Informed Search

Introduction to informed search
The A* search algorithm
Designing good admissible heuristics

(AIMA Chapter 3.5.1, 3.5.2, 3.6)

Is Uniform Cost Search the best we can do? Consider finding a route from Bucharest to Arad.

A Better Idea...
- Node expansion based on some estimate of distance to the goal, extending current path

- General approach of informed search:
  - Best-first search: node selected for expansion based on an evaluation function \( f(n) \)
  - \( f(n) \) includes estimate of distance to goal (new idea!)

- Implementation:
  - Sort frontier queue monotonically by this new \( f(n) \).
  - Special cases: greedy search, A* search

Simple, useful estimate: straight-line distances
A heuristic function

- New constraint: evaluation function \( f(n) \geq h(n) \) (heuristic)
  - \( h(n) = \text{estimated cost of the cheapest path from node } n \text{ to goal node.} \)
  - If \( n \) is goal then \( h(n)=0 \)
- Here: \( h_{SLD}(n) = \text{straight-line distance from } n \text{ to Bucharest} \)

Heuristic

Heureka! -- Archimedes

"A rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood."

Heuristic knowledge is useful, but not necessarily correct.
Heuristic algorithms use heuristic knowledge to solve a problem.
A heuristic function here takes a state and returns an estimate of the distance to the goal.

Breadth First for Games, Robots, ...

- Pink: Starting Point
- Blue: Goal
- Teal: Scanned squares
  - Darker: Closer to starting point...

(A great site for practical AI & game Programming)

An optimal informed search algorithm (A*)

- We add a heuristic estimate of distance to the goal
- Yellow: examined nodes with high estimated distance
- Blue: examined nodes with low estimated distance

Breadth first in a world with obstacles

Informed search (A*) in a world with obstacles
Greedy best-first search

- Review: Informed search = Best-first search: select node for expansion with minimal evaluation function \( f(n) \) that includes estimate heuristic \( h(n) \) of the remaining distance to goal

- Greedy best-first search: \( f(n) = h(n) \)
  - Expands the node that is estimated to be closest to goal
  - Ignores cost so far to get to that node \( g(n) \)
  - Here, \( h(n) = h_{SLD}(n) \) = straight-line distance from \( n \) to Bucharest

Greedy best-first search example

- Initial State = Arad
- Goal State = Bucharest

Frontier queue:
- Arad 366
- Hunedoara 317
- Oradea 380
- Sibiu 253
- Zerind 374
- Timisoara 329
- Fagaras 176
- Rimnicu Vilcea 193
- Bucharest 0

Arad 366
Bucharest 0
Sibiu 253
Zerind 374
Hunedoara 317
Oradea 380
Timisoara 329
Fagaras 176
Rimnicu Vilcea 193
Bucharest 0

Goal reached!!

Frontier queue:
- Bucharest 0
- Fagaras 176
- Rimnicu Vilcea 193
- Timisoara 329
- Arad 366
- Zerind 374
- Oradea 380

Greedy best-first search example

Properties of greedy best-first search

- Optimal?
  - No!
  - Found: Arad → Sibiu → Fagaras → Bucharest (450km)
  - Shorter: Arad → Sibiu → Rimnicu Vilcea → Pitesti → Bucharest (418km)
Properties of greedy best-first search

- **Complete?**
  - No – can get stuck in loops,
  - e.g., Iasi → Neamt → Iasi → Neamt → …

A* search

- Best-known form of best-first search.
- Key Idea: avoid expanding paths that are already expensive, but expand most promising first.
- Simple idea: \( f(n) = g(n) + h(n) \)
  - \( g(n) \) the cost (so far) to reach the node
  - \( h(n) \) estimated cost to get from the node to the goal
  - \( f(n) \) estimated total cost of path through \( n \) to goal
- Implementation: Frontier queue as priority queue by increasing \( f(n) \) (as expected…)

Admissible heuristics

- A heuristic \( h(n) \) is **admissible** if it *never overestimates* the cost to reach the goal; i.e. it is **optimistic**
  - Formally: \( \forall n, h(n) \leq h^*(n) \) where \( h^*(n) \) is the true cost from \( n \)
  - \( h(n) \geq 0 \) so \( h(G)=0 \) for any goal \( G \).
- Example: \( h_{SLD}(n) \) never overestimates the actual road distance

Theorem: If \( h(n) \) is admissible, A* using Tree Search is optimal
Optimality of \( A^* \) (Intuitive)

- **Lemma:** \( A^* \) expands nodes on frontier in order of increasing \( f \) value

- Gradually adds "\( f \)-contours" of nodes

- Contour \( i \) has all nodes with \( f = f_i \), where \( f_i < f_{i+1} \)

- (After all, \( A^* \) is just a variant of uniform-cost search,...)

**Optimality of \( A^* \) using Tree-Search (proof idea)**

- **Lemma:** \( A^* \) expands nodes on frontier in order of increasing \( f \) value

- Suppose some suboptimal goal \( G_2 \) (i.e., a goal on a suboptimal path) has been generated and is in the frontier along with an optimal goal \( G \).

  Must prove: \[ f(G_2) > f(G) \]

  (Why? Because if \( f(G_2) > f(G) \), then \( G_2 \) will never get to the front of the priority queue.)

  **Proof:**
  1. \( g(G_2) > g(G) \) since \( G_2 \) is suboptimal
  2. \( f(G_2) = g(G_2) + h(G_2) \) since \( f(G_2) = g(G_2) + h(G_2) \) & \( f(G_2) = 0 \), since \( G_2 \) is a goal similarly
  3. \( f(G) = g(G) \)
  4. \( f(G_2) > f(G) \) from 1, 2, 3

  Also must show that \( G \) is added to the frontier before \( G_2 \) is expanded – see AIMA for argument
A* search, evaluation

- Completeness: YES
  - Since bands of increasing $f$ are added
  - As long as $b$ is finite
    — (guaranteeing that there aren’t infinitely many nodes $n$ with $f(n) < f(G)$)

- Time complexity:
  - Number of nodes expanded is still exponential in the length of the solution.

- Space complexity:
  - It keeps all generated nodes in memory
  - Hence space is the major problem not time

A* search, evaluation

- Completeness: YES
- Time complexity: (exponential with path length)
- Space complexity: (all nodes are stored)
- Optimality: YES
  - Cannot expand $f_{i+1}$ until $f_i$ is finished.
  - $A^*$ expands all nodes with $f(n) < f(G)$
  - $A^*$ expands one node with $f(n) = f(G)$
  - $A^*$ expands no nodes with $f(n) > f(G)$
  - Also optimally efficient (not including ties)

Proof of Lemma: Consistency

- A heuristic is consistent if
  \[ h(n) \leq c(n,a,n') + h(n') \]

- Lemma: If $h$ is consistent,
  \[ f(n') = g(n') + h(n') \]
  \[ = g(n) + c(n,a,n') + h(n') \]
  \[ \geq g(n) + h(n) \]
  \[ \geq f(n) \]

  i.e. $f(n)$ is nondecreasing along any path.

Theorem: If $h(n)$ is consistent, $A^*$ using Graph-Search is optimal

Creating Good Heuristic Functions

AIMA 3.6
Heuristic functions

For the 8-puzzle
- Avg. solution cost is about 22 steps
  -(branching factor ≤ 3)
- Exhaustive search to depth 22: \(3.1 \times 10^{10}\) states
- A good heuristic function can reduce the search process

Admissible heuristics

E.g., for the 8-puzzle:
- \(h_{\text{hop}}(n)\) = number of out of place tiles
- \(h_{\text{md}}(n)\) = total Manhattan distance (i.e., # of moves from desired location of each tile)
- \(h_{\text{hop}}(S) = ?\)
- \(h_{\text{md}}(S) = ?\)

Relaxed problems

- A problem with fewer restrictions on the actions than the original 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 \(h_{\text{hop}}(n)\) gives the shortest solution
- If the rules are relaxed so that a tile can move to any adjacent square, then \(h_{\text{md}}(n)\) gives the shortest solution

Defining Heuristics: \(h(n)\)

- Cost of an exact solution to a relaxed problem (fewer restrictions on operator)

  Constraints on Full Problem:
  - A tile can move from square A to square B if A is adjacent to B and B is blank.

  Constraints on relaxed problems:
  - A tile can move from square A to square B if A is adjacent to B.
  - A tile can move from square A to square B if B is blank.
  - A tile can move from square A to square B. (\(h_{\text{md}}\))

Dominance

- If \(h_2(n) \geq h_1(n)\) for all n (both admissible)
  - then \(h_2\) dominates \(h_1\)
- So \(h_2\) is optimistic, but more accurate than \(h_1\)
  - \(h_2\) is therefore better for search
  - Notice: \(h_{\text{md}}\) dominates \(h_{\text{hop}}\)

Typical search costs (average number of nodes expanded):

- \(d=12\) iterative Deepening Search = 3,644,035 nodes
  - \(A'(h_{\text{hop}}) = 227\) nodes
  - \(A'(h_{\text{md}}) = 73\) nodes
- \(d=24\) IDS = too many nodes
  - \(A'(h_{\text{hop}}) = 39,135\) nodes
  - \(A'(h_{\text{md}}) = 1,641\) nodes
Iterative Deepening A* and beyond

Beyond our scope:

- Iterative Deepening A*
- Recursive best first search (incorporates A* idea, despite name)
- Memory Bounded A*
- Simplified Memory Bounded A* - R&N say the best algorithm to use in practice, but not described here at all.
  - (If interested, follow reference to Russell article on Wikipedia article for SMA*)

(see 3.5.3 if you're interested in these topics)