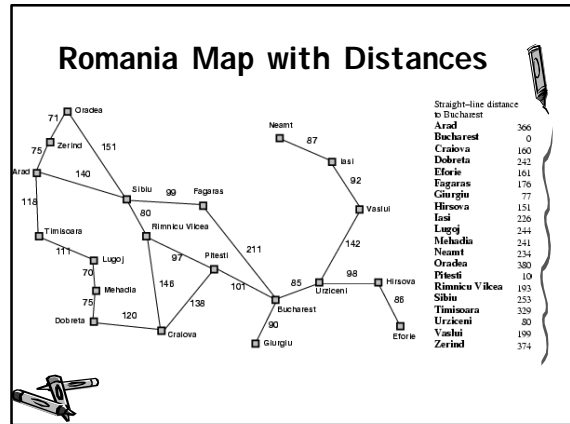


# Informed Search

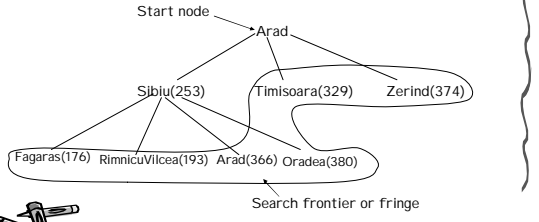
CIS\*3700 (Winter 2007)





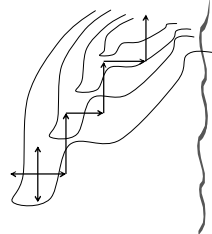
## Hill-Climbing Search

- Maintain fringe as a stack but successors are sorted by  $h(n)$  values before being added to the front



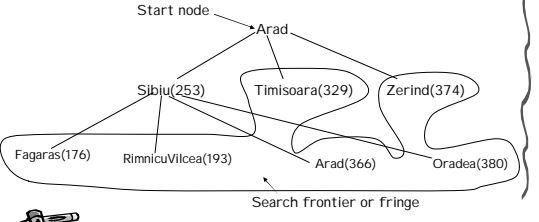
## Hill-Climbing Search

- Problems:
  - Foothills: a local optimum, but not a global optimum
  - Plateaus: a point where heuristic information tells nothing --- all directions are the same
  - Ridges: a point where all known directions are further away from the goal



## Greedy Best-First Search

- Maintain fringe as a sorted list by  $h(n)$  values for all the nodes



## Greedy Best-First Search

- Complete? No - can get stuck in loops, e.g., Iasi → Neamt → Iasi → Neamt → ...
- 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

- Maintain fringe as a sorted list by  $f(n) = g(n) + h(n)$  values for all the nodes
- $g(n)$  = 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\* Search for Map Example

$$\left[ \begin{array}{l} A_{0,366}^{366} \\ S_{140,253}^{393}, T_{118,329}^{447}, Z_{75,374}^{449} \end{array} \right]?$$

$$\left[ \begin{array}{l} S_{140,253}^{393}, T_{118,329}^{447}, Z_{75,374}^{449} \\ A_{280,253}^{533}, F_{239,176}^{415}, O_{291,380}^{671}, R_{220,193}^{413} \end{array} \right]?$$

$$\left[ \begin{array}{l} R_{220,193}^{413}, F_{239,176}^{415}, T_{118,329}^{447}, Z_{75,374}^{449}, A_{280,253}^{533}, O_{291,380}^{671} \\ C_{366,160}^{526}, P_{317,10}^{327}, S_{300,253}^{553} \end{array} \right]?$$

$$\left[ \begin{array}{l} P_{317,10}^{327}, F_{239,176}^{415}, T_{118,329}^{447}, Z_{75,374}^{449}, C_{366,160}^{526}, A_{280,253}^{533}, S_{300,253}^{553}, O_{291,380}^{671} \\ B_{418,0}^{418}, C_{455,160}^{615}, R_{414,193}^{607} \end{array} \right]?$$

$$\left[ \begin{array}{l} F_{239,176}^{415}, B_{418,0}^{418}, T_{118,329}^{447}, Z_{75,374}^{449}, C_{366,160}^{526}, A_{280,253}^{533}, S_{300,253}^{553}, C_{455,160}^{615}, R_{414,193}^{607} \\ O_{291,380}^{671} \end{array} \right]?$$



## A\* Search: Special Cases

- If  $g(n) = 0$ , then  $f(n) = h(n)$ 
  - Greedy Best-First Search
- If  $h(n) = 0$ , then  $f(n) = g(n)$ 
  - Uniform-cost search
- If  $h(n) = 0$  and unit-cost for  $g(n)$ , then  $f(n) = g(n)$ 
  - Breadth-First Search
- If  $h(n)$  is perfect, search will find the goal without waste effort, but what will happen when  $h(n)$  is neither perfect nor zero?



## Admissibility of A\* Search

- Theorem: A\* finds an optimal path (the first solution found) if action costs are bounded above zero, and  $h(n)$  is a lower bound on the actual minimum cost of the cheapest path from  $n$  to a goal node.
- There is no need to check cycles in A\* search.
  - In case of a cycle,  $g(n)$  will keep increasing, since all action costs are bounded above zero.
  - Eventually, the algorithm will pick a node on the solution path, if there exists one.

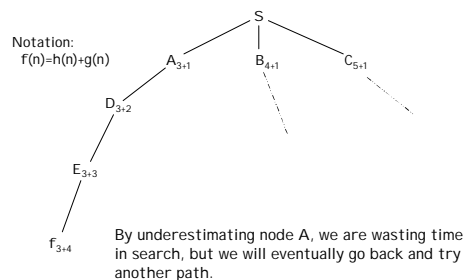


## Admissibility of A\* Search

- A\* search always finds the optimal solution.
  - Suppose  $n_0$  is the first goal node chosen from the fringe. If  $n$  is another node on the fringe and there is a path through  $n$  to another goal node  $n_1$ , then we have:  $g(n_1) \geq g(n) + h(n) = f(n)$ , since  $h(n)$  is a lower bound on the actual cost.
  - Since we assume that  $n_0$  is picked before  $n$ , we must have:  $g(n_0) = f(n_0) \leq f(n)$ .
  - Therefore, we get:  $g(n_0) \leq g(n_1)$ , implying that the first path is the optimal solution.



## Admission Heuristics



### Non-Admission Heuristics

Notation:  
 $f(n) = h(n) + g(n)$

By overestimating node C, we failed to find the optimal solution to G2.

How practical to assure that  $h(n)$  never overestimates?

### Graceful Decay of Admissibility

- Theorem: If  $h(n)$  rarely overestimates the actual minimum cost by more than  $\epsilon$ , then A\* search will rarely find a solution whose cost is more than  $\epsilon$  greater than the cost of the optimal solution.
- Let  $n_1$  be the first goal node found and  $n_0$  be the goal node on the optimal path. We need to show:  $g(n_1) \leq g(n_0) + \epsilon$ .

### Graceful Decay of Admissibility

- Suppose that at the time  $n_1$  is selected,  $m$  from the fringe is a node on the optimal path from  $s$  to  $n_0$ .
- Then, we know:  $f(n_1) \leq f(m)$ , since  $n_1$  was selected before  $m$ . Thus,  $g(n_1) = f(n_1) \leq f(m) = g(m) + h(m)$ , since  $n_1$  is a goal node.
- Let  $h^*(m)$  be the actual cost from  $m$  to  $n_0$ , then by the condition in the theorem:  $h(m) \leq h^*(m) + \epsilon$ .
- By combining the above formulas, we get:  $g(n_1) \leq g(m) + h^*(m) + \epsilon = g(n_0) + \epsilon$ .
- Therefore, the cost of the first solution found is bounded by the optimal cost plus the error  $\epsilon$ .

### A\* Search

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

### Commonly-Used Heuristics

- Minimizing the difference:
 

|   |   |   |
|---|---|---|
| 6 | 2 | 8 |
| 3 | 3 | 5 |
| 4 | 7 | 1 |

 →
 

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 8 | 6 | 4 |
| 7 | 6 | 5 |

 $h(n) = \#(\text{misplaced tiles}) = 7$
- Manhattan distance:
 

|  |  |   |
|--|--|---|
|  |  |   |
|  |  |   |
|  |  | 1 |

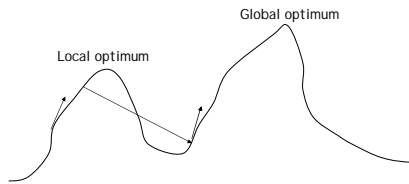
 $h(n) = \sum (|x-x'| + |y-y'|) = 4 + 0 + 2 + 3 + 1 + 3 + 1 + 3 = 17$

### Commonly-Used Heuristics

- Euclidean distance:
 
$$h(n) = \sum \sqrt{(x-x')^2 + (y-y')^2}$$
- How about more than one goal state?
  - Always choose the minimum  $h(n)$

## Simulated Annealing

- Occasionally make a “bad” move in order to find a global optimum.



## Simulated Annealing

- Analogy to metal casting:
  - Large steps are more probable when temperature is high
  - Small steps are more probable when temperature is low
  - Temperature is initially high, but decreases gradually over time
- It can be proved that if  $T$  decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching one.

## Bidirectional Search

- Forward search: from a start node to a goal node
  - Efficient when the forward branching factor is smaller
- Backward search: from a goal node to a start node
  - Efficient when the backward branching factor is smaller
- Bidirectional search: search forwards and backwards alternatively, but how to ensure that the two search frontiers will meet?
  - For DFS, no guarantee, but for BFS, guaranteed
  - When useful? Assuming that the branching factor is  $b$  and the goal is at depth  $k$ , BFS takes  $O(b^k)$  time, while a symmetric bidirectional search takes  $O(2b^{k/2})$  time