Artificial Intelligence

CS4365 --- Spring 2018

Informed Search

Reading: Sections 3.5-3.6, R&N

Informed Methods: Heuristic Search

Informed Methods use problem-specific knowledge.

best-first search algorithms: Nodes are selected for expansion based on an *evaluation function*, $f(n)$. Traditionally, f is a cost measure.

Use $h(n)$ = estimated cost of the cheapest path from the state at node n to a goal state (*heuristic function*)
Assumption: $h(n) = 0$ when n is a goal node.

Heuristic search is an attempt to search the most promising paths first. Uses heuristics, or rules of thumb, to find the best node to expand next.

Generic Best-First Search

1. Set L to be the initial node(s) representing the initial state(s).
2. If L is empty, fail. Let n be the node on L that is "most promising" according to f. Remove n from L.
3. If n is a goal node, stop and return it (and the path from the initial node to n).
4. Otherwise, add *successors(n)* to L. Return to step 2.

Two Instantiations of Best-First Search

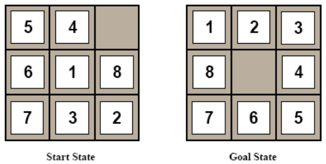
Greedy Best-First Search minimizes estimated cost to reach the goal, i.e., expand the node "closest" to the goal

A* minimizes total estimated path cost to reach the goal, i.e., expand the node on the "least-cost" solution path to the goal

Greedy Best-First Search

Let $f(n) = h(n)$ = estimated cost from node n to nearest goal node

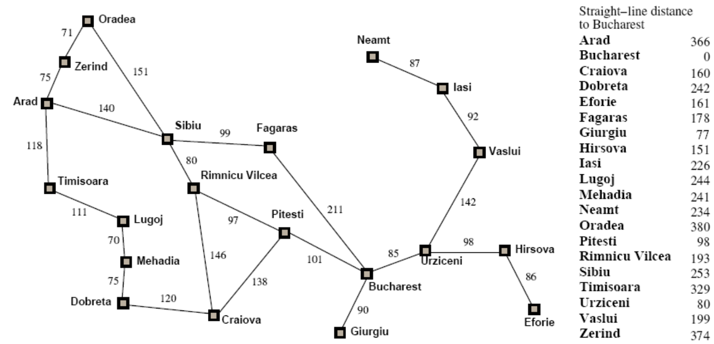
Example: 8-puzzle

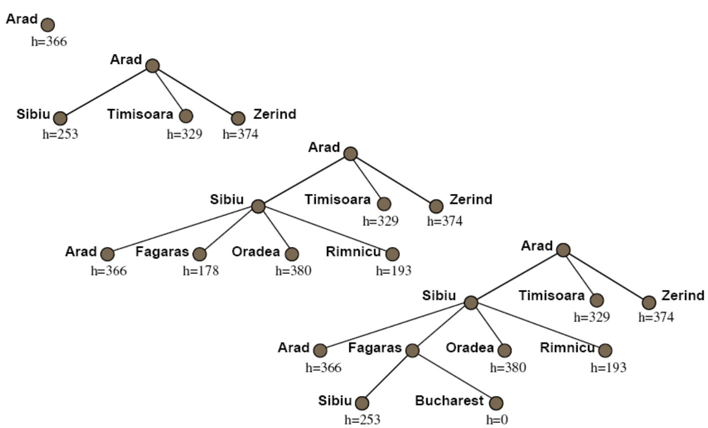


What are the candidates for $h(n)$?

Example

Task: Find a path from Arad to Bucharest





Greedy Best-First Search can be Suboptimal

From Arad to Sibiu to Fagaras --- but to Rimnicu would have been better.

Need to consider: cost of getting from start node (Arad) to intermediate nodes!

A* Search

Goal: Finds the least-cost solution:
Minimizes the total estimated solution cost.

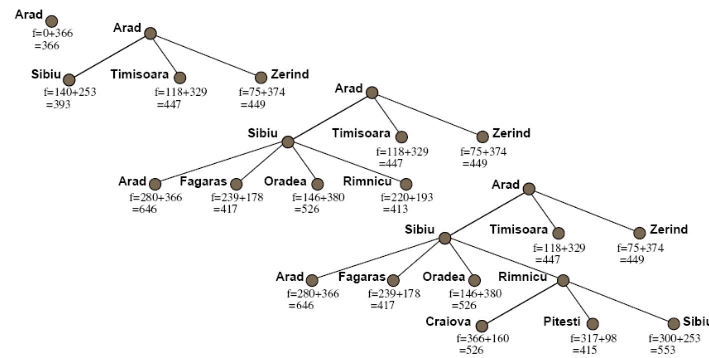
$g(n)$ Cost of reaching node n from initial node

$h(n)$ Estimated cost from node n to nearest goal

A* evaluation function:

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

$f(n)$ Estimated cost of cheapest solution through n



A* Finds Optimal Path

Now expands Rimnicu (f = (140 + 80) + 193 = 413) over Faragas (f = (140 + 99) + 178 = 417).

What if h(Faragas) = 170 (also an underestimate)?

Need Some Conditions

To guarantee that A* finds an optimal solution (and hence never returns a suboptimal goal node), we need that h **never overestimates** the cost of reaching the goal.

Called an *admissible* heuristics.

Transfers to f, i.e., f also doesn't overestimate.

Formal Definition of Admissibility

Let $h^*(n)$ be the *actual* cost to reach a goal from n.

A heuristic function h is **optimistic** or **admissible** if $h(n) \leq h^*(n)$ for all nodes n.

If h is **admissible**, then the A* algorithm will never return a suboptimal goal node. (h **never overestimates** the cost of reaching the goal.)

Example: Admissible Heuristic

What if $h(n) = h^*(n)$?

$$f(n) = g(n) + h^*(n)$$

The perfect heuristic function!

What if $h(n) = 0$?

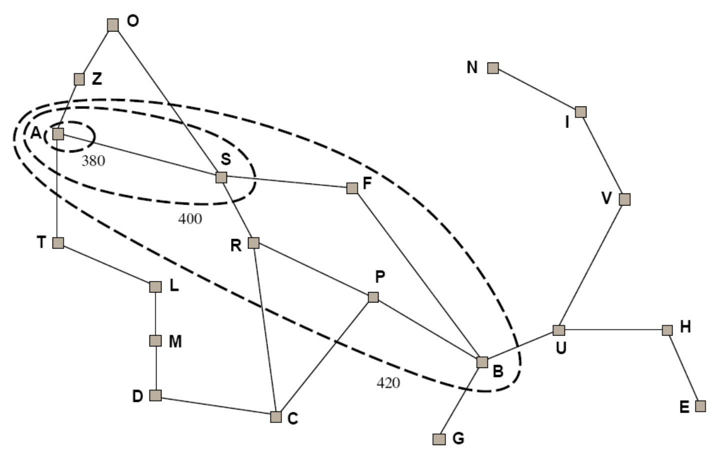
Intuition A*

Let f^* be the cost of the optimal solution path.

We have:

A* expands all nodes with $f(n) < f^*$

A* may then expand some nodes right on "goal contour", with $f(n) = f^*$ before selecting a goal node.



Proving the optimality of A*

Assume h is admissible.

Proof assumes f is non-decreasing along any path from the root.

1. $f = g + h$; g must be non-decreasing because we've disallowed negative costs on operators.
2. That means that the only thing that can happen to make f decrease along a path from the root is that our heuristic function is screwed up.
3. Situation: Node p, with $f = 3 + 4 = 7$; child n, with $f = 4+2 = 6$.

4. But because any path through n is also a path through p, we can see that the value 6 is meaningless, because we already know the true cost is at least 7 (because h is admissible).
5. So, make $f = max(f(p), g(n) + h(n))$

Proof of the optimality of A*

Assume: h admissible; f non-decreasing along any path from the root.

Let G be an optimal goal state, with path cost f^*

Let G_2 be a suboptimal goal state, with path cost $g(G_2) > f^*$

n is a leaf node on an optimal path to G

Because h is admissible, we must have

$$f^* \geq f(n).$$

Also, if n is not chosen over $G2$, we must have

$$f(n) \geq f(G_2).$$

Gives us $f^* \geq f(G_2) = g(G_2)$. (Then G_2 is *not* suboptimal!)

A*

Optimal: yes

A* is **optimally efficient**: given the information in h, no other optimal search method can expand fewer nodes.

Complete: Unless there are infinitely many nodes with $f(n) < f^*$. Assume locally finite:
(1) finite branching, (2) every operator costs at least $\delta > 0$.

Complexity (time and space): Still exponential because of breadth-first nature. Unless $|h(n) - h^*| \leq O(log(h^*(n))$, with h true cost of getting to goal.

8-puzzle

1. h_C = number of misplaced tiles
2. h_M = Manhattan distance

Which one should we use?

$$h_C \leq h_M \leq h^*$$

Importance of $h(n)$

$$h_C \leq h_M \leq h^*$$

Prefer h_M.

Note: Expand all nodes with $f(n) = g(n) + h(n) < f^*$?

So, $g(n) < f^* - h(n)$, higher h means fewer n's.

Aside. How would we get an h_{opt}?

Comparison of Search Costs on 8-Puzzle

	Search Cost			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Inventing Heuristics

Automatically

A tile can move from sq A to sq B if
A is adjacent to B and B is blank.

(a) A tile can move from sq A to sq B if A is adjacent to B.

(b) A tile can move from sq A to sq B if B is blank.

(c) A tile can move from sq A to sq B.

If all admissible, combine them by taking the *max*.