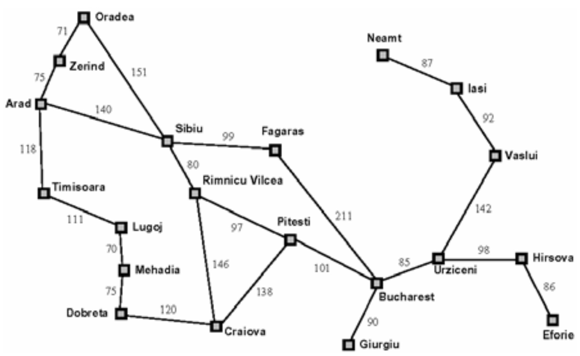
Artificial Intelligence

Informed search

Chapter 4, AIMA

Romania

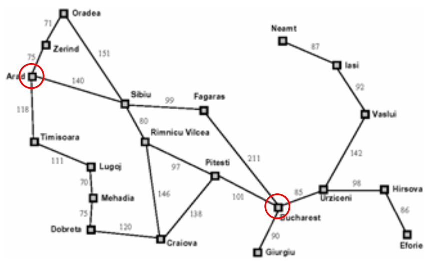
Romania



Romania problem

Initial state: Arad

Find the minimum distance path to Bucharest.



Informed search

Searching for the goal and knowing something about in which direction it is.

Evaluation function: $f(n)$
- Expand the node with minimum $f(n)$

Heuristic function: $h(n)$
- Our estimated cost of the path from node n to the goal.

Example heuristic function $h(n)$

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Dobreta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

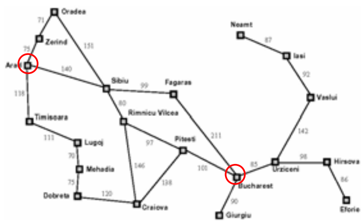
h_{SLD} – Straight-line distances (km) to Bucharest

Greedy best-first (GBFS)

Expand the node that appears to be closest to the goal: $f(n) = h(n)$

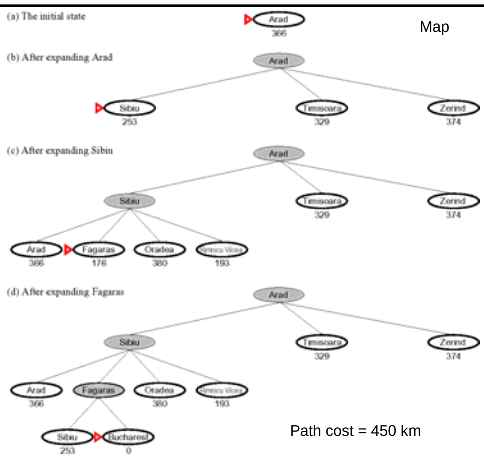
- Incomplete (infinite paths, loops)
- Not optimal (unless the heuristic function is a correct estimate)
- Space and time complexity ~ $O(b^d)$

Assignment: Expand the nodes in the greedy-best-first order, beginning from Arad and going to Bucharest



Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Dobreta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

These are the h(n) values.

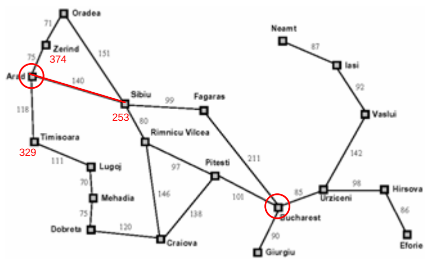


Path cost = 450 km

Romania problem: GBFS

Initial state: Arad

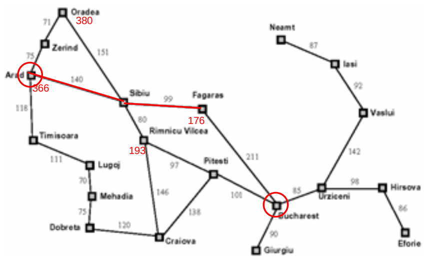
Find the minimum distance path to Bucharest.



Romania problem: GBFS

Initial state: Arad

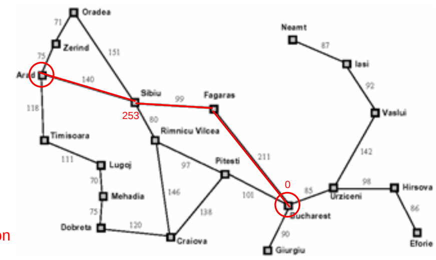
Find the minimum distance path to Bucharest.



Romania problem: GBFS

Initial state: Arad

Find the minimum distance path to Bucharest.



Not the optimal solution

Path cost = 450 km

A and A* best-first search

A: Improve greedy search by discouraging wandering off: $f(n) = g(n) + h(n)$

Here $g(n)$ is the cost to get to node n from the start position.

This penalizes taking steps that don't improve things considerably.

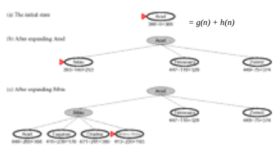
A*: Use an *admissible* heuristic, i.e. a heuristic $h(n)$ that never overestimates the true cost for reaching the goal from node n.

Assignment: Expand the nodes in the A* order, beginning from Arad and going to Bucharest

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Dobreta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

⬆ These are the g(n) values.

⬅ These are the h(n) values.



The straight-line distance never overestimates the true distance; it is an admissible heuristic.

A* on the Romania problem.

Rimnicu-Vilcea is expanded before Fagaras.

The straight-line distance never overestimates the true distance; it is an admissible heuristic.

A* on the Romania problem.

Rimnicu-Vilcea is expanded before Fagaras.

The gain from expanding Rimnicu-Vilcea is too small so the A* algorithm backs up and expands Fagaras.

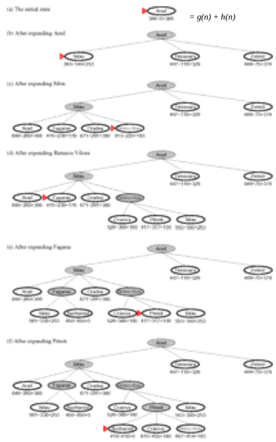
The straight-line distance never overestimates the true distance; it is an admissible heuristic.

A* on the Romania problem.

Rimnicu-Vilcea is expanded before Fagaras.

The gain from expanding Rimnicu-Vilcea is too small so the A* algorithm backs up and expands Fagaras.

None of the descentants of Fagaras is better than a path through Rimnicu-Vilcea; the algorithm goes back to Rimnicu-Vilcea and selects Pitesti.



The straight-line distance never overestimates the true distance; it is an admissible heuristic.

A* on the Romania problem.

Rimnicu-Vilcea is expanded before Fagaras.

The gain from expanding Rimnicu-Vilcea is too small so the A* algorithm backs up and expands Fagaras.

None of the descentants of Fagaras is better than a path through Rimnicu-Vilcea; the algorithm goes back to Rimnicu-Vilcea and selects Pitesti.

The final path cost = 418 km

This is the optimal solution.

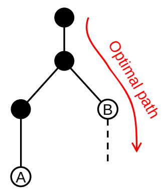
Romania problem: A*

Initial state: Arad

Find the minimum distance path to Bucharest.

The optimal solution
Path cost = 418 km

Theorem: A* tree-search is optimal

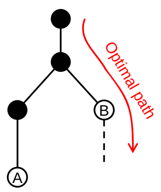


A and B are two nodes on the fringe.

A is a <u>suboptimal</u> goal node and B is a node on the optimal path.

Optimal path cost = C

Theorem: A* tree-search is optimal



A and B are two nodes on the fringe.

A is a <u>suboptimal</u> goal node and B is a node on the optimal path.

Optimal path cost = C

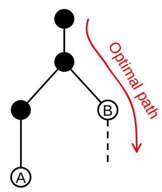
$h(A) = 0$

$$f(A) = g(A) + h(A) = g(A) > C$$

$$f(B) = g(B) + h(B) \leq C$$

$h(n)$ is admissive heuristic

Theorem: A* tree-search is optimal



A and B are two nodes on the fringe.

A is a <u>suboptimal</u> goal node and B is a node on the optimal path.

Optimal path cost = C

$$f(A) = g(A) + h(A) = g(A) > C$$

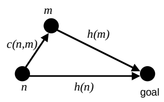
$$f(B) = g(B) + h(B) \leq C$$

$\Rightarrow$ No suboptimal goal node will be selected before the optimal goal node

Is A* graph-search optimal?

- Previous proof works only for tree-search
- For graph-search we add the requirement of *consistency* (monotonicity):

$$h(n) \le c(n,m) + h(m)$$



$c(n,m)$ = step cost for going from node n to node m (n comes before m)

A* graph search with consistent heuristic is optimal

Theorem:

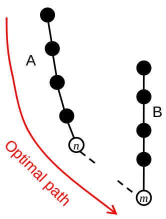
If the consistency condition on $h(n)$ is satisfied, then when A* expands a node n, it has already found an optimal path to n.

This follows from the fact that consistency means that $f(n)$ is nondecreasing along a path in the graph

$$f(m) = g(m) + h(m) = g(n) + c(n,m) + h(m) \ge g(n) + h(n) = f(n)$$

if m comes after n along a path

Proof



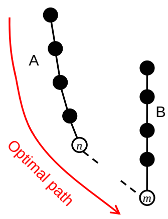
A* has reached node m along the alternative path B.

Path A is the <u>optimal</u> path to node m.

Node n precedes m along the optimal path A.

Both n and m are on the fringe and A* is about to expand m.

Proof



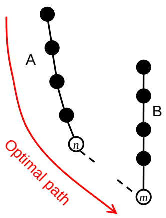
A* has reached node m along the alternative path B.

Path A is the <u>optimal</u> path to node m.
$\Rightarrow g_A(m) \le g_B(m)$

Node n precedes m along the optimal path A.
$\Rightarrow f_A(n) \le f_A(m)$

Both n and m are on the fringe and A* is about to expand m.
$\Rightarrow f_B(m) \le f_A(n)$

Proof



$$f_B(m) = g_B(m) + h(m) \le g_A(n) + h(n) = f_A(n)$$
$$h(n) \le c_A(n,m) + h(m)$$
$$\Rightarrow$$
$$g_B(m) \le g_A(n) + c_A(n,m) = g_A(m)$$

But path A is optimal to reach m why $g_A(m) \le g_B(m)$

Thus, either $m = n$ or contradiction.

$\Rightarrow$ A* graph-search with consistent heuristic always finds the optimal path

A*

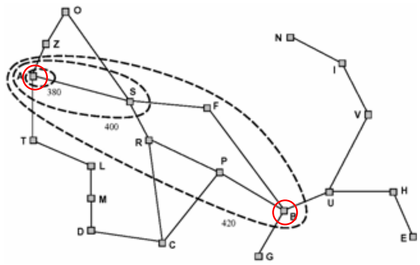
- Optimal
- Complete
- Optimally efficient
  (no algorithm expands fewer nodes)
- Memory requirement exponential...(bad)

- A* expands all nodes with $f(n) < C$
- A* expands some nodes with $f(n) = C$

Romania problem: A*

Initial state: Arad

Find the minimum distance path to Bucharest.



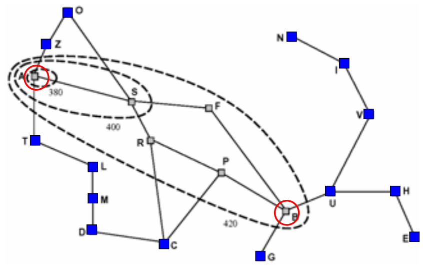
The optimal solution
Path cost = 418 km

Romania problem: A*

Initial state: Arad

Find the minimum distance path to Bucharest.

■ Never tested nodes



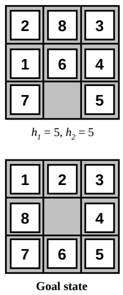
The optimal solution
Path cost = 418 km

Variants of A*

- Iterative deepening A* (IDA*) (uses *f* cost)
- Recursive best-first search (RBFS)
  - Depth-first but keep track of best *f*-value so far above.
- Memory-bounded A* (MA*/SMA*)
  - Drop old/bad nodes when memory gets full.

Best of these is SMA*

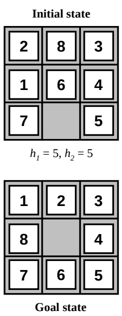
Heuristic functions 8-puzzle



- h_1 = The number of misplaced tiles.
- h_2 = The sum of the distances of the tiles from their respective goal positions (Manhattan distance).

Both are admissive

Heuristic functions 8-puzzle



- h_1 = The number of misplaced tiles.

Assignment: Expand the first three levels of the search tree using A* and the heuristic h_1.

A* on 8-puzzle, h_1 heuristic

Only nodes in shaded area are expanded

Goal reached in node #13

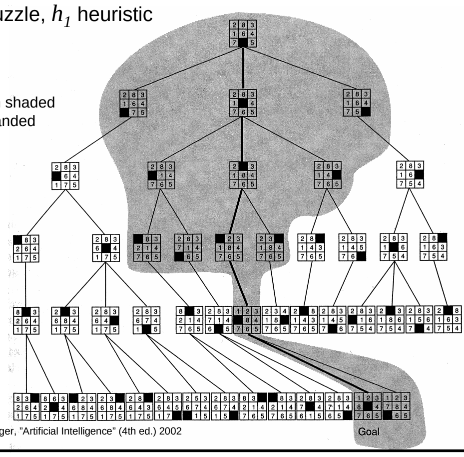


Image from G. F. Luger, "Artificial Intelligence" (4th ed.) 2002

Domination

It is obvious from the definitions that $h_1(n) \le h_2(n)$.
We say that h_2 dominates h_1.

$$h_1(n) \le h_2(n) \le \text{true path cost to node } n$$

All nodes expanded with h_2 are also expanded with h_1 (but not vice versa). Thus, h_2 is better.

Local search

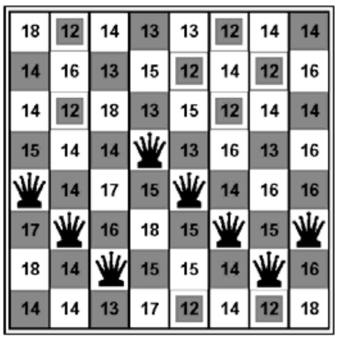
- In many problems, one does not care about the path – only the goal state is of interest.
- Use local searches that only keep track of the last state (saves memory).

Example: N-queens

From initial state (in N × N chessboard), try to move to other configurations such that the number of conflicts is reduced.

Hill-climbing

- Current node = n_i.
- Grab a neighbor node n_{i+1} and move there if it improves things, i.e. if $\Delta f = f(n_i) - f(n_{i+1}) > 0$



Heuristic: Number of pairs of queens that threat each other.
Best moves are marked.

Simulated annealing

- Current node = n_i.
- Grab a neighbor node n_{i+1} and move there if there is improvement or if the decrease is small in relation to the ”temperature”. Accept the move with probability p

$$p = \min[1, \exp(-\Delta f / T)]$$

(This is a common and useful algorithm)

Yields Boltzmann statistics

$$f(t+1) = f(t) + \Delta f(t)$$

Local beam search

- Start with k random states
- Expand all k states and test their children states.
- Keep the k best children states
- Repeat until goal state is found

Genetic algorithms

- Start with k random states
- Selective breeding by mating the best states (with mutation)