Learning Objectives

At the end of the class you should be able to:

- devise an useful heuristic function for a problem
- demonstrate how best-first and A^* search will work on a graph
- predict the space and time requirements for best-first and A^* search

Heuristic Search

- Idea: don't ignore the goal when selecting paths.

Heuristic Search

- Idea: don't ignore the goal when selecting paths.
- Often there is extra knowledge that can be used to guide the search: heuristics.

Heuristic Search

- Idea: don't ignore the goal when selecting paths.
- Often there is extra knowledge that can be used to guide the search: heuristics.
- A heuristic function $h(n)$ is a nonnegative estimate of the cost of the least-cost path from node n to a goal node.

Heuristic Search

- Idea: don't ignore the goal when selecting paths.
- Often there is extra knowledge that can be used to guide the search: heuristics.
- A heuristic function $h(n)$ is a nonnegative estimate of the cost of the least-cost path from node n to a goal node.
- $h(n)$ needs to be efficient to compute.

Heuristic Search

- Idea: don't ignore the goal when selecting paths.
- Often there is extra knowledge that can be used to guide the search: heuristics.
- A heuristic function $h(n)$ is a nonnegative estimate of the cost of the least-cost path from node n to a goal node.
- $h(n)$ needs to be efficient to compute.
- h can be extended to paths: $h(\langle n_0, \ldots, n_k \rangle) = h(n_k)$.

Heuristic Search

- Idea: don't ignore the goal when selecting paths.
- Often there is extra knowledge that can be used to guide the search: heuristics.
- A heuristic function $h(n)$ is a nonnegative estimate of the cost of the least-cost path from node n to a goal node.
- $h(n)$ needs to be efficient to compute.
- h can be extended to paths: $h(\langle n_0, \ldots, n_k \rangle) = h(n_k)$.
- $h(n)$ is an underestimate if there is no path from n to a goal with cost less than $h(n)$.

Heuristic Search

- Idea: don't ignore the goal when selecting paths.
- Often there is extra knowledge that can be used to guide the search: heuristics.
- A heuristic function $h(n)$ is a nonnegative estimate of the cost of the least-cost path from node n to a goal node.
- $h(n)$ needs to be efficient to compute.
- h can be extended to paths: $h(\langle n_0, \ldots, n_k \rangle) = h(n_k)$.
- $h(n)$ is an underestimate if there is no path from n to a goal with cost less than $h(n)$.
- An admissible heuristic is a heuristic function that is an underestimate of the actual cost of a path to a goal.

Example Heuristic Functions

- If the nodes are points on a Euclidean plane and the cost is the distance, $h(n)$ can be

Example Heuristic Functions

- If the nodes are points on a Euclidean plane and the cost is the distance, $h(n)$ can be the straight-line distance from n to the closest goal.

Example Heuristic Functions

- If the nodes are points on a Euclidean plane and the cost is the distance, $h(n)$ can be the straight-line distance from n to the closest goal.
- If the nodes are locations and cost is time, we can use

Example Heuristic Functions

- If the nodes are points on a Euclidean plane and the cost is the distance, $h(n)$ can be the straight-line distance from n to the closest goal.
- If the nodes are locations and cost is time, we can use the distance to a goal divided by the maximum speed.
- If the goal is to collect all of the coins and not run out of fuel,

Example Heuristic Functions

- If the nodes are points on a Euclidean plane and the cost is the distance, $h(n)$ can be the straight-line distance from n to the closest goal.
- If the nodes are locations and cost is time, we can use the distance to a goal divided by the maximum speed.
- If the goal is to collect all of the coins and not run out of fuel, the cost is an estimate of how many steps it will take to collect the rest of the coins, refuel when necessary, and return to goal position.

Example Heuristic Functions

- If the nodes are points on a Euclidean plane and the cost is the distance, $h(n)$ can be the straight-line distance from n to the closest goal.
- If the nodes are locations and cost is time, we can use the distance to a goal divided by the maximum speed.
- If the goal is to collect all of the coins and not run out of fuel, the cost is an estimate of how many steps it will take to collect the rest of the coins, refuel when necessary, and return to goal position.
- A heuristic function can be found by solving a simpler (less constrained) version of the problem.

Heuristic depth-first Search

- Idea: in depth-first search select a neighbor that is closest to a goal according to the heuristic function.

Heuristic depth-first Search

- Idea: in depth-first search select a neighbor that is closest to a goal according to the heuristic function.
- It inherits all of the advantages/disadvantages of depth-first search, but locally heads towards a goal.

Best-first Search

- Idea: select a path whose end is closest to a goal according to the heuristic function.

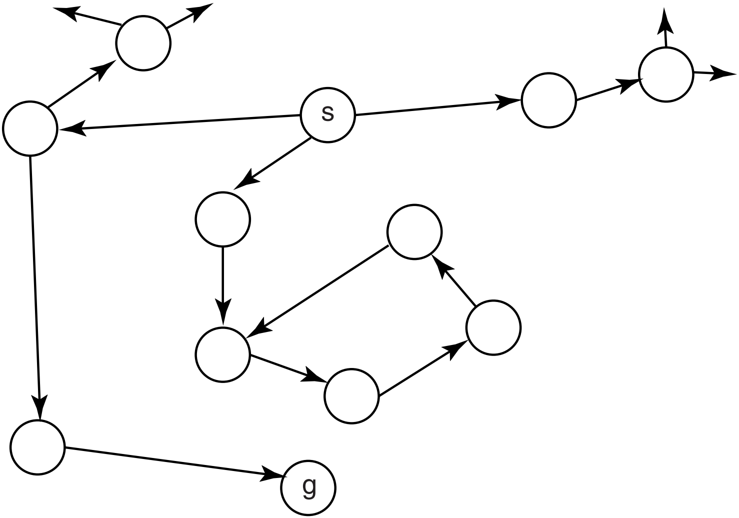
Best-first Search

- Idea: select a path whose end is closest to a goal according to the heuristic function.
- Best-first search selects a path on the frontier with minimal h-value.

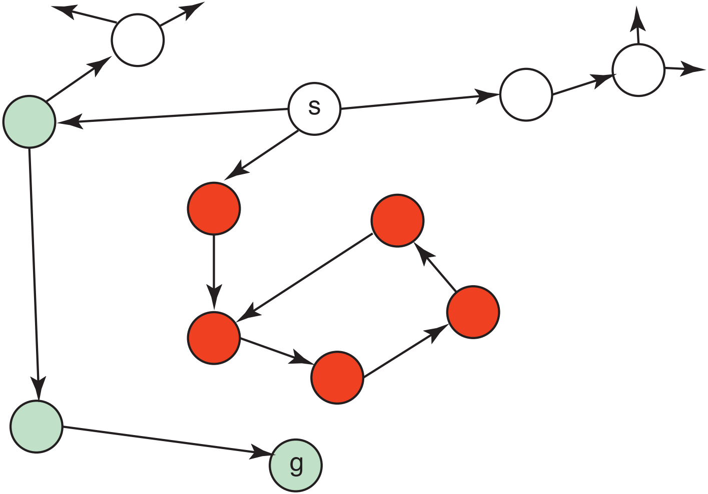
Best-first Search

- Idea: select a path whose end is closest to a goal according to the heuristic function.
- Best-first search selects a path on the frontier with minimal h-value.
- It treats the frontier as a priority queue ordered by h.

Illustrative Graph — Heuristic Search



Illustrative Graph — Heuristic Search



Complexity of Best-first Search

- Does best-first search guarantee to find a least-cost path?

Complexity of Best-first Search

- Does best-first search guarantee to find a least-cost path?
- Does best-first search guarantee to find a path with fewest arcs?

Complexity of Best-first Search

- Does best-first search guarantee to find a least-cost path?
- Does best-first search guarantee to find a path with fewest arcs?
- What happens on infinite graphs or on graphs with cycles if there is a solution?

Complexity of Best-first Search

- Does best-first search guarantee to find a least-cost path?
- Does best-first search guarantee to find a path with fewest arcs?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- What is the time complexity as a function of length of the path selected?

Complexity of Best-first Search

- Does best-first search guarantee to find a least-cost path?
- Does best-first search guarantee to find a path with fewest arcs?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- What is the time complexity as a function of length of the path selected?
- What is the space complexity as a function of length of the path selected?

Complexity of Best-first Search

- Does best-first search guarantee to find a least-cost path?
- Does best-first search guarantee to find a path with fewest arcs?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- What is the time complexity as a function of length of the path selected?
- What is the space complexity as a function of length of the path selected?
- How does the goal affect the search?

A^* Search

- A^* search uses both path cost and heuristic values

A^* Search

- A^* search uses both path cost and heuristic values
- $cost(p)$ is the cost of path p.
- $h(p)$ estimates the cost from the end of p to a goal.

A^* Search

- A^* search uses both path cost and heuristic values
- $cost(p)$ is the cost of path p.
- $h(p)$ estimates the cost from the end of p to a goal.
- Let $f(p) = cost(p) + h(p)$.
  $f(p)$ estimates the total path cost of going from a start node to a goal via p.

$$\underbrace{\underbrace{start \xrightarrow{\text{path } p} n}_{cost(p)} \underbrace{\xrightarrow{\text{estimate}} goal}_{h(p)}}_{f(p)}$$

A^* Search

- A^* search uses both path cost and heuristic values
- $cost(p)$ is the cost of path p.
- $h(p)$ estimates the cost from the end of p to a goal.
- Let $f(p) = cost(p) + h(p)$.
  $f(p)$ estimates the total path cost of going from a start node to a goal via p.

$$\underbrace{\underbrace{start \xrightarrow{\text{path } p} n}_{cost(p)} \underbrace{\xrightarrow{\text{estimate}} goal}_{h(p)}}_{f(p)}$$

- In A^* search, the frontier is a priority queue ordered by $f(p)$.

A^* Search

- A^* search uses both path cost and heuristic values
- $cost(p)$ is the cost of path p.
- $h(p)$ estimates the cost from the end of p to a goal.
- Let $f(p) = cost(p) + h(p)$.
  $f(p)$ estimates the total path cost of going from a start node to a goal via p.

$$\underbrace{\underbrace{start \xrightarrow{\text{path } p} n}_{cost(p)} \underbrace{\xrightarrow{\text{estimate}} goal}_{h(p)}}_{f(p)}$$

- In A^* search, the frontier is a priority queue ordered by $f(p)$.
- It always selects the path on the frontier with the lowest estimated cost from the start to a goal node constrained to go via that path.

A^* Search Algorithm

- A^* is a mix of lowest-cost-first and best-first search.

A^* Search Algorithm

- A^* is a mix of lowest-cost-first and best-first search.
- It treats the frontier as a priority queue ordered by $f(p)$.

A^* Search Algorithm

- A^* is a mix of lowest-cost-first and best-first search.
- It treats the frontier as a priority queue ordered by $f(p)$.
- It always selects the node on the frontier with the lowest estimated distance from the start to a goal node constrained to go via that node.

Complexity of A^* Search

- Does A^* search guarantee to find the path with fewest arcs?

Complexity of A^* Search

- Does A^* search guarantee to find the path with fewest arcs?
- Does A^* search guarantee to find the least-cost path?

Complexity of A^* Search

- Does A^* search guarantee to find the path with fewest arcs?
- Does A^* search guarantee to find the least-cost path?
- What happens on infinite graphs or on graphs with cycles if there is a solution?

Complexity of A^* Search

- Does A^* search guarantee to find the path with fewest arcs?
- Does A^* search guarantee to find the least-cost path?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- What is the time complexity as a function of length of the path selected?

Complexity of A^* Search

- Does A^* search guarantee to find the path with fewest arcs?
- Does A^* search guarantee to find the least-cost path?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- What is the time complexity as a function of length of the path selected?
- What is the space complexity as a function of length of the path selected?

Complexity of A^* Search

- Does A^* search guarantee to find the path with fewest arcs?
- Does A^* search guarantee to find the least-cost path?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- What is the time complexity as a function of length of the path selected?
- What is the space complexity as a function of length of the path selected?
- How does the goal affect the search?

Admissibility of A^*

If there is a solution, A^* always finds an optimal solution – -the first path to a goal selected — if

Admissibility of A^*

If there is a solution, A^* always finds an optimal solution – -the first path to a goal selected — if

- the branching factor is finite

Admissibility of A^*

If there is a solution, A^* always finds an optimal solution – -the first path to a goal selected — if

- the branching factor is finite
- arc costs are bounded above zero (there is some $\epsilon > 0$ such that all of the arc costs are greater than ϵ), and

Admissibility of A^*

If there is a solution, A^* always finds an optimal solution – -the first path to a goal selected — if

- the branching factor is finite
- arc costs are bounded above zero (there is some $\epsilon > 0$ such that all of the arc costs are greater than ϵ), and
- $h(n)$ is nonnegative and an underestimate of the cost of the shortest path from n to a goal node:

$$0 \leq h(n) \leq \text{cost of shortest path from } n \text{ to a goal}$$

Why is A^* admissible?

- If a path p to a goal is selected from the frontier, can there be a lower cost path to a goal?

Why is A^* admissible?

- If a path p to a goal is selected from the frontier, can there be a lower cost path to a goal?
- $h(p) =$

Why is A^* admissible?

- If a path p to a goal is selected from the frontier, can there be a lower cost path to a goal?
- $h(p) = 0$

Why is A^* admissible?

- If a path p to a goal is selected from the frontier, can there be a lower cost path to a goal?
- $h(p) = 0$
- Suppose path p' is on the frontier. Because p was chosen before p', and $h(p) = 0$:

Why is A^* admissible?

- If a path p to a goal is selected from the frontier, can there be a lower cost path to a goal?
- $h(p) = 0$
- Suppose path p' is on the frontier. Because p was chosen before p', and $h(p) = 0$:

$$cost(p) \leq cost(p') + h(p').$$

Why is A^* admissible?

- If a path p to a goal is selected from the frontier, can there be a lower cost path to a goal?
- $h(p) = 0$
- Suppose path p' is on the frontier. Because p was chosen before p', and $h(p) = 0$:

$$cost(p) \leq cost(p') + h(p').$$

- Because h is an underestimate:

  for any path p'' to a goal that extends p'.

Why is A^* admissible?

- If a path p to a goal is selected from the frontier, can there be a lower cost path to a goal?
- $h(p) = 0$
- Suppose path p' is on the frontier. Because p was chosen before p', and $h(p) = 0$:

$$cost(p) \leq cost(p') + h(p').$$

- Because h is an underestimate:

$$cost(p') + h(p') \leq cost(p'')$$

for any path p'' to a goal that extends p'.

Why is A^* admissible?

- If a path p to a goal is selected from the frontier, can there be a lower cost path to a goal?
- $h(p) = 0$
- Suppose path p' is on the frontier. Because p was chosen before p', and $h(p) = 0$:

$$cost(p) \leq cost(p') + h(p').$$

- Because h is an underestimate:

$$cost(p') + h(p') \leq cost(p'')$$

  for any path p'' to a goal that extends p'.
- So $cost(p) \leq cost(p'')$ for any other path p'' to a goal.

Why is A^* admissible?

A^* can always find a solution if there is one:

- The frontier always contains the initial part of a path to a goal, before that goal is selected.

Why is A^* admissible?

A^* can always find a solution if there is one:

- The frontier always contains the initial part of a path to a goal, before that goal is selected.
- A^* halts, as the costs of the paths on the frontier keeps increasing, and will eventually exceed any finite number.

How do good heuristics help?

Suppose c is the cost of an optimal solution. What happens to a path p from a start node, where

- $cost(p) + h(p) < c$

How do good heuristics help?

Suppose c is the cost of an optimal solution. What happens to a path p from a start node, where

- $cost(p) + h(p) < c$
  It will be expanded
- $cost(p) + h(p) > c$

How do good heuristics help?

Suppose c is the cost of an optimal solution. What happens to a path p from a start node, where

- $cost(p) + h(p) < c$
  It will be expanded
- $cost(p) + h(p) > c$
  It will not be expanded
- $cost(p) + h(p) = c$

How do good heuristics help?

Suppose c is the cost of an optimal solution. What happens to a path p from a start node, where

- $cost(p) + h(p) < c$
  It will be expanded
- $cost(p) + h(p) > c$
  It will not be expanded
- $cost(p) + h(p) = c$
  It might or might not be expanded.

How do good heuristics help?

Suppose c is the cost of an optimal solution. What happens to a path p from a start node, where

- $cost(p) + h(p) < c$
  It will be expanded
- $cost(p) + h(p) > c$
  It will not be expanded
- $cost(p) + h(p) = c$
  It might or might not be expanded.

How can a better heuristic function help?

Summary of Search Strategies

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added			
Breadth-first	First node added			
Heuristic depth-first	Local min $h(p)$			
Best-first	Global min $h(p)$			
Lowest-cost-first	Minimal $cost(p)$			
A^*	Minimal $f(p)$			

Complete — if there a path to a goal, it can find one, even on infinite graphs.

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path

Summary of Search Strategies

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	Linear
Breadth-first	First node added	Yes	No	Exp
Heuristic depth-first	Local min $h(p)$	No	No	Linear
Best-first	Global min $h(p)$	No	No	Exp
Lowest-cost-first	Minimal $cost(p)$	Yes	No	Exp
A^*	Minimal $f(p)$	Yes	No	Exp

Complete — if there a path to a goal, it can find one, even on infinite graphs.

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path