

16-350

Planning Techniques for Robotics

Search Algorithms:

*Multi-goal A**

Maxim Likhachev

Robotics Institute

Carnegie Mellon University

Support for Multiple Goal Candidates

- How to compute a least-cost path to any one of the possible goals?
 - Example 1: Computing a least-cost path to a parking spot given multiple parking spaces (some are better, some are worse, some are closer, some are further)
 - Example 2: Catching a moving target whose future trajectory is known (i.e., multiple potential intercept points)
 - Example 3: Mapping/exploration (next lecture)

A* Search

Main function

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

ComputePath();

publish solution;

ComputePath function

while(s_{goal} is not expanded and $OPEN \neq \emptyset$)

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

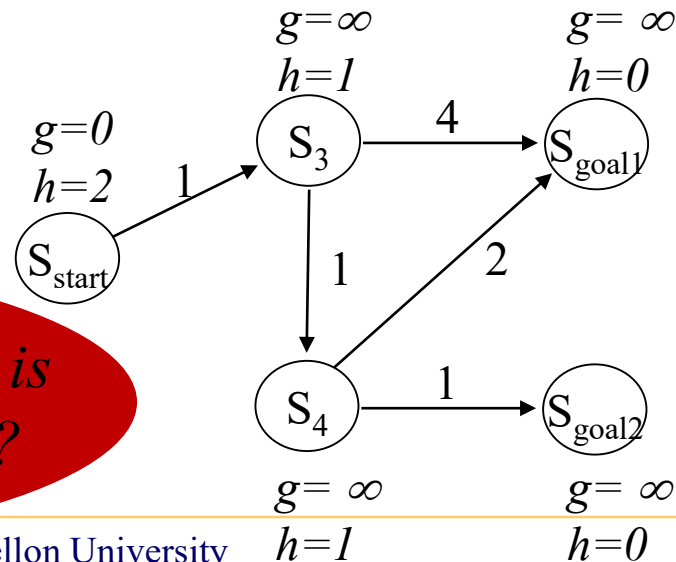
 insert s into $CLOSED$;

 for every successor s' of s such that s' not in $CLOSED$

 if $g(s') > g(s) + c(s, s')$

$g(s') = g(s) + c(s, s')$;

 insert s' into $OPEN$;



How to find a least-cost path that is lowest across all possible goals?

Introducing “imaginary” goal

Main function

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

ComputePath();

publish solution;

ComputePath function

while(s_{goal} is not expanded and $OPEN \neq \emptyset$)

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

 insert s into $CLOSED$;

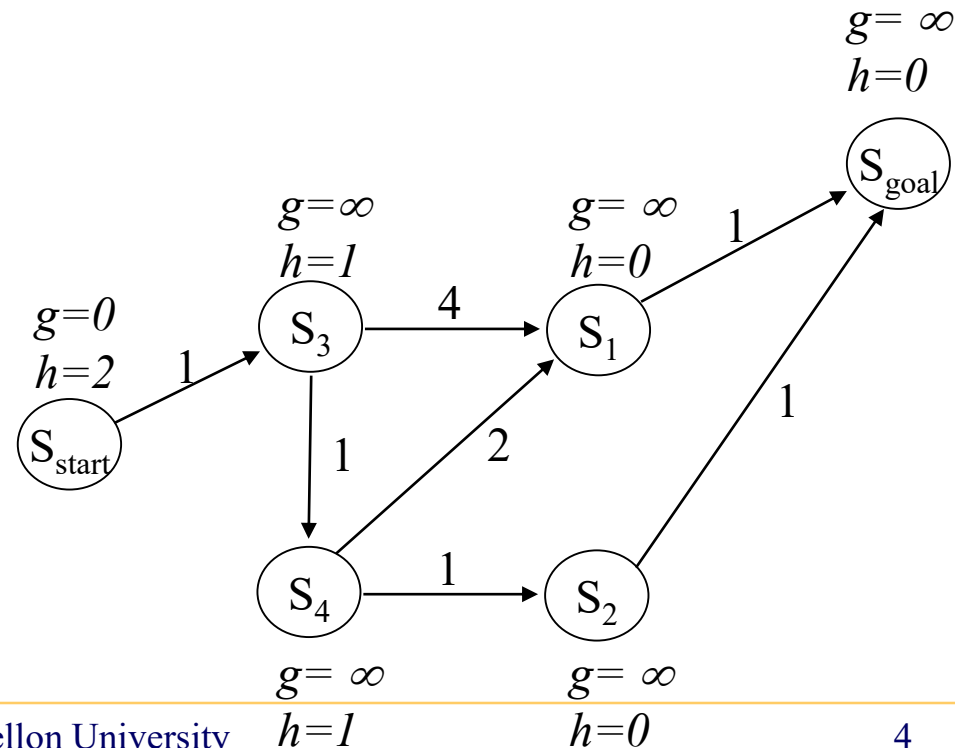
 for every successor s' of s such that s' not in $CLOSED$

 if $g(s') > g(s) + c(s, s')$

$g(s') = g(s) + c(s, s')$;

 insert s' into $OPEN$;

Equivalent problem but with a single goal!



Introducing “imaginary” goal

Main function

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

ComputePath();

publish solution;

ComputePath function

while(s_{goal} is not expanded and $OPEN \neq \emptyset$)

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

 insert s into $CLOSED$;

 for every successor s' of s such that s' not in $CLOSED$

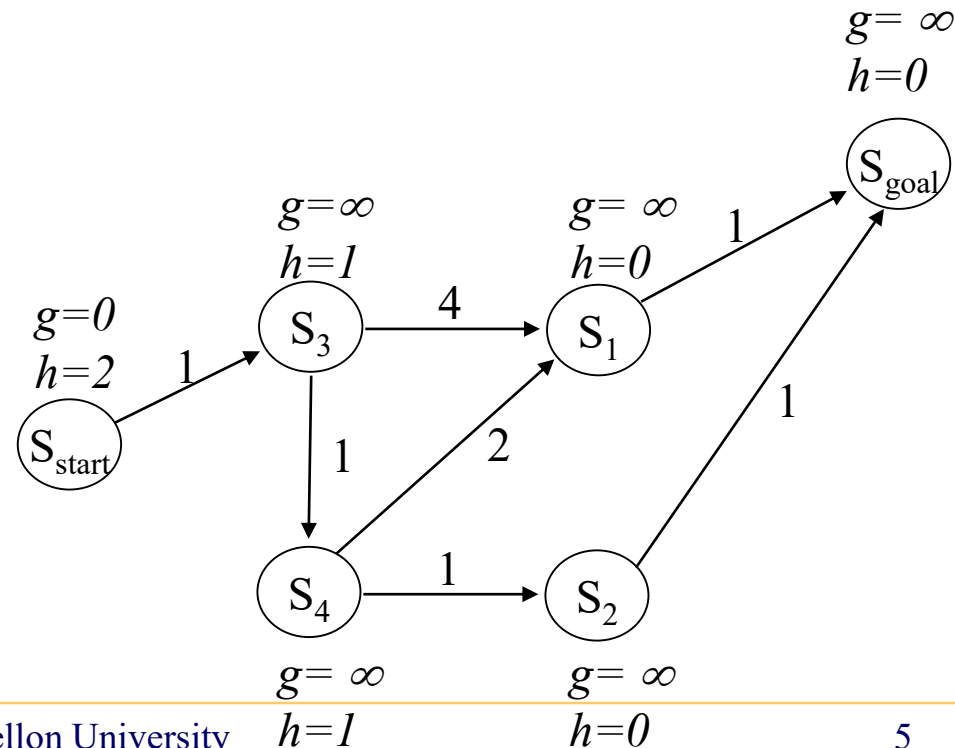
 if $g(s') > g(s) + c(s, s')$

$g(s') = g(s) + c(s, s')$;

 insert s' into $OPEN$;

Equivalent problem but with a single goal!

How to prove it?



Support for “unequal” goals

Main function

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

ComputePath();

publish solution;

ComputePath function

while(s_{goal} is not expanded and $OPEN \neq \emptyset$)

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

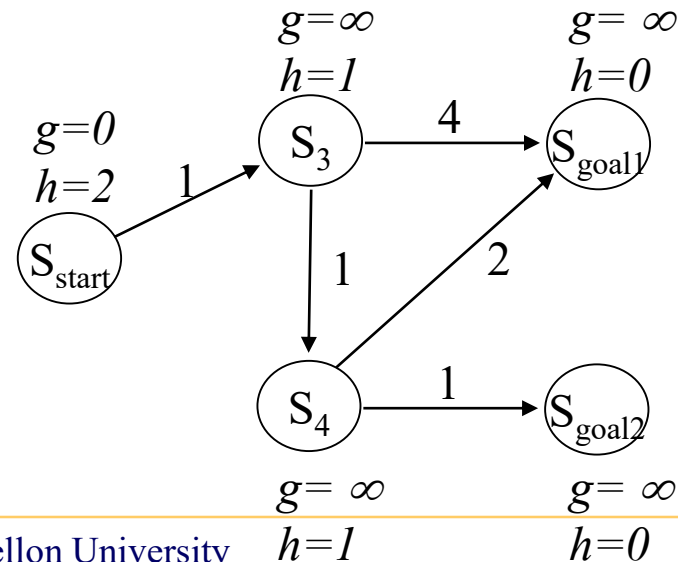
 insert s into $CLOSED$;

 for every successor s' of s such that s' not in $CLOSED$

 if $g(s') > g(s) + c(s, s')$

$g(s') = g(s) + c(s, s')$;

 insert s' into $OPEN$;



*What if some goals
are better than others?*

Support for “unequal” goals

Main function

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

ComputePath();

publish solution;

ComputePath function

while(s_{goal} is not expanded and $OPEN \neq \emptyset$)

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

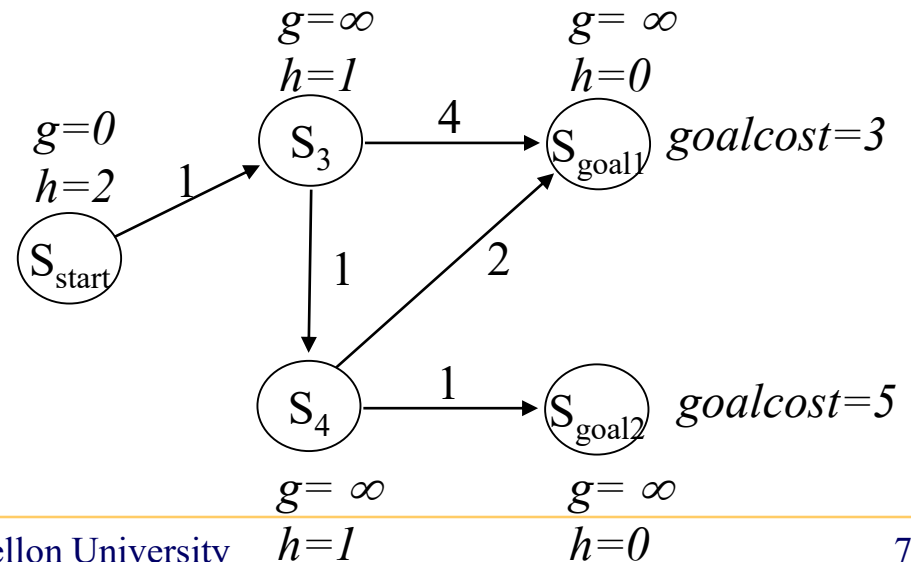
 insert s into $CLOSED$;

 for every successor s' of s such that s' not in $CLOSED$

 if $g(s') > g(s) + c(s, s')$

$g(s') = g(s) + c(s, s')$;

 insert s' into $OPEN$;



What if some goals are better than others?

Support for “unequal” goals

Main function

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

ComputePath();

publish solution;

ComputePath function

while(s_{goal} is not expanded and $OPEN \neq \emptyset$)

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

 insert s into $CLOSED$;

 for every successor s' of s such that s' not in $CLOSED$

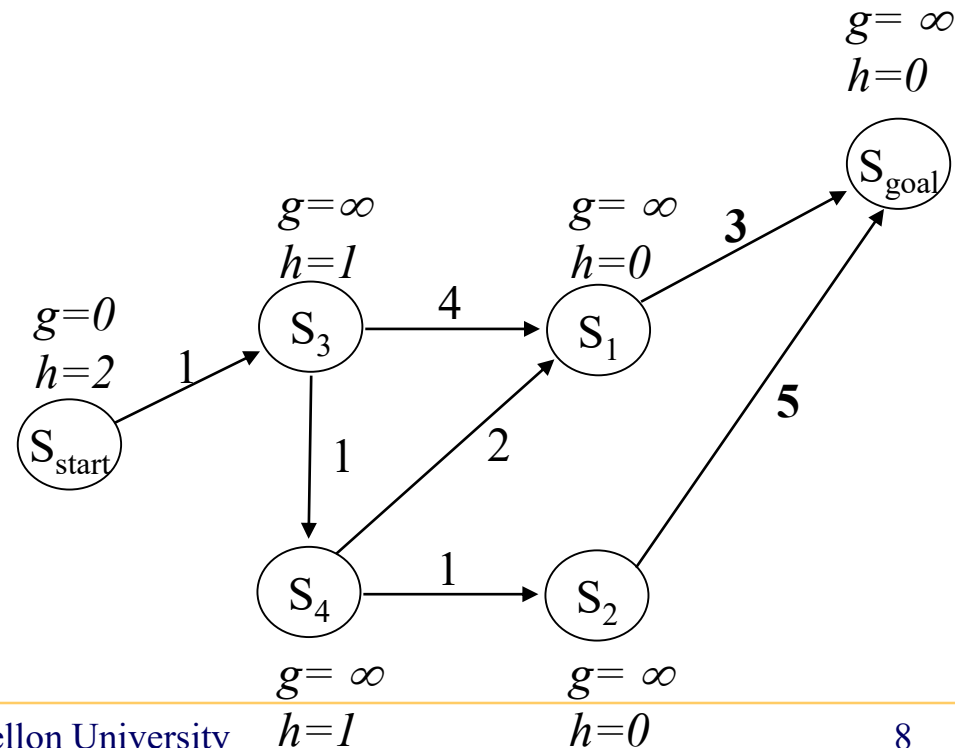
 if $g(s') > g(s) + c(s, s')$

$g(s') = g(s) + c(s, s')$;

 insert s' into $OPEN$;

Equivalent problem but with a single goal!

How to prove it?



Support for “unequal” goals

Main function

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

ComputePath();

publish solution;

ComputePath function

while(s_{goal} is not expanded and $OPEN \neq \emptyset$)

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

insert s into $CLOSED$;

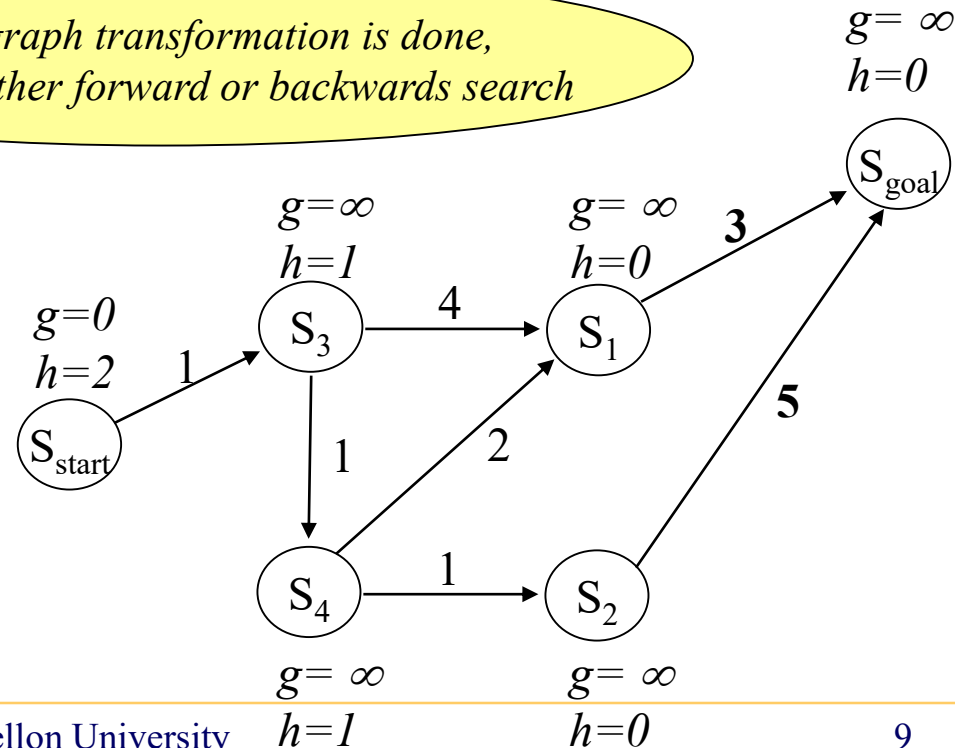
for every successor s' of s such that s' not in $CLOSED$

if $g(s') > g(s) + c(s, s')$

$g(s') = g(s) + c(s, s')$;

insert s' into $OPEN$;

Once the graph transformation is done,
you can run either forward or backwards search



Support for “unequal” goals

Main function

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

ComputePath();

publish solution;

ComputePath function

while(s_{goal} is not expanded and $OPEN \neq \emptyset$)

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

insert s into $CLOSED$;

for every successor s' of s such that s' not in $CLOSED$

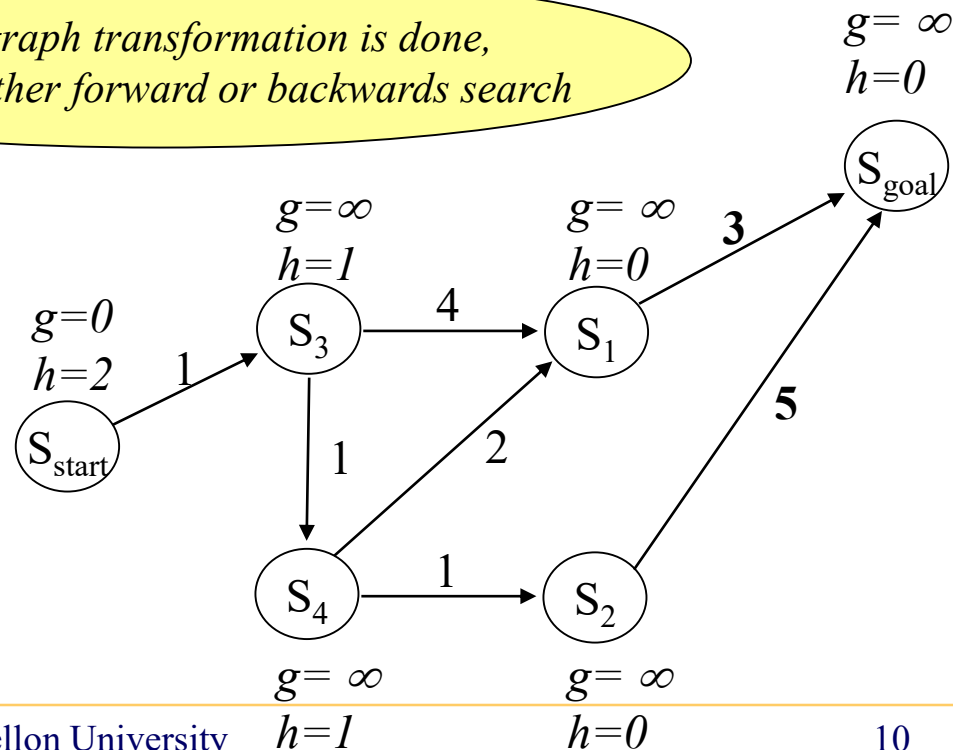
if $g(s') > g(s) + c(s, s')$

$g(s') = g(s) + c(s, s')$;

insert s' into $OPEN$;

*Any impact on how
heuristics is computed?*

*Once the graph transformation is done,
you can run either forward or backwards search*



What You Should Know...

- How to search for a path that is cost-minimal given multiple potential goals with different goalcosts (e.g., know how the graph transformation using “imaginary” goal)