



# Informed Search and Exploration for Agents

R&N: § 3.5, 3.6

Jacques Fleuriot



University of Edinburgh

Informatics 2D



## Outline

- Best-first search
- Greedy best-first search
- A\* search
- Heuristics
- Admissibility

Informatics 2D



## Review: Tree 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

Informatics 2D



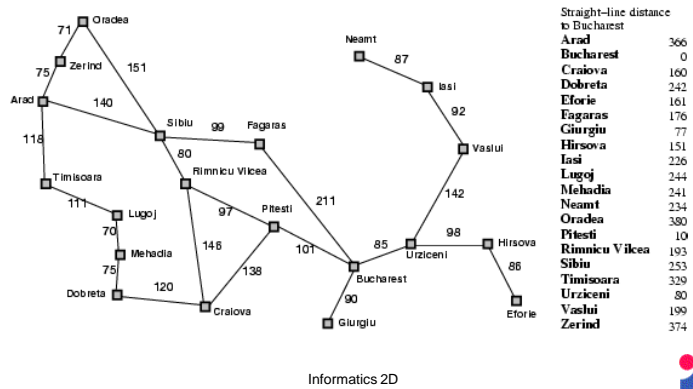
## Best-first search

- An instance of general TREE-SEARCH or GRAPH-SEARCH
- **Idea:** use an evaluation function  $f(n)$  for each node  $n$ 
  - estimate of "desirability"
  - Expand most desirable unexpanded node, usually the node with the **lowest** evaluation
- **Implementation:**
  - Order the nodes in frontier in decreasing order of desirability
- Special cases:
  - Greedy best-first search
  - A\* search

Informatics 2D



## Romania with step costs in km



## Greedy best-first search

- Evaluation function  $f(n) = h(n)$  (heuristic)
- $h(n)$  = **estimated cost** of cheapest path from state at node  $n$  to a *goal* state
  - e.g.,  $h_{SLD}(n)$  = straight-line distance from  $n$  to Bucharest
- Greedy best-first search expands the node that **appears** to be closest to goal

## Greedy best-first search example



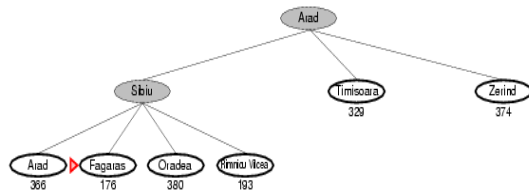
Informatics 2D

## Greedy best-first search example



Informatics 2D

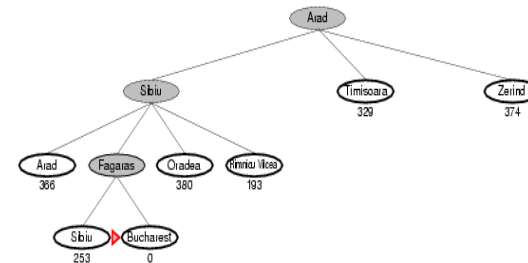
## Greedy best-first search example



Informatics 2D



## Greedy best-first search example



Informatics 2D



## Properties of greedy best-first search

- **Complete?** No – can get stuck in loops
  - Graph search version is complete in finite space, but not in infinite ones
- **Time?**  $O(b^m)$  for tree version, but a good heuristic can give dramatic improvement
- **Space?**  $O(b^m)$  – keeps all nodes in memory
- **Optimal?** No

Informatics 2D



## 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

Informatics 2D



# A\* search example



Informatics 2D



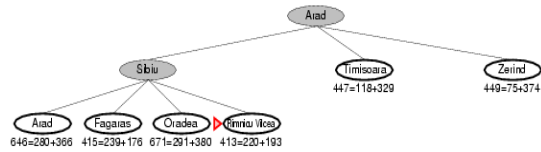
# A\* search example



Informatics 2D



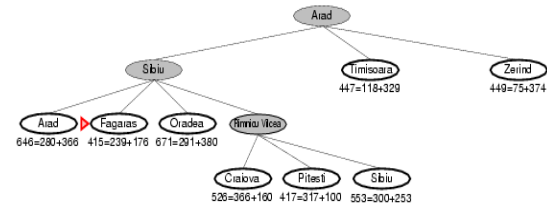
# A\* search example



Informatics 2D



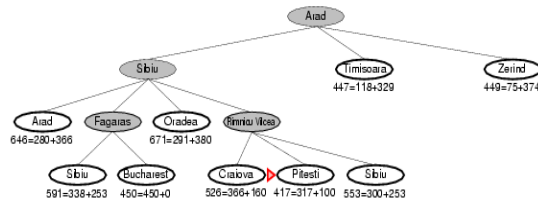
# A\* search example



Informatics 2D



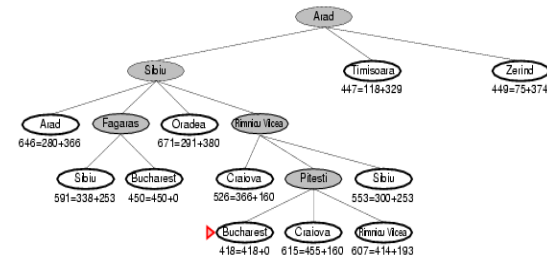
## A\* search example



Informatics 2D



## A\* search example



Informatics 2D



## Admissible heuristics



- A heuristic  $h(n)$  is admissible if for every node  $n$ ,  $h(n) \leq h^*(n)$ , 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
  - Thus,  $f(n) = g(n) + h(n)$  never overestimates the true cost of a solution
- Example:  $h_{SLD}(n)$  (never overestimates the actual road distance)
- **Theorem:** If  $h(n)$  is admissible, A\* using **TREE-SEARCH** is **optimal**

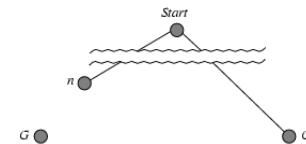
Informatics 2D



## 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  $n$  is on a **shortest path** to an **optimal** goal  $G$ .



- $f(G_2) = g(G_2)$  since  $h(G_2) = 0$
- $g(G_2) > g(G)$  since  $G_2$  is suboptimal
- $f(G) = g(G)$  since  $h(G) = 0$
- $f(G_2) > f(G)$  from above

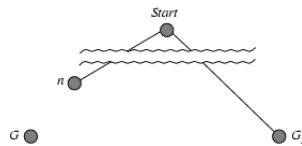
Informatics 2D



## Optimality of A\* (proof *contd.*)



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



- $f(G_2) > f(G)$  from above
- $h(n) \leq h^*(n)$  since  $h$  is admissible
- $g(n) + h(n) \leq g(n) + h^*(n)$
- $f(n) \leq f(G)$

Hence  $f(G_2) > f(n)$ , and A\* will never select  $G_2$  for expansion  
Informatics 2D



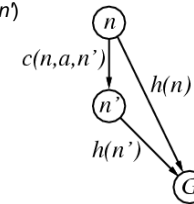
## Consistent heuristics



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

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

- If  $h$  is consistent, we have
 
$$\begin{aligned} f(n) &= g(n) + h(n) \\ &= g(n) + c(n, a, n') + h(n') \\ &\geq g(n) + h(n) \\ &\geq f(n) \end{aligned}$$
- i.e.,  $f(n)$  is non-decreasing along any path.
- Theorem:** If  $h(n)$  is consistent, A\* using **GRAPH-SEARCH** is optimal



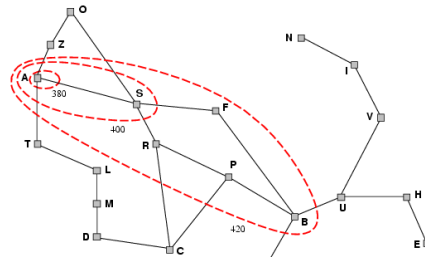
Informatics 2D



## 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  $f_i < f_{i+1}$



Informatics 2D



## Properties of A\*



- Complete?** Yes (unless there are infinitely many nodes with  $f \leq f(G)$ )
- Time?** Exponential
- Space?** Keeps all nodes in memory
- Optimal?** Yes

Informatics 2D



## Admissible heuristics



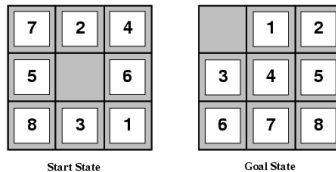
### Example:

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)

Exercise: Calculate these two values.

- $h_1(S) = ?$
- $h_2(S) = ?$



Informatics 2D



## Dominance



- If  $h_2(n) \geq h_1(n)$  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):
  - $d=12$     IDS = 3,644,035 nodes  
 $A^*(h_1)$  = 227 nodes  
 $A^*(h_2)$  = 73 nodes
  - $d=24$     IDS = too many nodes  
 $A^*(h_1)$  = 39,135 nodes  
 $A^*(h_2)$  = 1,641 nodes

Informatics 2D



## Relaxed problems



- A problem with fewer restrictions on the actions is called a **relaxed problem**
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original 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
- Can use relaxation to automatically generate admissible heuristics

Informatics 2D



## Summary



Smart search based on **heuristic scores**.

- Best-first search
- Greedy best-first search
- $A^*$  search
- Admissible heuristics and optimality.

Informatics 2D

