Outline

- Best-first search
- Greedy best-first search
- A* search
- Heuristics
- Admissibility
Review: Tree search

A search strategy is defined by picking the order of node expansion from the frontier

```
function TREE-SEARCH(problem) returns a solution, or failure
    initialize the frontier using the initial state of problem
    loop do
        if the frontier is empty then return failure
        choose a leaf node and remove it from the frontier
        if the node contains a goal state then return the corresponding solution
        expand the chosen node, adding the resulting nodes to the frontier
```
Best-first search

- An instance of general TREE-SEARCH or GRAPH-SEARCH
- **Idea:** use an evaluation function $f(n)$ for each node $n$
  - estimate of “desirability”
    - Expand most desirable unexpanded node, usually the node with the lowest evaluation
- **Implementation:**
  - Order the nodes in frontier in decreasing order of desirability
- **Special cases:**
  - Greedy best-first search
  - $A^*$-search
Romania with step costs in km

Straight-line distance to Bucharest

- Arad: 366
- Bucharest: 0
- Craiova: 160
- Dobrota: 242
- Eforie: 161
- Fagaras: 176
- Giurgiu: 77
- Hirsova: 151
- Iasi: 226
- Lugoj: 244
- Medias: 241
- Neamt: 234
- Oradea: 380
- Pitesti: 10
- Rimnicu Vilcea: 193
- Sibiu: 253
- Timisoara: 329
- Vaslui: 199
- Zerind: 374

26/01/2017 Michael Herrmann Inf2D 5
Greedy best-first search

- Evaluation function $f(n) = h(n)$ (heuristic)
- $h(n)$: estimated cost of cheapest path from state at node $n$ to a goal state
  - e.g., $h_{\text{SLD}}(n)$: straight-line distance from $n$ to goal (Bucharest)
  - Greedy best-first search expands the node that appears to be closest to goal
What is a heuristic?

- From the greek word “heuriskein” meaning “to discover” or “to find”
- A heuristic is any method that is believed or practically proven to be useful for the solution of a given problem, although there is no guarantee that it will always work or lead to an optimal solution.
- Here we will use heuristics to guide tree search. This may not change the worst case complexity of the algorithm, but can help in the average case.
- We will introduce conditions (admissibility, consistency, see below) in order to identify good heuristics, i.e. those which actually lead to an improvement over uninformed search.
- See also: https://en.wikipedia.org/wiki/Heuristic
Greedy best-first search example
Greedy best-first search example
Greedy best-first search example
Properties of greedy best-first search

- **Complete?** No – can get stuck in loops
  - Graph search version is complete in finite space, but not in infinite ones
- **Time?** $O(b^m)$ for tree version, but a good heuristic can give dramatic improvement
- **Space?** $O(b^m)$ – keeps all nodes in memory
- **Optimal?** No
A* search

- **Idea:** avoid expanding paths that are already expensive
- Evaluation function $f(n) = g(n) + h(n)$
  - $g(n)$: cost so far to reach $n$
  - $h(n)$: estimated cost from $n$ to goal
  - $f(n)$: estimated total cost of path through $n$ to goal
- $A^*$ is both complete and optimal if $h(n)$ satisfies certain conditions
A* search example

Arad

366 = 0 + 366
A* search example
A* search example

![Diagram of A* search example]

- Arad
- Fagaras
- Oradea
- Sibiu
- Timisoara
- Zahind

Costs: 646=280+366, 415=239+176, 671=291+380, 413=220+193, 447=118+329, 449=75+374
A* search example
A* search example
A* search example

- Arad
- Sibiu
- Fagaras
- Oradea
- Rimnicu Vilcea
- Timisoara
- Bucharest
- Craiova
- Pitesti
- Sibiu
- Zerind

Weights:
- Arad: 646 = 280 + 366
- Sibiu: 591 = 338 + 253
- Fagaras: 671 = 291 + 380
- Oradea: 450 = 450 + 0
- Rimnicu Vilcea: 526 = 366 + 160
- Timisoara: 447 = 118 + 329
- Bucharest: 553 = 300 + 253
- Craiova: 615 = 455 + 160
- Pitesti: 607 = 414 + 193
- Zerind: 449 = 75 + 374
A heuristic $h(n)$ is admissible if for every node $n$, $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from $n$.

An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic

- Thus, $f(n) = g(n) + h(n)$ never overestimates the true cost of a solution

Example: $h_{SLD}(n)$ (never overestimates the actual road distance)

**Theorem:** If $h(n)$ is admissible, $A^*$ using TREE-SEARCH is optimal.
Suppose some suboptimal goal $G_2$ has been generated and is in the frontier. Let $n$ be an unexpanded node in the frontier such that $n$ is on a shortest path to an optimal goal $G$.

- $f(G_2) = g(G_2)$ since $h(G_2) = 0$
- $g(G_2) > g(G)$ since $G_2$ is suboptimal
- $f(G) = g(G)$ since $h(G) = 0$
- $f(G_2) > f(G)$ from above
Suppose some suboptimal goal $G_2$ has been generated and is in the fringe. Let $n$ be an unexpanded node in the fringe such that $n$ is on a shortest path to an optimal goal $G$.

- $f(G) < f(G_2)$ from above ($G_2$ is suboptimal)
- $h(n) \leq h^*(n)$ since $h$ is admissible
- $g(n) + h(n) \leq g(n) + h^*(n) = f(G)$
- $f(n) \leq f(G)$

Hence $f(n) < f(G_2) \implies A^*$ will never select $G_2$ for expansion.
A heuristic is **consistent** if for every node $n$, every successor $n'$ of $n$ generated by any action $a$,

$$h(n) \leq c(n, a, n') + h(n')$$

If $h$ is consistent, we have

\[
egin{align*}
  f(n') &= g(n') + h(n') \\
  &= g(n) + c(n, a, n') + h(n') \\
  &\geq g(n) + h(n) \\
  &\geq f(n)
\end{align*}
\]

i.e., $f(n)$ is non-decreasing along any path.

**Theorem:**

If $h(n)$ is consistent, $A^*$ using GRAPH-SEARCH is optimal.
Optimality of $A^*$

- $A^*$ expands nodes in order of increasing $f$ value
- Gradually adds “$f$-contours” of nodes
- Contour $i$ has all nodes with $f = f_i$, where $f_i < f_{i+1}$
Properties of A*

- **Complete?** Yes (unless there are infinitely many nodes with $f \leq f(G)$)
- **Time?** Exponential
- **Space?** Keeps all nodes in memory
- **Optimal?** Yes
Admissible heuristics

Example:

- for the 8-puzzle:
  - \( h_1(n) \): number of misplaced tiles
  - \( h_2(n) \): total Manhattan distance

(i.e., no. of squares from desired location of each tile)

Exercise: Calculate these two values:

- \( h_1(S) = ? \)
- \( h_2(S) = ? \)
If $h_2(n) \geq h_1(n)$ for all $n$ (both admissible) then

- $h_2$ dominates $h_1$
- $h_2$ is better for search

Typical search costs (average number of nodes expanded):

- $d = 12$  
  - IDS = 3,644,035 nodes  
  - $A^*(h_1) = 227$ nodes  
  - $A^*(h_2) = 73$ nodes

- $d = 24$  
  - IDS = too many nodes  
  - $A^*(h_1) = 39,135$ nodes  
  - $A^*(h_2) = 1,641$ nodes
A problem with fewer restrictions on the actions is called a relaxed problem.

The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem.

If the rules of the 8-puzzle are relaxed so that a tile can move anywhere,
- then $h_1(n)$ gives the shortest solution.

If the rules are relaxed so that a tile can move to any adjacent square,
- then $h_2(n)$ gives the shortest solution.

Can use relaxation to automatically generate admissible heuristics.
Smart search based on **heuristic scores**.

- Best-first search
- Greedy best-first search
- $A^*$ search
- Admissible heuristics and optimality.