Informed Methods: Heuristic Search

Informed Methods use problem-specific knowledge.

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.

Relies on an *evaluation function* — indicates the desirability of expanding a node. E.g., *path cost*.

$h(n)$ = estimated cost of the cheapest path from the state at node n to a goal state (*heuristic function*)

> Given a list of nodes to be expanded, choose the one that the heuristic function estimates as the closest to the goal.

Best-First Search

1. Set L to be the initial node(s).
2. Let n be the node on L that is "most promising" according to the evaluation function. If L is empty, fail.
3. If n is a goal node, stop and return it (and the path from the initial node to n).
4. Otherwise, remove n from L and add all of n's children to L (labeling each with its path from the initial node). Return to step 2.

Two Instantiations of Best-First Search

Issue: Best-First depends on *evaluation function*.

Alternatives:

Greedy Search minimize estimated cost to reach the goal, i.e., expand the node "closest" to the goal.

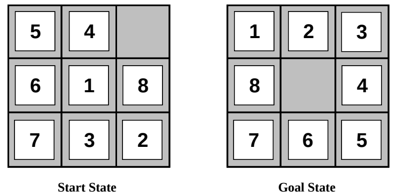
A* minimize total estimated path cost to reach the goal, i.e., expand the node on the "least-cost" solution path to the goal.

Greedy Search

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

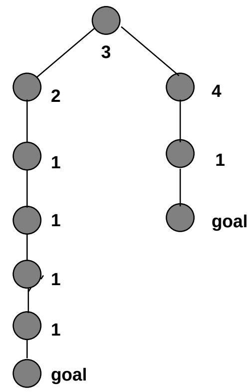
1. Set L to be the initial node(s).
2. Let n be the node on L that minimizes $h(n)$. If L is empty, fail.
3. If n is a goal node, stop and return it (and the path from the initial node to n).
4. Otherwise, remove n from L and add all of n's children to L (labeling each with its path from the initial node). Return to step 2.

8-puzzle



Example

Suboptimal Best-First Search



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

Evaluation Function for A* Algorithm

Goal:

Find the *shallowest*, i.e. min path cost goal as quickly as possible.

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

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

New evaluation function:

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

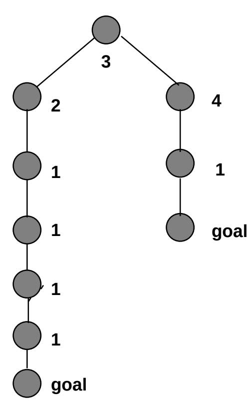
$f(n)$ Estimated cost of cheapest solution through n

A*

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

1. Set L to be the initial node(s).
2. Let n be the node on L that minimizes to $f(n)$. If L is empty, fail.
3. If n is a goal node, stop and return it (and the path from the initial node to n).
4. Otherwise, remove n from L and add all of n's children to L (labeling each with its path from the initial node). Return to step 2.

Example



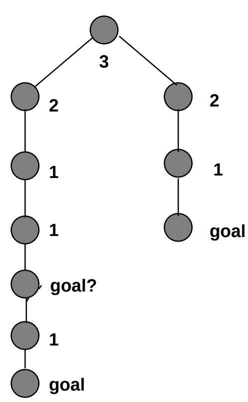
Admissibility

$a(n)$ *Actual* distance from a node n to a goal node.

A heuristic function h is **optimistic** or **admissible** if $h(n) \leq a(n)$ 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



Proof of the optimality of A^*

Assume f is monotonically increasing along any path from the root.

1. $f = g + h$; g must be increasing because we've disallowed negative costs on operators.
2. That means that the only thing that can happen to make f not monotonic 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 f is monotonically increasing 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!)

Observations A*

A* is **optimally efficient**: given the information in h,
no other optimal search method can expand fewer nodes.
Non-trivial and quite remarkable!

Note: A* combines information of cost to get to intermediate nodes (like "uniform cost search") with (under) estimate of cost from intermediate node to goal ("greedy search").

A*

Optimal

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: 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.

See: IDA* (p. 106 R&N, "iterative deepening for A*".)

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.

Each iteration uses conventional depth-first search.

Example: Admissible Heuristic

What if $h(n) = a(n)$?

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

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?

$$h_C \leq h_M \leq a$$

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