4

Informed Search Algorithms

Claudia Chirita

School of Informatics, University of Edinburgh

Based on slides by: Jacques Fleuriot, Michael Rovatsos, Michael Herrmann, Vaishak Belle

4.a

Best-first search

```
function TREE-SEARCH(problem) returns a solution, or failure
  initialize the frontier using the initial state of problem
  loop do
     if the frontier is empty then return failure
     choose a leaf node and remove it from the frontier
     if the node contains a goal state then return the corresponding solution
     expand the chosen node, adding the resulting nodes to the frontier
```

A **search strategy** is defined by picking the order of node expansion from the frontier.

BEST-FIRST SEARCH

An instance of general TREE-SEARCH or GRAPH-SEARCH.

IDEA use an **evaluation function** $f(n)$ for each node n

estimate of "desirability"

$\longrightarrow$ expand most desirable unexpanded node, usually the node with the **lowest** evaluation

IMPLEMENTATION

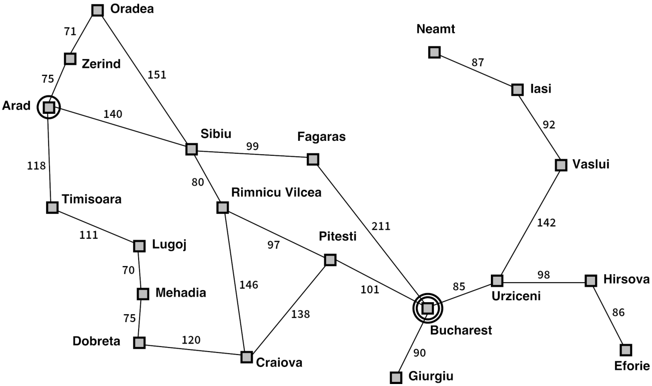
frontier is a queue of nodes sorted in decreasing order of desirability

SPECIAL CASES

Greedy best-first search

A* search

ROMANIA · STEP COSTS IN KM



Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

GREEDY BEST-FIRST SEARCH

Evaluation function $f(n) = h(n)$ (**heuristic**)

estimated cost of cheapest path from state at node n to a goal state

EXAMPLE

$h_{\text{SLD}}(n)$ = straight-line distance from n to goal (Bucharest)

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

HEURISTICS?

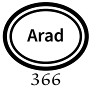
From the greek “*heuriskein*” meaning “*to discover*” or “*to find*”.

Any method that is believed or practically proven to be useful for the solution of a given problem, although there is no guarantee that it will always work or lead to an optimal solution.

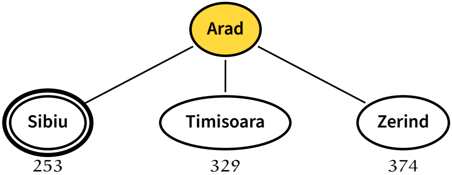
We use heuristics to guide tree search. This may not change the worst case complexity of the algorithm, but can help in the average case.

We introduce conditions (admissibility, consistency) in order to identify good heuristics, i.e. those which actually lead to an improvement over uninformed search.

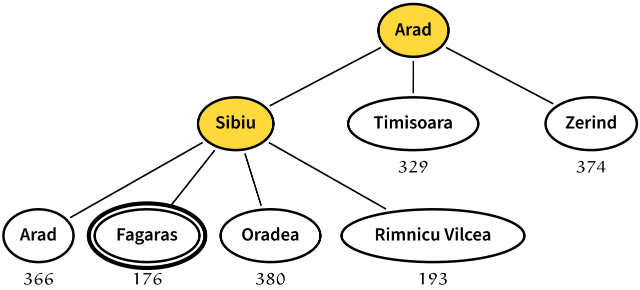
GREEDY SEARCH · EXAMPLE



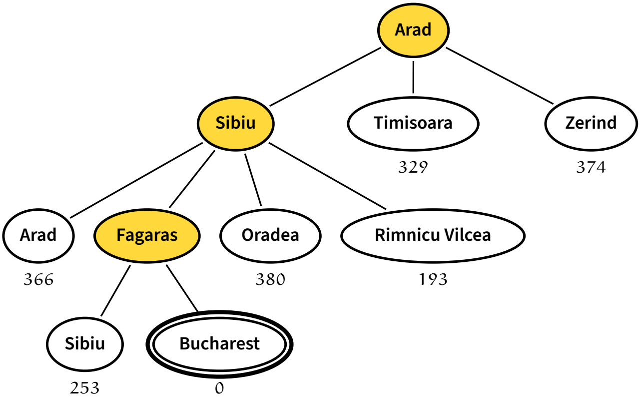
GREEDY SEARCH · EXAMPLE



GREEDY SEARCH · EXAMPLE



GREEDY SEARCH · EXAMPLE



GREEDY SEARCH · PROPERTIES

- complete
- optimal
- time
- space

GREEDY SEARCH · PROPERTIES

complete	No	can get stuck in loops
		Graph search version is complete in finite spaces.
❓ optimal		
time		
space		

GREEDY SEARCH · PROPERTIES

complete	No	can get stuck in loops
		Graph search version is complete in finite spaces.
optimal	No	
time		
space		

GREEDY SEARCH · PROPERTIES

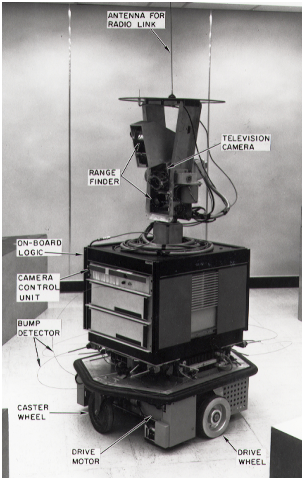
complete	No	can get stuck in loops Graph search version is complete in finite spaces.
optimal	No	
time	$O(b^m)$	for tree version, but a good heuristic can give dramatic improvement
space		

GREEDY SEARCH · PROPERTIES

complete	No	can get stuck in loops Graph search version is complete in finite spaces.
optimal	No	
time	$O(b^m)$	for tree version, but a good heuristic can give dramatic improvement
space	$O(b^m)$	keeps all nodes in memory

4.b

A* search



Shakey the Robot

A* SEARCH

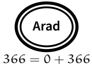
IDEA avoid expanding paths that are already expensive

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

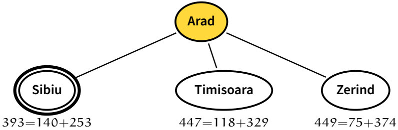
- $g(n)$ cost so far to reach n
- $h(n)$ estimated cost from n to goal
- $f(n)$ estimated total cost of path through n to goal

A* is both complete and optimal if $h(n)$ satisfies certain conditions.

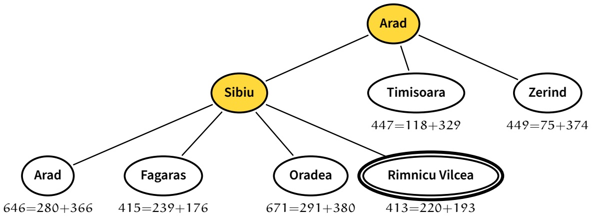
A* SEARCH · EXAMPLE



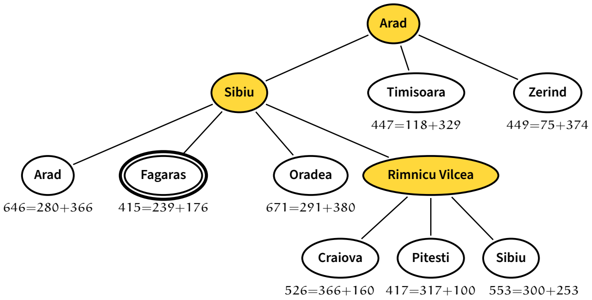
A* SEARCH · EXAMPLE



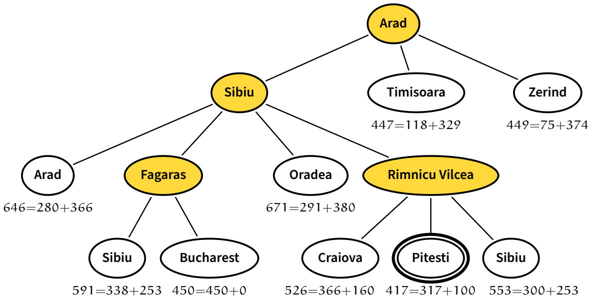
A* SEARCH · EXAMPLE



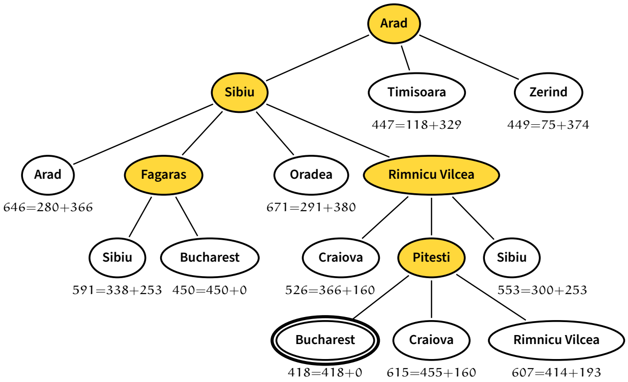
A* SEARCH · EXAMPLE



A* SEARCH · EXAMPLE



A* SEARCH · EXAMPLE



ADMISSIBLE HEURISTICS

A heuristic $h(n)$ is **admissible** if for every node n, $h(n) \leqslant h^*(n)$, where $h^*(n)$ is the *true* cost to reach the goal state from n.

An admissible heuristic never overestimates the cost to reach the goal, i.e. it is optimistic.

$f(n) = g(n) + h(n)$ never overestimates the true cost of a solution

EXAMPLE

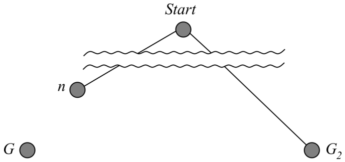
$h_{\text{SLD}}(n)$ never overestimates the actual road distance.

THEOREM

If $h(n)$ is admissible, A* using TREE-SEARCH is **optimal**.

OPTIMALITY OF A* · PROOF

Suppose some suboptimal goal G_2 has been generated and is in the frontier. Let n be an unexpanded node in the frontier such that it is on a shortest path to an optimal goal G.



$f(G_2) = g(G_2)$ since $h(G_2) = 0$

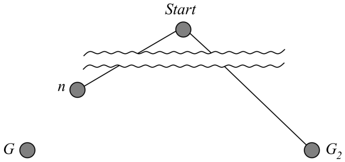
$f(G) = g(G)$ since $h(G) = 0$

$g(G_2) > g(G)$ since G_2 is suboptimal

$f(G_2) > f(G)$

OPTIMALITY OF A* · PROOF

Suppose some suboptimal goal G_2 has been generated and is in the frontier. Let n be an unexpanded node in the frontier such that it is on a shortest path to an optimal goal G.



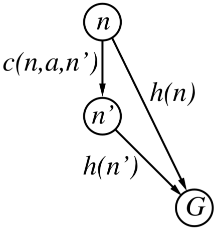
$$f(G) < f(G_2)$$
$$h(n) \leqslant h^*(n) \qquad \text{since } h \text{ is admissible}$$
$$g(n) + h(n) \leqslant g(n) + h^*(n) = f(G)$$
$$f(n) \leqslant f(G)$$

Hence $f(n) < f(G_2) \rightarrow$ A* will never select G_2 for expansion.

CONSISTENT HEURISTICS

A heuristic is **consistent** if for every node n, every successor n' of n generated by any action a,

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



If h is consistent, we have

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

i.e. $f(n)$ is non-decreasing along any path.

THEOREM

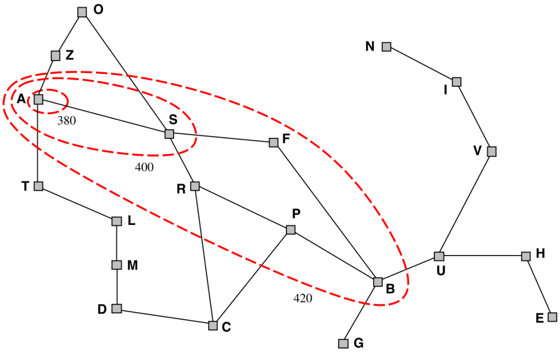
If $h(n)$ is consistent, A* using GRAPH-SEARCH is optimal.

OPTIMALITY OF A*

A* expands nodes in order of increasing f value.

Gradually adds “f-contours” of nodes.

Contour i has all nodes with $f = f_i$, where $f_i < f_{i+1}$.



A* · PROPERTIES

- complete
- optimal
- time
- space

A* · PROPERTIES

complete	Yes	unless there are infinitely many nodes with $f \leqslant f(G)$
optimal		
time		
space		

A* · PROPERTIES

complete	Yes	unless there are infinitely many nodes with $f \leqslant f(G)$
optimal	Yes	
time		
space		

A* · PROPERTIES

complete	Yes	unless there are infinitely many nodes with $f \leqslant f(G)$
optimal	Yes	
time	Exponential	
space		

A* · PROPERTIES

complete	Yes	unless there are infinitely many nodes with $f \leqslant f(G)$
optimal	Yes	
time	Exponential	
space	Keeps all nodes in memory	

4.c

Admissible heuristics

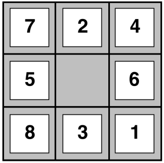
ADMISSIBLE HEURISTICS

EXAMPLE **8-PUZZLE**

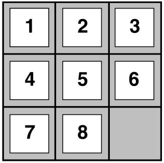
$h_1(n)$ number of misplaced tiles

$h_2(n)$ total Manhattan distance

(no. of squares from desired location of each tile)



Start State



Goal State

$h_1(s) = ?$

$h_2(s) = ?$

DOMINANCE

If $h_2(n) \geqslant h_1(n)$ for all n (both admissible) then

h_2 dominates h_1

h_2 is better for search

Typical search costs (average number of nodes expanded)

$d = 14$ 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

RELAXED PROBLEMS

A **relaxed problem** is a problem with fewer restrictions on the actions.

The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem.

EXAMPLE

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.

IDEA Use relaxation to automatically generate admissible heuristics.

SAY AGAIN?

Relaxing is a good idea!

Play with the interactive animations in the exploratory explanation

The Explanation & Comparison of Graph Searches

by Logan Ringer, Ryan Turner, and Gabe Carroll