

INFORMED SEARCH ALGORITHMS

CHAPTER 4, SECTIONS 1–2

Best-first search

Idea: use an **evaluation function** for each node
– estimate of “desirability”

⇒ Expand most desirable unexpanded node

Implementation:

fringe is a queue sorted in decreasing order of desirability

Special cases:

greedy search

A* search

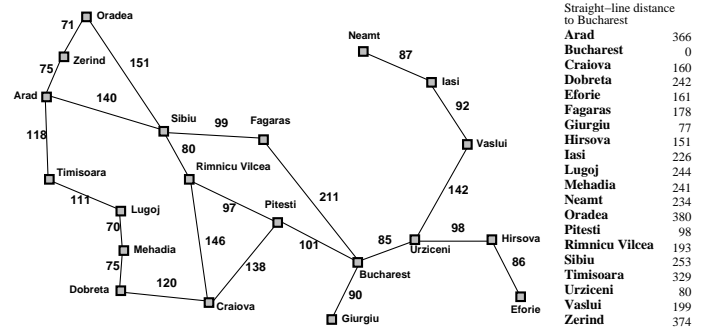
Chapter 4, Sections 1–2 1

Chapter 4, Sections 1–2 4

Outline

- ◇ Best-first search
- ◇ A* search
- ◇ Heuristics

Romania with step costs in km



Chapter 4, Sections 1–2 2

Chapter 4, Sections 1–2 5

Review: Tree search

```
function TREE-SEARCH(problem, fringe) returns a solution, or failure
  fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node ← REMOVE-FRONT(fringe)
    if GOAL-TEST[problem] applied to STATE(node) succeeds return node
    fringe ← INSERTALL(EXPAND(node, problem), fringe)
```

A strategy is defined by picking the **order of node expansion**

Greedy search

Evaluation function $h(n)$ (**h**euristic)

= estimate of cost from n to the closest goal

E.g., $h_{SLD}(n)$ = straight-line distance from n to Bucharest

Greedy search expands the node that **appears** to be closest to goal

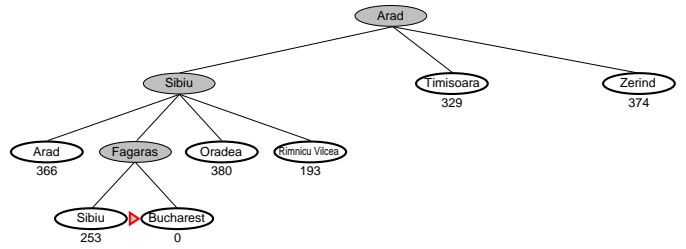
Chapter 4, Sections 1–2 3

Chapter 4, Sections 1–2 6

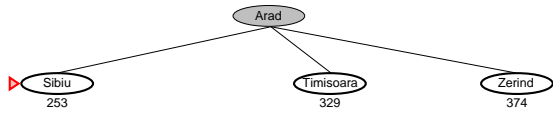
Greedy search example



Greedy search example



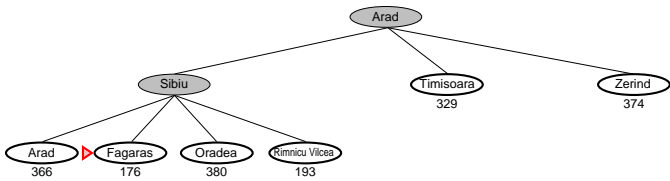
Greedy search example



Properties of greedy search

Complete??

Greedy search example



Properties of greedy search

Complete?? No—can get stuck in loops, e.g., with Oradea as goal,
 lasi → Neamt → lasi → Neamt →

Complete in finite space with repeated-state checking

Time??

Properties of greedy search

Complete?? No—can get stuck in loops, e.g., with Oradea as goal,
Iasi → Neamt → Iasi → Neamt →
Complete in finite space with repeated-state checking

Time?? $O(b^m)$, but a good heuristic can give dramatic improvement

Space??

Properties of greedy search

Complete?? No—can get stuck in loops, e.g., with Oradea as goal,
Iasi → Neamt → Iasi → Neamt →
Complete in finite space with repeated-state checking

Time?? $O(b^m)$, but a good heuristic can give dramatic improvement

Space?? $O(b^m)$ —keeps all nodes in memory

Optimal??

Properties of greedy search

Complete?? No—can get stuck in loops, e.g., with Oradea as goal,
Iasi → Neamt → Iasi → Neamt →
Complete in finite space with repeated-state checking

Time?? $O(b^m)$, but a good heuristic can give dramatic improvement

Space?? $O(b^m)$ —keeps all nodes in memory

Optimal?? No

A* search

Idea: avoid expanding paths that are already expensive

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

$g(n)$ = cost so far to reach n

$h(n)$ = estimated cost to goal from n

$f(n)$ = estimated total cost of path through n to goal

A* search uses an **admissible** heuristic

i.e., $h(n) \leq h^*(n)$ where $h^*(n)$ is the **true** cost from n .

(Also require $h(n) \geq 0$, so $h(G) = 0$ for any goal G .)

E.g., $h_{\text{SLD}}(n)$ never overestimates the actual road distance

Theorem: A* search is optimal

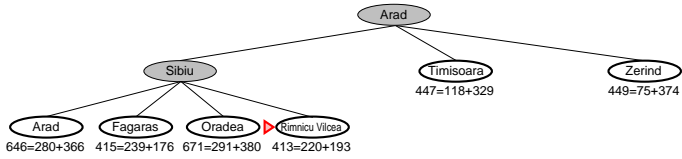
A* search example



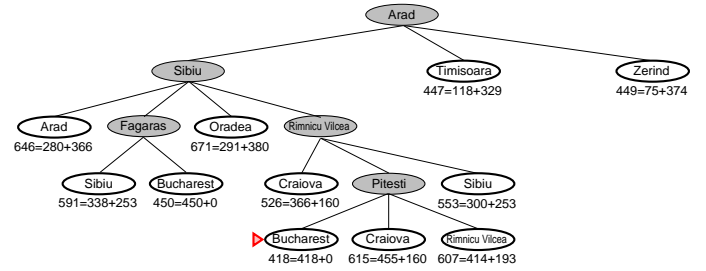
A* search example



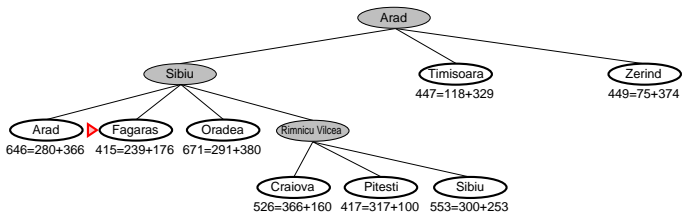
A* search example



A* search example

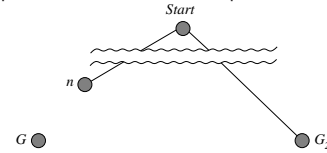


A* search example



Optimality of A* (standard proof)

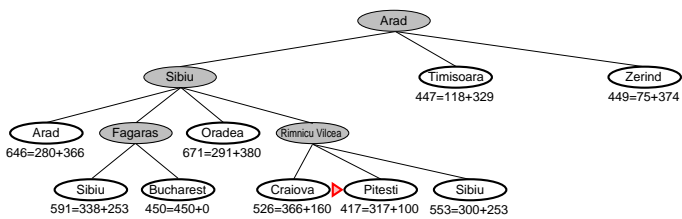
Suppose some suboptimal goal G_2 has been generated and is in the queue. Let n be an unexpanded node on a shortest path to an optimal goal G_1 .



$$\begin{aligned}
 f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\
 &> g(G_1) && \text{since } G_2 \text{ is suboptimal} \\
 &\geq f(n) && \text{since } h \text{ is admissible}
 \end{aligned}$$

Since $f(G_2) > f(n)$, A* will never select G_2 for expansion

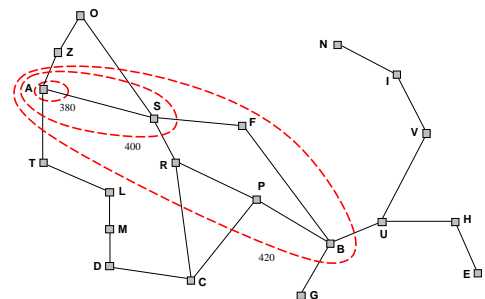
A* search example



Optimality of A* (more useful)

Lemma: A* expands nodes in order of increasing f value*

Gradually adds " f -contours" of nodes (cf. breadth-first adds layers)
Contour i has all nodes with $f = f_i$, where $f_i < f_{i+1}$



Properties of A*

Complete??

Chapter 4, Sections 1-2 25

Properties of A*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$

Time?? Exponential in [relative error in $h \times$ length of soln.]

Space?? Keeps all nodes in memory

Optimal??

Chapter 4, Sections 1-2 28

Properties of A*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$

Time??

Chapter 4, Sections 1-2 26

Properties of A*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$

Time?? Exponential in [relative error in $h \times$ length of soln.]

Space?? Keeps all nodes in memory

Optimal?? Yes—cannot expand f_{i+1} until f_i is finished

If h is consistent:

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

A* expands some nodes with $f(n) = C^*$

A* expands no nodes with $f(n) > C^*$

Chapter 4, Sections 1-2 29

Properties of A*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$

Time?? Exponential in [relative error in $h \times$ length of soln.]

Space??

Chapter 4, Sections 1-2 27

Proof of lemma: Consistency

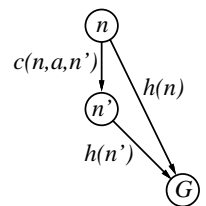
A heuristic is consistent if

$$h(n) \leq c(n, a, n') + h(n')$$

If h is consistent, we have

$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n, a, n') + h(n') \\ &\geq g(n) + h(n) \\ &= f(n) \end{aligned}$$

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



Chapter 4, Sections 1-2 30

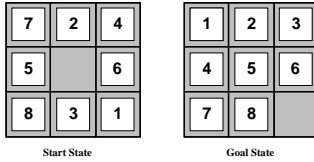
Admissible heuristics

E.g., for the 8-puzzle:

$h_1(n)$ = number of misplaced tiles

$h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)



$h_1(S) = ??$

$h_2(S) = ??$

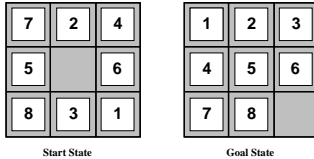
Admissible heuristics

E.g., for the 8-puzzle:

$h_1(n)$ = number of misplaced tiles

$h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)



$h_1(S) = ??$ 6

$h_2(S) = ??$ $4+0+3+3+1+0+2+1 = 14$

Dominance

If $h_2(n) \geq h_1(n)$ for all n (both admissible)
then h_2 **dominates** h_1 and is better for search

Typical search costs:

$d = 14$ IDS = 3,473,941 nodes

$A^*(h_1) = 539$ nodes

$A^*(h_2) = 113$ nodes

$d = 24$ IDS $\approx 54,000,000,000$ nodes

$A^*(h_1) = 39,135$ nodes

$A^*(h_2) = 1,641$ nodes

Given any admissible heuristics h_a, h_b ,

$$h(n) = \max(h_a(n), h_b(n))$$

is also admissible and dominates h_a, h_b

Relaxed problems

Admissible heuristics can be derived from the **exact** solution cost of a **relaxed** version of the problem

If the rules of the 8-puzzle are relaxed so that a tile can move **anywhere**, then $h_1(n)$ gives the shortest solution

If the rules are relaxed so that a tile can move to **any adjacent square**, then $h_2(n)$ gives the shortest solution

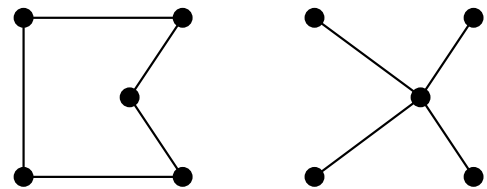
Key point: the optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem

Note: relaxed problem must be easily solvable

Relaxed problems contd.

Well-known example: **travelling salesperson problem** (TSP)

Find the shortest tour visiting all cities exactly once



Minimum spanning tree can be computed in $O(n^2)$ and is a lower bound on the shortest (open) tour

Summary

Heuristic functions estimate costs of shortest paths

Good heuristics can dramatically reduce search cost

Greedy best-first search expands lowest h
– incomplete and not always optimal

A^* search expands lowest $g + h$
– complete and optimal
– also optimally efficient (up to tie-breaks, for forward search)

Admissible heuristics can be derived from exact solution of relaxed problems