

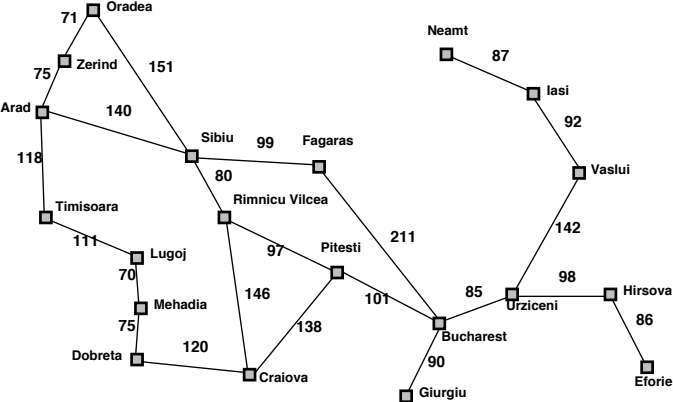
Artificial Intelligence: Heuristic search

Thomas Trappenberg

September 17, 2009

Based on the slides provided by Russell and Norvig, Chapter 4, Section 1–2,(4)

Romania with 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 search

Evaluation function $h(n)$ (**h**euristic)
= estimate of cost from n to the closest goal

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

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

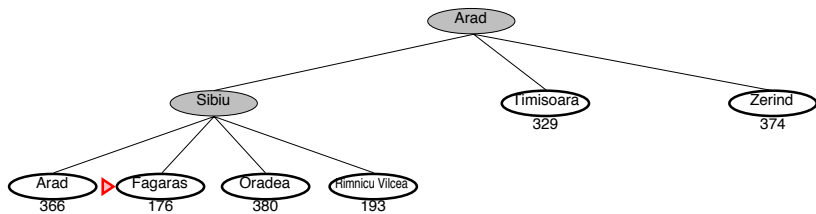
Greedy search example

▶ Arad
366

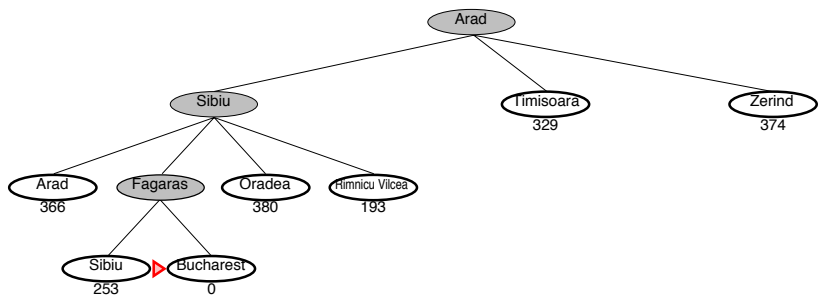
Greedy search example



Greedy search example



Greedy search example



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 .)

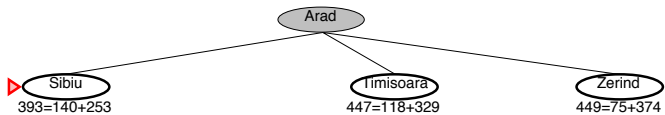
E.g., $h_{\text{SLD}}(n)$ never overestimates the actual road distance

Theorem: A* search is optimal

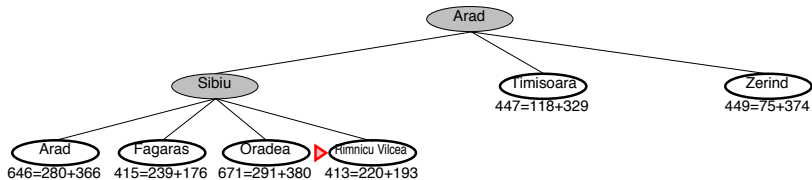
A* search example

Arad
 $366 = 0 + 366$

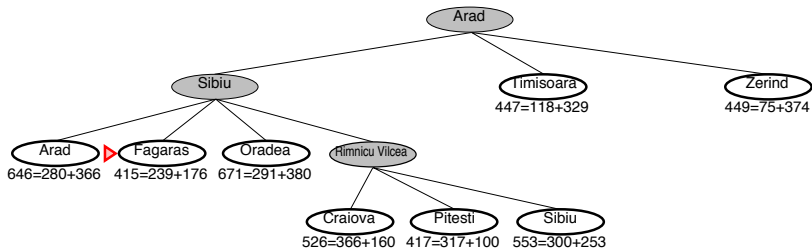
A* search example



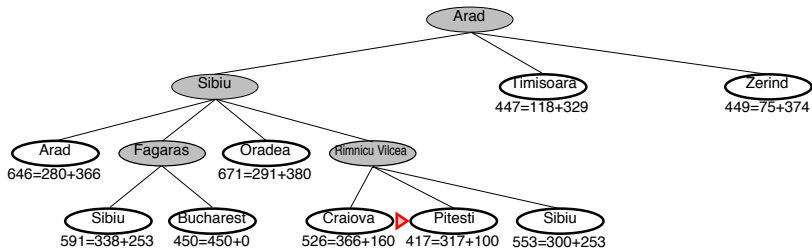
A* search example



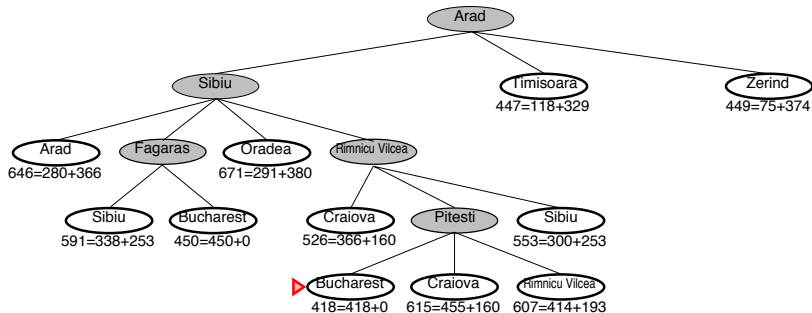
A* search example



A* search example



A* search example



Local beam search

Idea: keep k states instead of 1; choose top k of all their successors

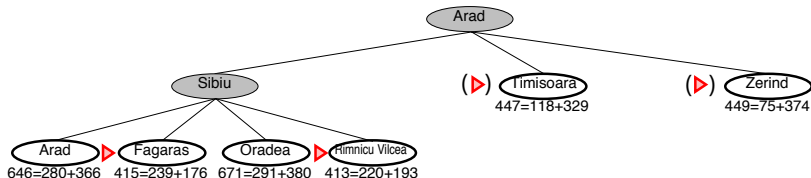
Not the same as k searches run in parallel!

Searches that find good states recruit other searches to join them

Problem: quite often, all k states end up on same local hill

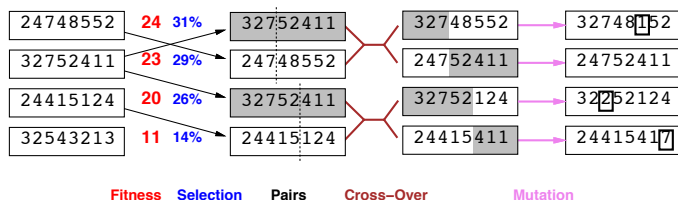
Idea: choose k successors randomly, biased towards good ones

Observe the close analogy to natural selection!



Genetic algorithms

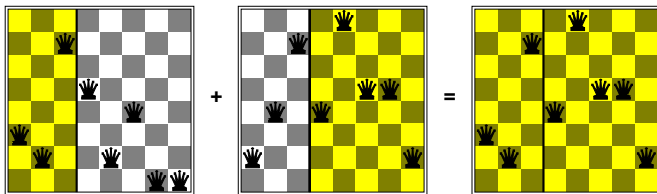
= stochastic local beam search + generate successors from **pairs** of states



Genetic algorithms contd.

GAs require states encoded as strings (GPs use)

Crossover helps **iff substrings are meaningful components**



GAs \neq evolution: e.g., real genes encode replication machinery!

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