Search

- Concepts: state space and search tree
- Basic search
  - finding a goal node
- Local search
  - finding the "fittest" node
- Shortest Path
  - Dynamic Programming
  - Heuristic search
Best-first search (cost, not fitness)

- Each node $n$ has a cost value $f(n)$.
- Lower is better! (don’t confuse with fitness)
- **Frontier** is sorted best-first.
Algorithm A

Best-first search, using the following cost function:

\[ f(n) = \text{estimated shortest path from start to goal via } n \]
\[ = g(n) + \text{length of shortest path to n found so far} + h(n) \text{ [heuristic: estimated distance from n to goal]} \]
Compare

\[ f(n) \text{ [estimated shortest path from start to goal via } n] \]
\[ = \]
\[ g(n) \text{ [length of shortest path to } n \text{ found so far]} \]
\[ + \]
\[ h(n) \text{ [heuristic: estimated distance from } n \text{ to goal]} \]

\[ f^*(n) \text{ [actual shortest distance from start to goal via } n] \]
\[ = \]
\[ g^*(n) \text{ [length of actual shortest path to } n] \]
\[ + \]
\[ h^*(n) \text{ [actual shortest path from } n \text{ to goal]} \]
Algorithm A* [3.5.2]

Best-first search where \( f(n) = g(n) + h(n) \) and:

- \( h \) is optimistic:
  - \( h(n) \leq h^*(n) \) for all nodes \( n \).
- \( h \) is monotonic
  - \( h(n) \leq h(n\') + c(n,n\') \), for all nodes, \( n \), and all successors \( n' \) of \( n \).

A* guarantees that when we have visited the goal, we have found the shortest path.

- A* using tree-search requires optimism.
- A* using graph-search and therefore not revisiting visited nodes requires monotonicity (which implies optimism).
Example

Frontier

Visited

\( S(12) \)
Example

\[
\text{Frontier: } S(12), A(12), B(13) \\
\text{Visited: } S(12)
\]
Example

Visited: S(12) S(12) A(12)
Example

Frontier          Visited
S(12)            S(12)
A(12) B(13)     S(12)
B(13) C(15)     S(12) A(12)
C(13)            S(12) A(12) B(13)
Example

Frontier | Visited
----------|--------
S(12)     | S(12)  
A(12) B(13) | S(12) A(12) 
C(13)     | S(12) A(12) B(17) G(18) ...
G(18)     |
Observe: visit the goal node!

We have found the goal G(9), but need to visit A(7) first.
Observe: visit the goal node!

Visiting A(7), we find G(8).
When we visit G(8) – we know it’s optimal.
Proof (informal)

\[ f(G) = g(G) + h(G) = g(G) + 0 = g(G) = \text{the length of the path to the goal found by } A^*. \]

1. Suppose this path is not optimal. Let \( n \) be the first node on the optimal path that is not visited.
2. Since \( n \) is the first node on the optimal path not visited, when we visit it from the optimal path \( g(n) = g^*(n) \).
3. Since \( h \) is optimistic: \( h(n) \leq h^*(n) \).
4. Since the found path was not optimal:
   \[
   f(G) > f^*(G) = g^*(n) + h^*(n) \geq f(n)
   \]
5. Therefore \( f(n) < f(G) \) and \( n \) should have been visited before \( G \).
Example (not optimistic)

Frontier: S(12), A(12) B(17), C(15) B(17), G(16) B(17)
Visited: S(12), A(12), C(15), G(16)

The high estimate for B blocks the shortest path.
Example (optimistic, not monotonic)

Frontier | Visited
----------|--------
S(12)    | S(12)  
A(12) B(17) | S(12) A(12)  
C(15) B(17) | S(12) A(12) C(15)  
B(17) G(20) | S(12) A(12) C(15)  
A h=4    | B h=10
C h=1    | G h=0
S h=12   |
Selecting the Heuristic

A good estimate is one which is close to the actual value.

- Which nodes will be expanded?
  - If $h$ is monotonic, all nodes where $f(n) = g^*(n) + h(n) < f(G)$
- So if $h$ is higher, fewer nodes are expanded.
- Trade-off:
  - Computing ’smarter’ $h$ can take more time.
Selecting the Heuristic

A good estimate is one which is close to the actual value.

The worst optimistic and monotonic heuristic is $h(n)=0$. This makes A* just look at the cost of getting to nodes and is equivalent to best-first search.
Selecting the Heuristic

A good estimate is one which is close to the actual value.

Getting good estimates is often domain dependent. But if all edges have cost $\geq 1$ then the minimum number of steps between a node and the goal is optimistic and monotonic, and often easy to calculate.

(This might be really useful for project 1! 😊)
Which nodes are expanded?

- If $h$ is monotonic:
  all nodes $n$ with $f(n) = g^*(n) + h(n) < f(G)$
  and some nodes where $f(n) = f(G)$
- If $h$ is higher, fewer nodes are expanded.

- Trade-off:
  computing "smarter" $h$ takes more time.