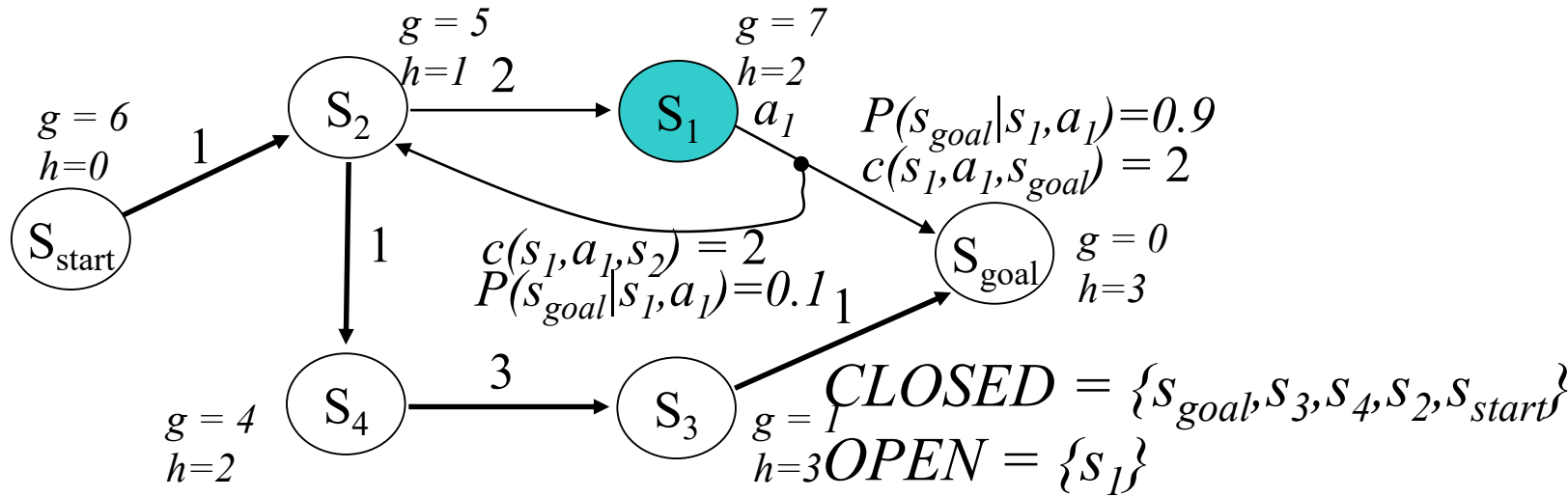
$g(s') = \max_{u \in succ(s', a)} c(s', u) + g(u)$;

insert s' into $OPEN$;

*What are its branches?
Why tree, and not graph?*

*in this example, the computed policy is a path,
but in general it is a tree*

Computing Minimax Plans



• Minimax backward A*:

DONE

$g(s_{goal}) = 0$; all other g -values are infinite; $OPEN = \{s_{goal}\}$;

while(s_{start} not expanded)

remove s with the smallest $[f(s) = g(s) + h(s)]$ from $OPEN$;

insert s into $CLOSED$;

for every s' s.t $s \in succ(s', a)$ for some a and s' not in $CLOSED$

if $g(s') > \max_{u \in succ(s', a)} c(s', u) + g(u)$

$g(s') = \max_{u \in succ(s', a)} c(s', u) + g(u)$

insert s' into $OPEN$;

Minimax A guarantees to find an optimal path,
and never expands a state more than once,
provided heuristics are consistent (just like A*)*

Computing Minimax Plans

- Pros/cons of minimax plans
 - robust to uncertainty
 - overly pessimistic
 - harder to compute than normal paths
 - especially if backwards minimax A^* does not apply
 - even if backwards minimax A^* does apply, still more expensive than computing a single path with A^* (heuristics are not guiding well)

Why?

Summary

- MDP (Markov Decision Process): a graph where one or more actions have multiple possible outcomes, each outcome with its probability of happening
- Minimax formulation: ignores probabilities and tries to compute the plan that is the “best” assuming the worst possible outcome for every action
- Minimax backward A^*
 - computes minimax optimal plan
 - searches backwards which sometimes can be hard/computationally very expensive (consider moving-target search, what is a goal?)