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.

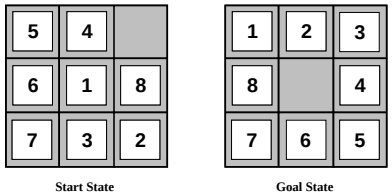
Generic Best-First Search

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.

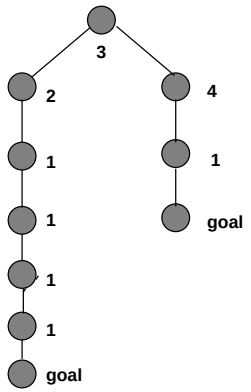
Greedy Best-First Search

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

Example: 8-puzzle



Suboptimal Best-First Search



There exist strategies that enable optimal paths to be found without examining all possible paths.

A* Search

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

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

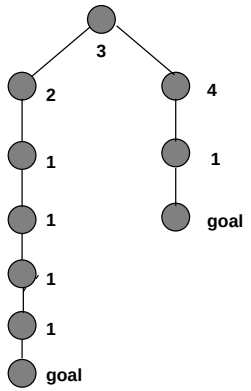
$h(n)$ Estimated cost from node n to nearest goal

A* evaluation function:

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

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

Example



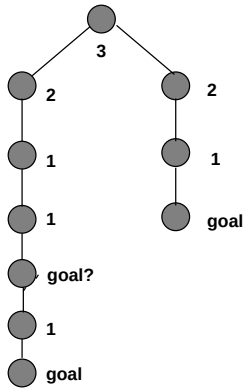
Admissibility

$h^*(n)$ *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



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 G_2, 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 $f(n) < f^\star$. 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 $|h(n) - h^\star| \leq O(log(h^\star(n))$, with $h^\star$ true cost of getting to goal.

IDA*

Memory is a problem for the A* algorithms.

IDA* is like iterative deepening, but uses an f-cost limit rather than a depth limit.

At each iteration, the cutoff value is the smallest f-cost of any node that exceeded the cutoff on the previous iteration.

Each iteration uses conventional depth-first search.

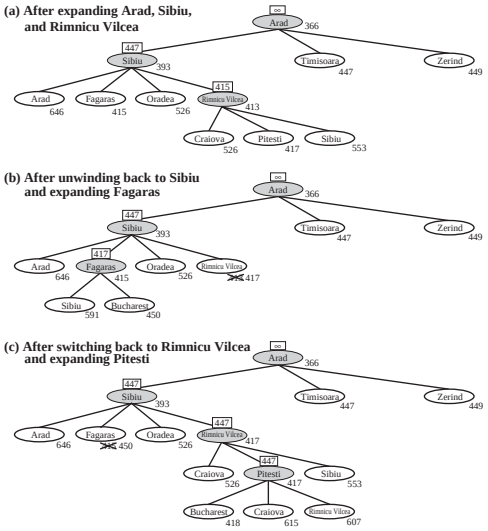
Recursive best-first search (RBFS)

Similar to a recursive DFS, but keeps track of the f-value of the best alternative path available from any ancestor of the current node.

If current node exceeds this limit, recursion unwinds back to the alternative path, replacing the f-value of each node along the path with the best f-value of its children.

(RBFS remembers the f-value of the best leaf in the forgotten subtree.)

RBFS Example



SMA*

Simplified Memory-bounded A* Search

Proceeds just like A*, expanding the best leaf until memory is full.

Drops the **worst** leaf node — the one the highest f-cost; and stores this value in its parent node.

(Won't know which way to go from this node, but we will have some idea of how worthwhile it is to explore the node.)

Example: Admissible Heuristic

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

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

The perfect heuristic function!

Example: Admissible Heuristic

What if $h(n) = 0$?

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

8-puzzle

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

Which one should we use?

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

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

Constructing Admissible Heuristics

- Use an admissible heuristic derived from a **relaxed** version of the problem.
- Use information from **pattern databases** that store exact solutions to subproblems of the problem.
- Use **inductive learning** methods.