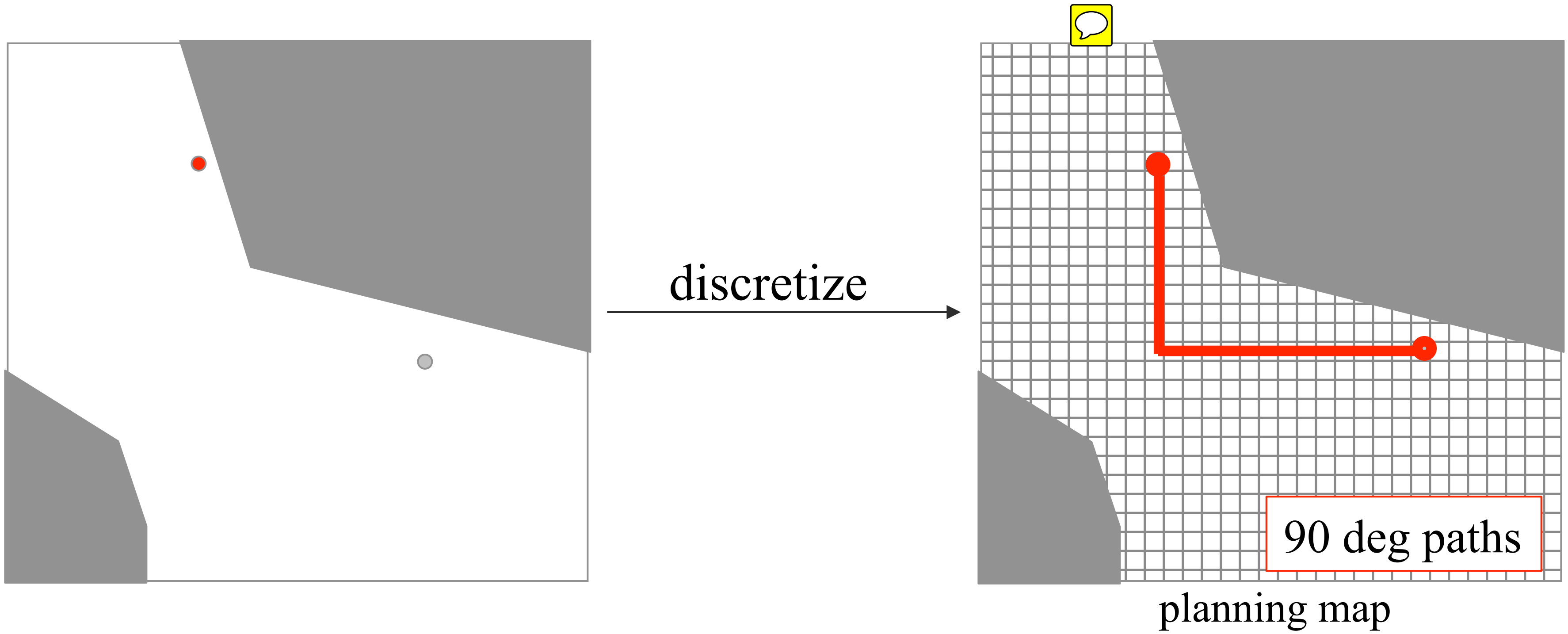
Approximate Cell Decomposition

4-connected graph

- Each cell is connected to four neighbors (N, S, E, W)

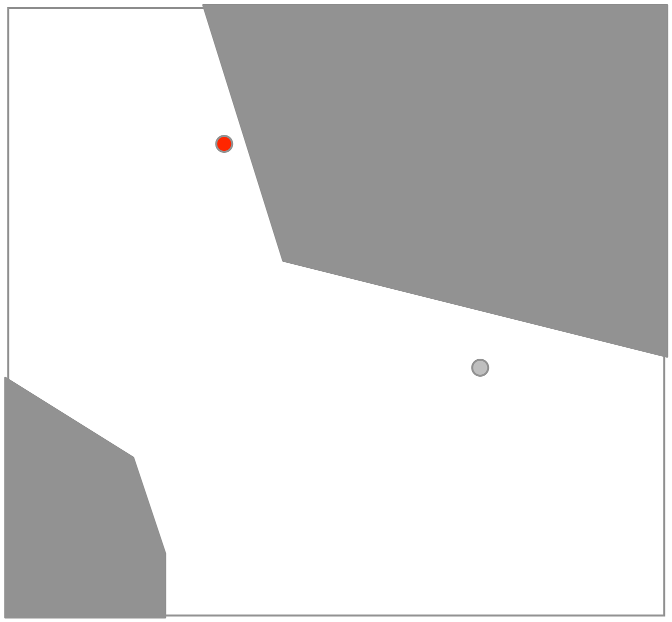


Can we get "finer" paths with the same resolution?

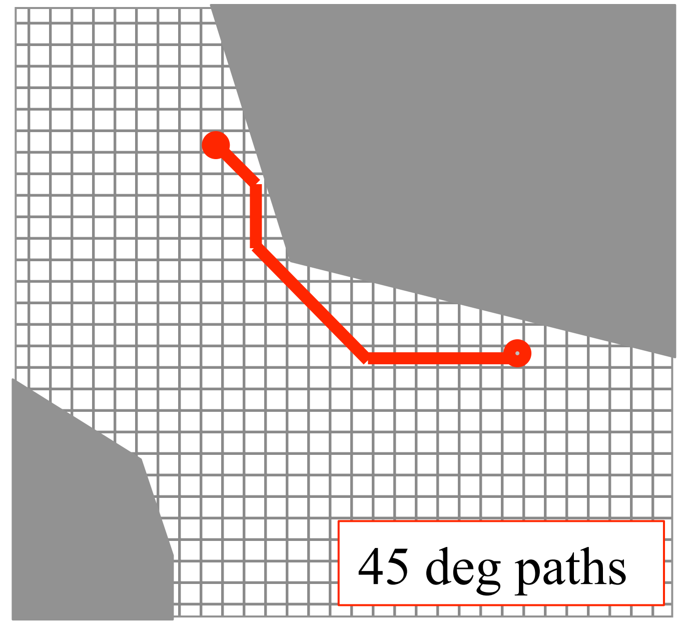
4-connected versus 8-connected

8-connected graph

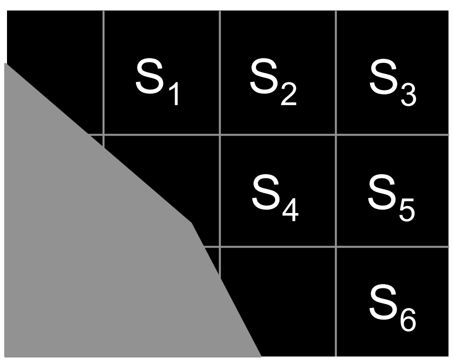
- Each cell is connected to eight neighbors (N, S, E, W, NE, SE, NW, SW)



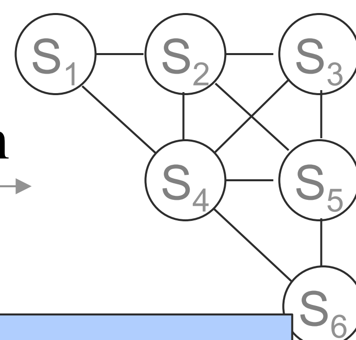
discretize



planning map



convert into a graph

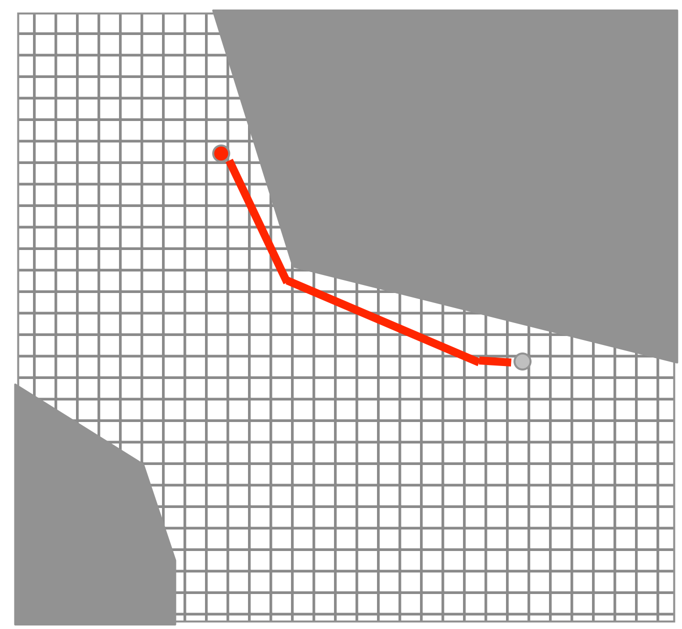


search the graph for a least-cost path from s_{start} to s_{goal}

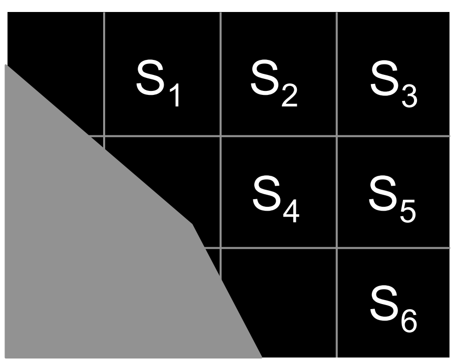
Eight-connected graph

Planning via Approximate Cell Decomposition

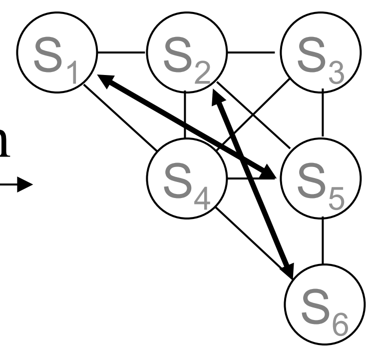
- Graph construction:
  - connect cells to neighbor of neighbors
  - path is restricted to 22.5º degrees



16-connected grid

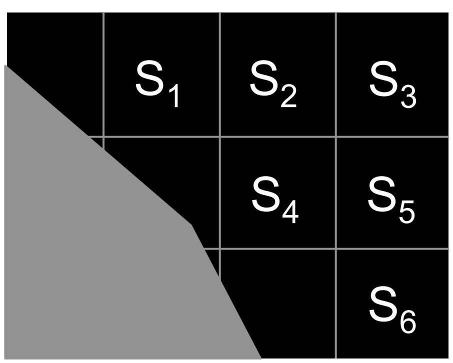


convert into a graph

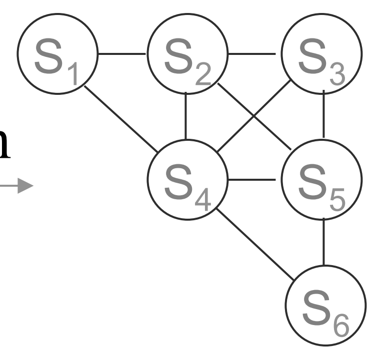


Planning via Approximate Cell Decomposition

- Approximate Cell Decomposition:
  - what to do with partially blocked cells?



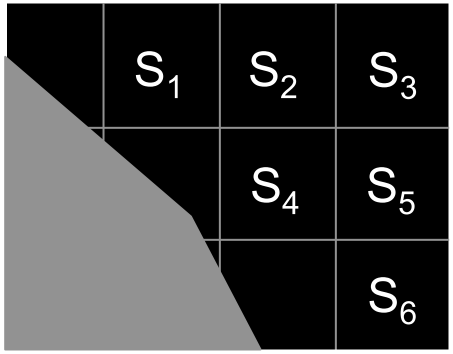
convert into a graph



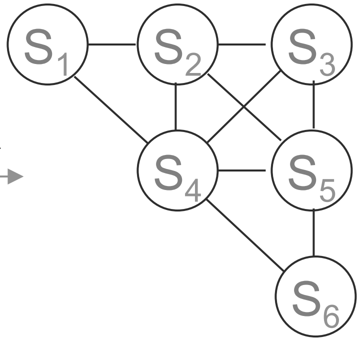
search the graph for a least-cost path from s_{start} to s_{goal}

Planning via Approximate Cell Decomposition

- Approximate Cell Decomposition:
  - what to do with partially blocked cells?
  - make it untraversable – incomplete (may not find a path that exists)



convert into a graph

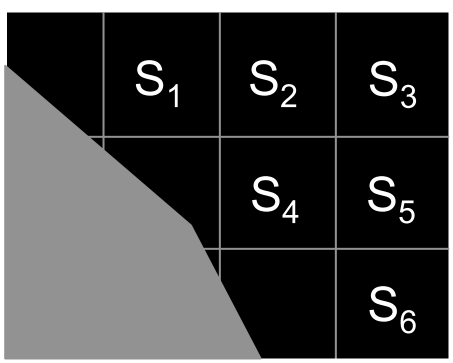


search the graph for a least-cost path from s_{start} to s_{goal}

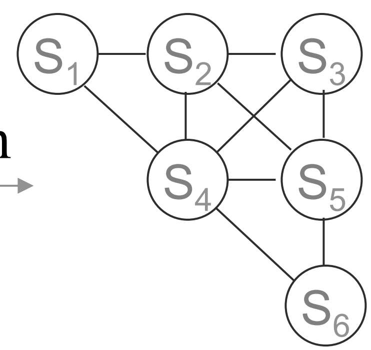
Planning via Approximate Cell Decomposition

- Approximate Cell Decomposition:
  - what to do with partially blocked cells?
  - make it traversable – incorrect (may return valid paths when none exist)

so, what's the solution?



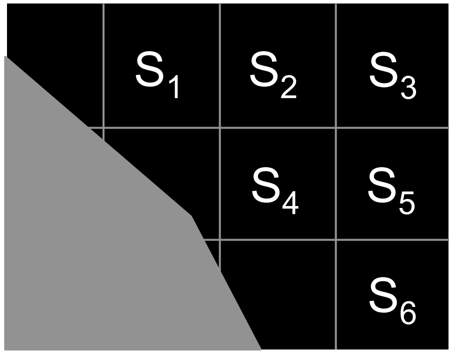
convert into a graph



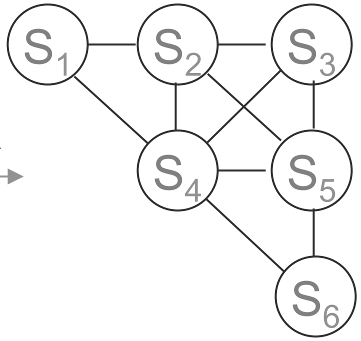
search the graph for a least-cost path from s_{start} to s_{goal}

Planning via Approximate Cell Decomposition

- Approximate Cell Decomposition:
  - solution 1:
    - make the discretization very fine
    - expensive, especially in high-D



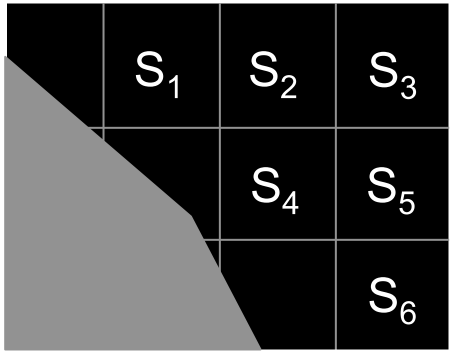
convert into a graph



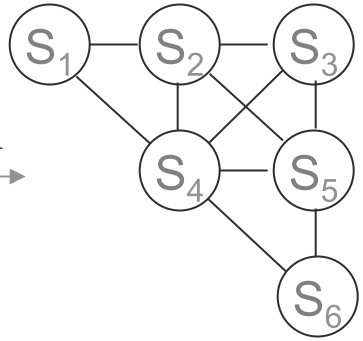
search the graph for a least-cost path from s_{start} to s_{goal}

Planning via Approximate Cell Decomposition

- Approximate Cell Decomposition:
  - solution 2:
    - make the discretization adaptive



convert into a graph



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

Graph Search

Background: Planning as Tree Search

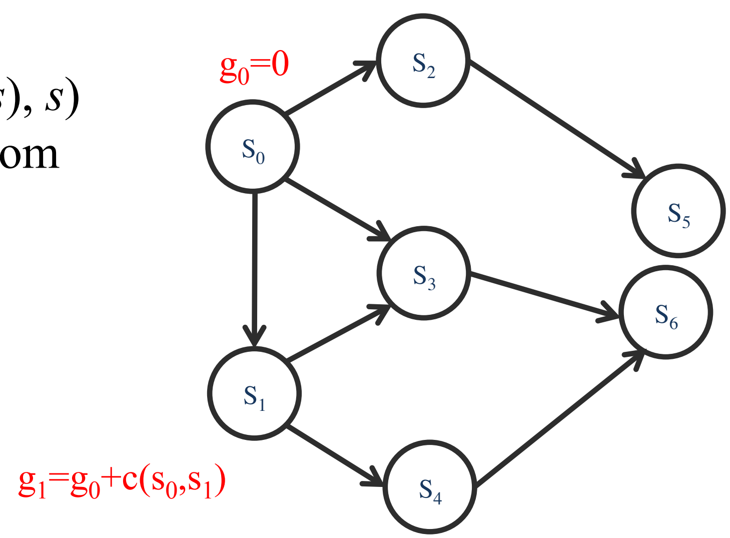
Perform tree-based search

(need cost function *c*)

- Construct the root of the tree as the start state, and give it value 0
- While there are unexpanded leaves in the tree
  - Find the leaf *s* with the lowest value
  - For each action, create a new child leaf of *s*
  - Set the value of each child as:

    $$g(s) = g(parent(s)) + c(parent(s), s)$$

    where $c(s, s')$ is the cost of moving from s to s'



Penn Engineering

Background: What action to choose?

Depth-first search

Breadth-first search

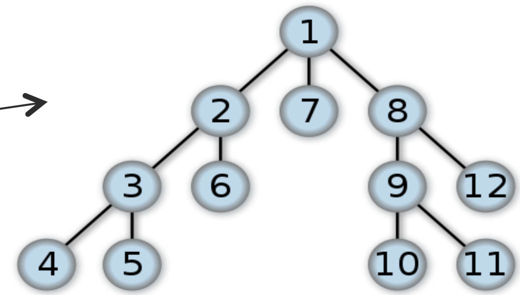
Best-first search

- Djikstra (1959)

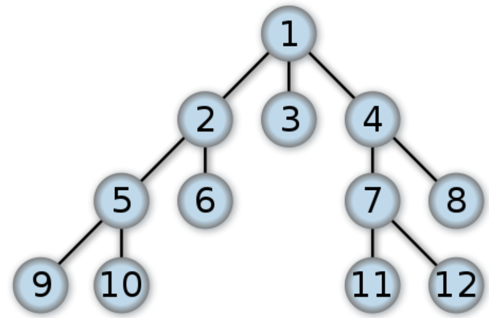
  Single-source shortest path problem for routing

- Hart (1968)

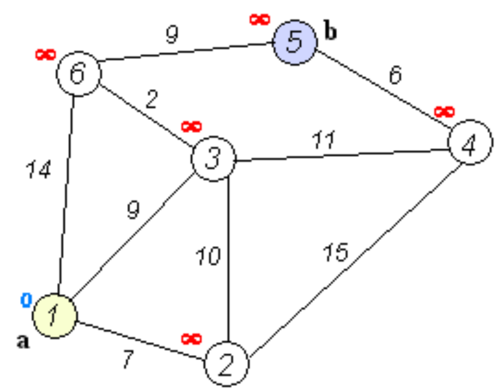
  A* algorithm



Ref: Wikipedia



Ref: Wikipedia



Ref: Wikipedia

Penn Engineering

Background: What action to choose?

How to determine the lowest-cost child to consider next?

1 Shallowest next (breadth-first)

- Guaranteed shortest
- Storage intensive

2 Deepest next (depth-first)

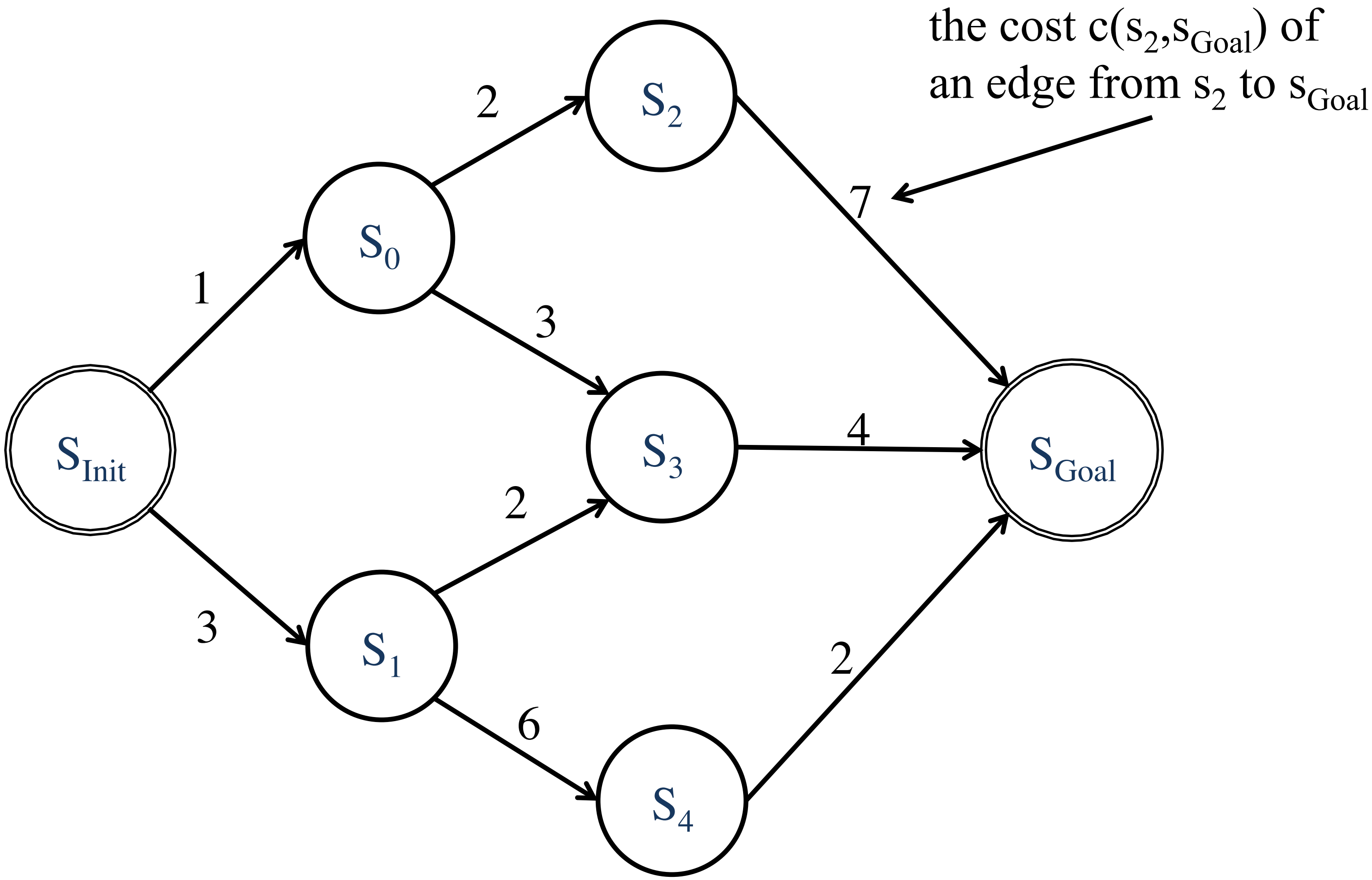
- No optimality
- Potentially storage cheap

3 A*

- Optimal
- Complete
- Efficient if good heuristics are used

Searching Graphs for a Least-cost Path

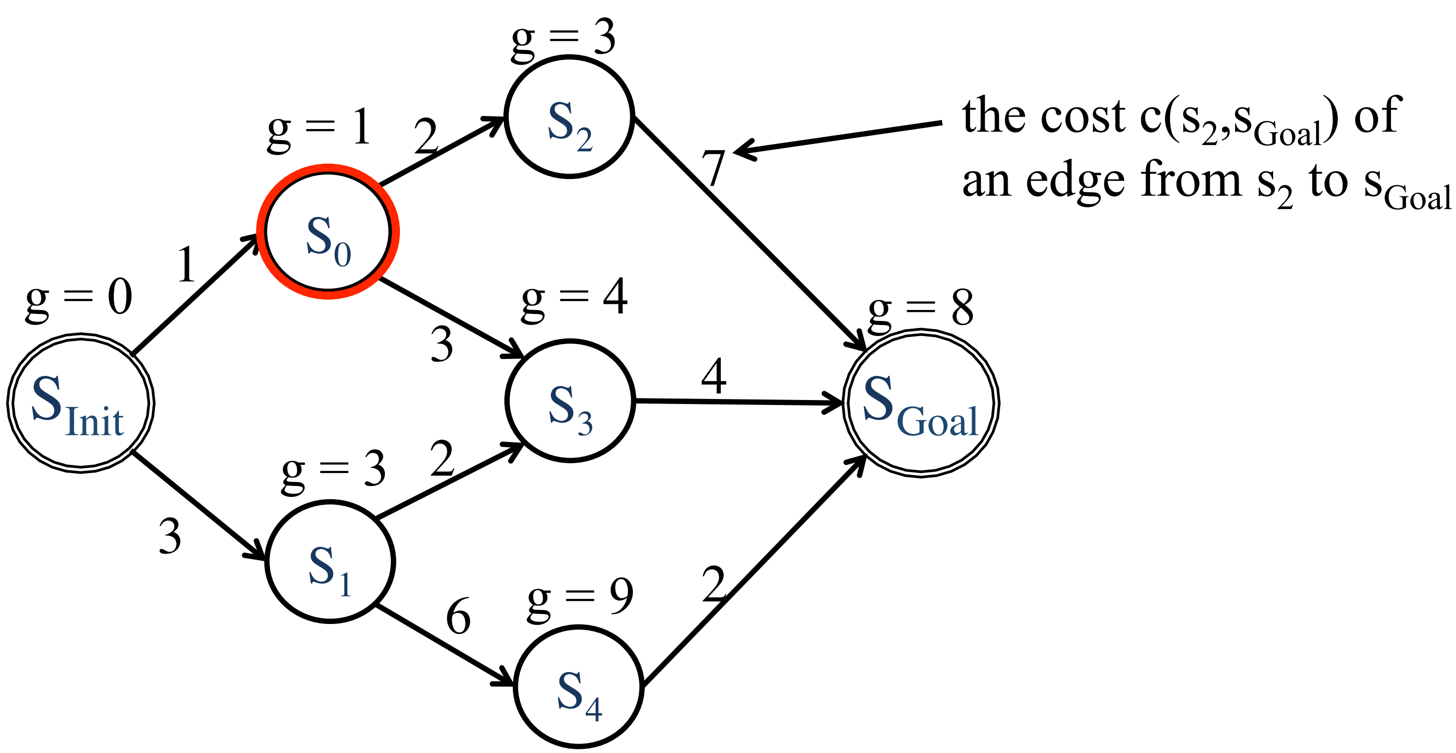
Total_cost (s_{Init}, s, s_{Goal}) = running_cost (s_{Init}, s) + cost_to_go(s, s_{Goal})



Searching Graphs for a Least-cost Path

Estimate $g(s)$, the *running_cost*(s_{Init}, s), for each state s.

- $g(s)$ – an estimated cost of a least-cost path from s_{Init} to s
- optimal values satisfy: $g(s) = \min_{p \in predecessor(s)} g(p) + c(p,s)$



A* Search

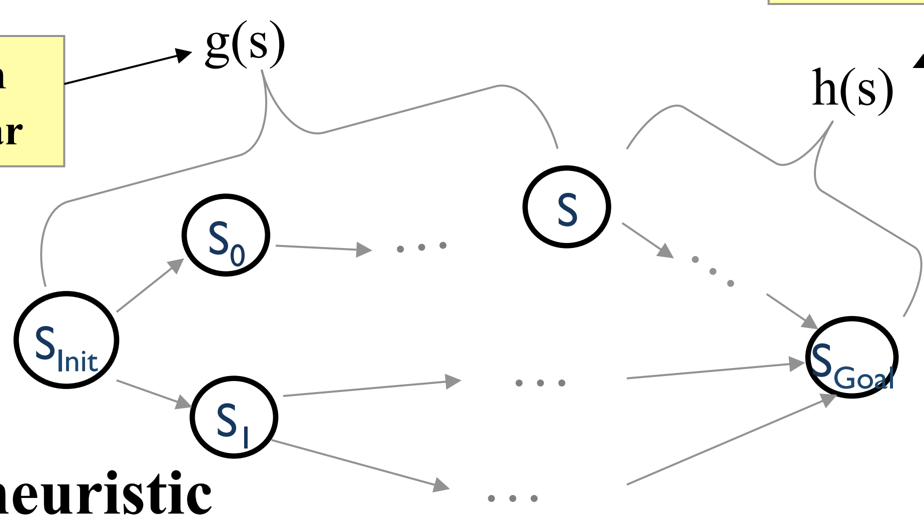
The cost of each state s = g(s) + h(s)

Where *g*(*s*) is an optimal running cost for state s and

h(s) is an (under)estimated cost to reach the goal state from state s.

an (under) estimate of the cost of a shortest path from s to s_{goal}

the cost of a shortest path from s_{Init} to s **found so far**



h(s) is called a **heuristic**

What is a good heuristic that under-estimates the cost to go?

A* Search (Heuristic)

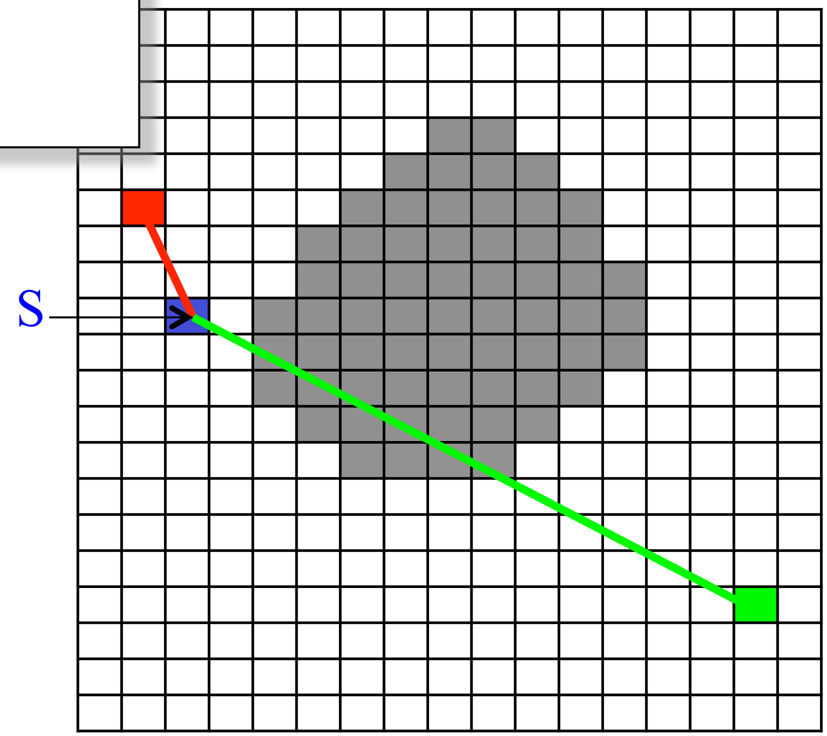
Each state gets a value

$$f(s)=g(s)+h(s)$$

Actual (minimal) cost: $c^*(s,g) = 25$

Estimated cost from here to the goal: "heuristic" cost

Cost incurred from the start



For example (4-connected)

- g(s) = 4
- h(s) = ||s-g||
  $=\mathrm{sqrt}(8^2+13^2)$
  =15.3
- f(s) =19.3

A* Search

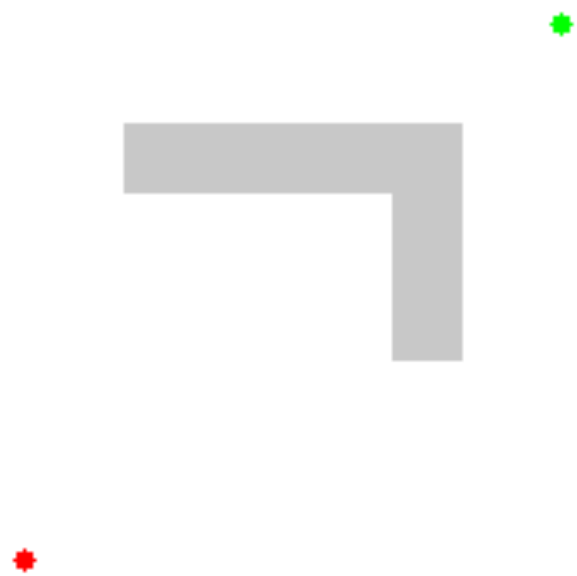
minimal cost from s to s_{Goal}

Heuristic function must be:

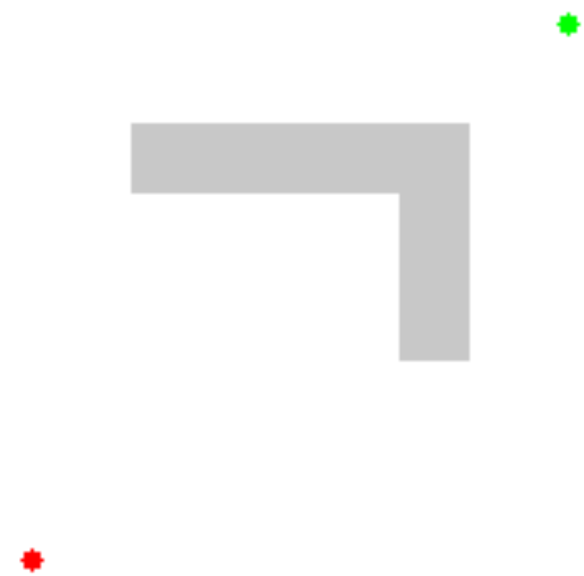
- admissible: for every state s, $h(s) \leq c^*(s, s_{Goal})$
- consistent (satisfy triangle inequality):

  $h(s_{Goal}, s_{Goal}) = 0$ *and for every* $s \neq s_{goal}$, $h(s) \leq c(s, succ(s)) + h(succ(s))$
- Consistency implies admissibility (not necessarily the other way around)

Dijkstra VS A*



Dijkstra



A*

Ref: Wikipedia