Problem Solving by Search 2

Uwe Egly

Vienna University of Technology
Institute of Information Systems
Knowledge-Based Systems Group

Outline

Introduction

Heuristic Search
- Greedy Search
- A^*-search

Admissible Heuristics
- Dominance
- Relaxed Problems

Summary

Overview

Search: very important technique in CS and AI

Different kinds of search:

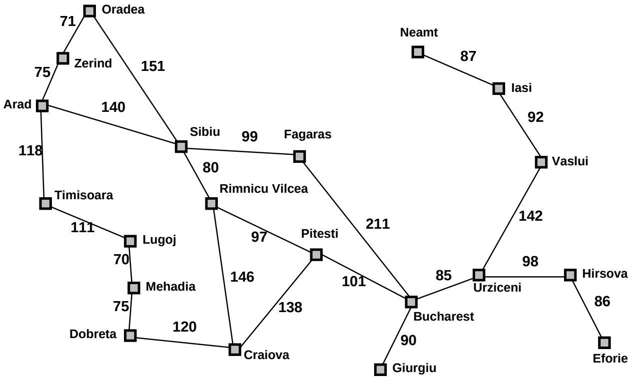
- Deterministic search
  - Uninformed ("blind") search strategies ✔
  - Informed or heuristic search strategies:
    use information about problem structure
- Local search
- Search in game trees (not covered in this course)

In this lecture: Heuristic search

Basic Ideas of Heuristic Search

- Use problem- or domain-specific knowledge during search
- Implemented by a heuristic function $h \rightsquigarrow$ “desirability”
  (expand “most-desired” node next (= a kind of best-first search))
- Evaluation function $f(n)$: estimation for a function $f^*(n)$
- $h(n)$: estimates minimal costs from state n to a goal state
  ($h(n) = 0$ always holds for a goal state)
- Computation of $h(n)$ must be easy
- Consider two algorithms:
  - greedy search and
  - A^* search

Romania Again



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

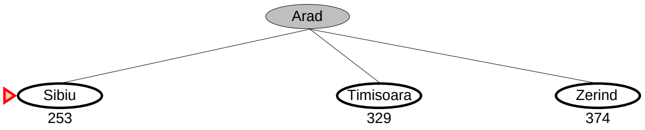
- Uses evaluation function $f(n) = h(n)$
- Does not take already "spent" costs into account
  (decisions are based on local information)
- Example: $h(n)$ = straight-line distance from n to Bucharest
- Greedy search expands the node with the smallest f-value
  (node appears to be closest to goal)

Greedy Search for the Travel Example

Expand the only node Arad

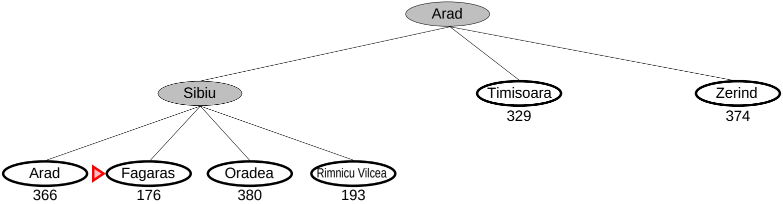
Greedy Search for the Travel Example

Expand the node Sibiu, because it has the smallest *f*-value



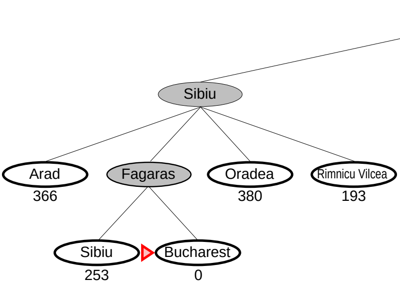
Greedy Search for the Travel Example

Expand the node Fagaras, because it has the smallest *f*-value

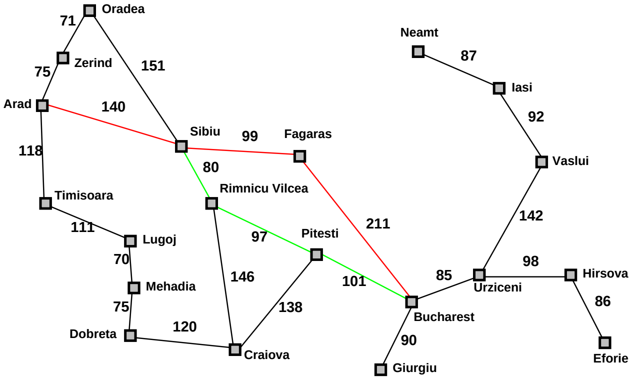


Greedy Search for the Travel Example

Solution found: Arad–Sibiu–Fagaras–Bucharest (450km)



An Alternative and Optimal Solution



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

Non-optimal solution: Arad–Sibiu–Fagaras–Bucharest (450km)
Arad–Sibiu–Rimnicu Vilcea–Pitesti–Bucharest is shorter

Properties of Greedy Search

Completeness:	No (can get stuck in loops) Yes with loop checks
Space complexity: (keep any node)	$O(b^m)$, i.e., exponential in m
Time complexity:	$O(b^m)$, i.e., exponential in m
Optimality:	No

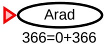
A^*-search

- Problems of greedy search: loops and non-optimality
  (induced by the use of only local info)
- A^*: Use evaluation function $f(n) = g(n) + h(n)$
  (and avoid expanding paths that are already expensive)
  - $g(n)$: path costs from start to n, (i.e., costs so far to reach n)
  - $h(n)$: estimated cost to goal from n (like in greedy search)
  - $f(n)$: estimated total cost of path through n to goal
- $h(n)$ has to be admissible, i.e., for all n, it holds:
  - $h(n) \leq h^*(n)$ where $h^*(n)$ is the true cost from n
    ($h(n)$ is "optimistic")
  - $h(g) = 0$ for any goal g
  - $h(n) > 0$ for any non-goal state n

`http://en.wikipedia.org/wiki/A*_search_algorithm`

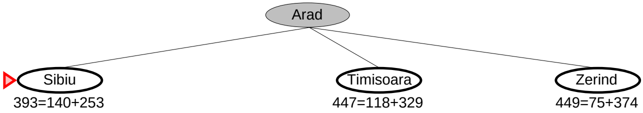
A^*-search for the Travel Example

We use the straight-line distance to obtain an optimistic heuristic
Expand the only node Arad



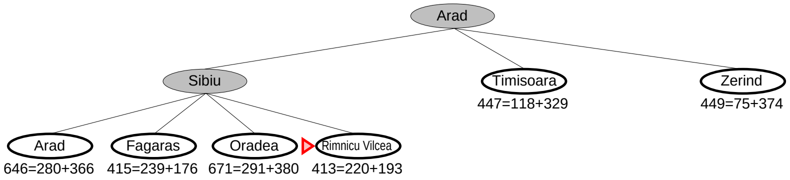
A^*-search for the Travel Example

Expand the node Sibiu, because it has the smallest f-value



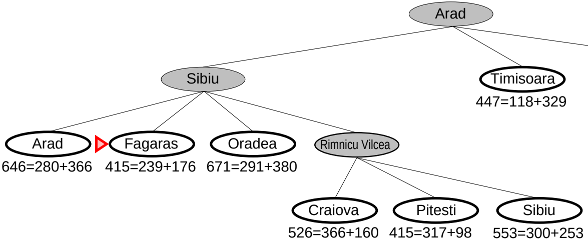
A^*-search for the Travel Example

Expand the node Rimnicu Vilcea (and not Fagaras as in GS)



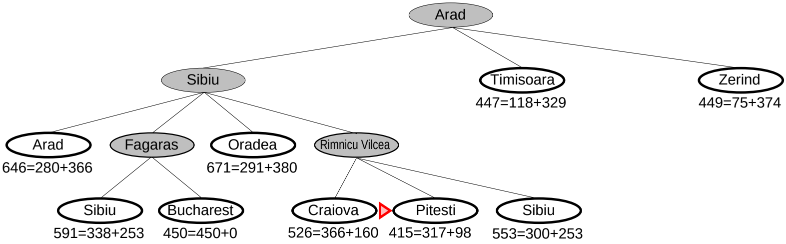
A^*-search for the Travel Example

Expand the node Fagaras, because it has the smallest f-value



A^*-search for the Travel Example

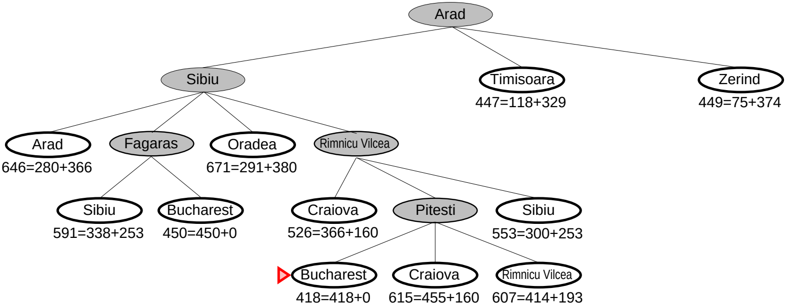
Expand the node Pitesti, because it has the smallest f-value



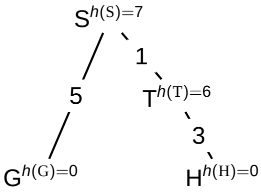
A^*-search for the Travel Example

Solution: Arad–Sibiu–Rimnicu Vilcea–Pitesti–Bucharest

Since f-value of all other open nodes are bigger ⇝ terminate



Do we Really Need Admissible Heuristics?



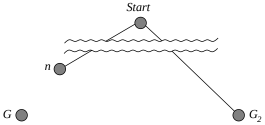
- S is the start state
- *h* is not optimistic
- S expands immediately to G and T
- $f(\mathrm{G}) = 5$ and $f(\mathrm{T}) = 7$, so we are done
- Solution is obviously not optimal
- Hence, heuristics must be optimistic!

Theorem

If h is admissible, then A^* using tree search is optimal.

Optimality of A^*: The Standard Proof

- Let h be admissible, goals G optimal and G_2 suboptimal
- Suppose some suboptimal goal G_2 has been generated
- Let n be an unexpanded node on a shortest path to G



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

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

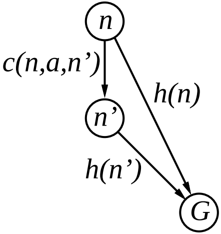
Problems with Optimality of A^* Using Graph Search

- A^* does not require for a path start–n that $g(n)$ is minimal
- No problem for tree search (there is only one path)
- In graph search (GS), we can reach nodes with non-optimal costs
- GS can discard optimal paths even if h is admissible
  ➥ optimality is lost
- Two possibilities to fix the problem:
  - Change algorithm and add more complicated bookkeeping
    (But what's the effect on the run time?)
  - Impose a stronger restriction, consistency, on the heuristic
    ➥ Implies that f-value is non-decreasing on any path

Optimality of A^* Using Graph Search

Definition

A heuristic is consistent if, for every node n, every successor n' of n and any operator a,



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

holds, where $c(n, a, n')$ are the path costs for a.

If h is consistent, then f is non-decreasing along any path, i.e.,

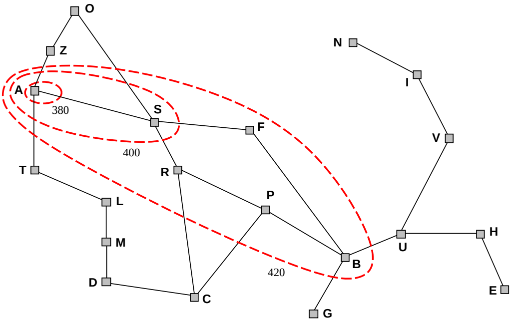
$$\begin{aligned} f(n') &= g(n') + h(n') = g(n) + c(n, a, n') + h(n') \\ &\geq g(n) + h(n) = f(n) \end{aligned}$$

Theorem

If h is consistent, the A^* using graph search is optimal

Optimality of A^*

- A^* expands nodes in order of increasing f-values
- The "f-contours" of nodes are added gradually
  (cf. BFS adds layers)
- Contour i has all nodes with $f = f_i$, where $f_i < f_{i+1}$



Properties of A^*

Completeness: Yes unless there are infinite many nodes with $f \leq f(G)$

Space complexity: Exponential (Keeps all nodes in memory)

Time complexity: Exponential in [relative error in $h \times$ length of solution].

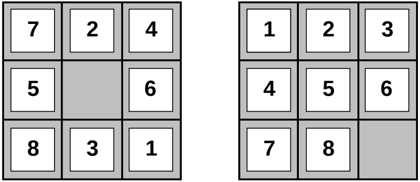
Optimality: Yes

- A^* expands all nodes with $f(n) < C^*$
- A^* expands some nodes with $f(n) = C^*$
- A^* expands no nodes with $f(n) > C^*$

Admissible Heuristics

Consider the following heuristics for the 8-puzzle:

- $h_1(n)$: number of misplaced tiles
- $h_2(n)$: total Manhattan distance
  (no. of squares ($\leftrightarrow$ and $\updownarrow$) from desired location of each tile)



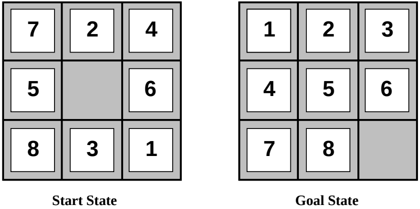
Start State **Goal State**

h_1(Start State) = ?
h_2(Start State) = ?

Admissible Heuristics

Consider the following heuristics for the 8-puzzle:

- $h_1(n)$: number of misplaced tiles
- $h_2(n)$: total Manhattan distance
  (no. of squares ($\leftrightarrow$ and $\updownarrow$) from desired location of each tile)



$h_1(\text{Start State}) = 6$
$h_2(\text{Start State}) = 4+0+3+3+1+0+2+1 = 14$

Dominance

- Given two admissible heuristics h_1 and h_2
- h_2 dominates h_1 (and is better for search) if $h_2(n) \geq h_1(n)$
- Typical search costs:

$d = 14$	DFIDS requires 3,473,941 nodes
	A^* with h_1 requires 539 nodes
	A^* with h_2 requires 113 nodes
$d = 24$	DFIDS requires $\approx$ 54,000,000,000 nodes
	A^* with h_1 requires 39,135 nodes
	A^* with h_2 requires 1,641 nodes

- Given any admissible heuristics h_1, h_2,

$$h(n) = \max(h_1(n), h_2(n))$$

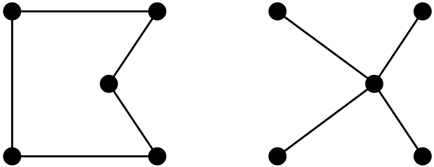
is also admissible and dominates h_1, h_2

Admissible Heuristics from Relaxed Problems

- Derive admissible heuristics from 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: optimal solution cost of a relaxed problem is not greater than the optimal solution cost of the real problem

Relaxed problems cont'd

Well-known example: traveling salesperson problem (TSP)
Find the shortest tour visiting all cities exactly once



Minimum spanning tree (MST) can be computed in $O(n^2)$ (e.g., by the algorithms of Kruskal or Prim) and is a lower bound on the shortest (open) tour

Recall: A ST of a connected, undirected graph G is a subgraph of G which is a tree and connects all the vertices together

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 but space complexity exponential
  - Iterative-deepening A^* (IDA^*) reduces space complexity to polynomial
- Admissible heuristics can be derived from exact solution of relaxed problems