

## Announcements

- CS Department Undergrads

A decision was made to restrict departmental scholarship awards to students who have a current departmental scholarship application on file. Furthermore, applications dated earlier than January 1 of this year will be considered out of date. The next Scholarship Committee is scheduled for Feb 5th, so update your application before that date.

- See the department web page

<http://www.cs.tamu.edu> for the awards/schlarships page.

1

## Overview

- Best-first search
- Heuristic function
- Greedy search
- A\*
- Designing good heuristics
- IDA\* basics
- Iterative improvement algorithms
  1. Hill-climbing
  2. Simulated annealing

2

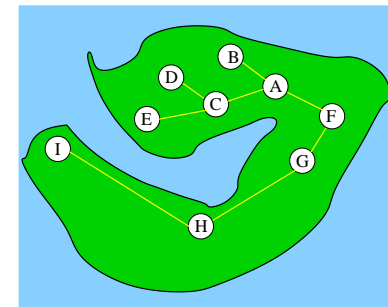
## Best First Search

```
function Best-First-Search (problem, Eval-Fn)
    Queueing-Fn ← sorted list by Eval-Fn(node)
    return General-Search(problem, Queueing-Fn)
```

- The queuing function queues the expanded nodes, and sorts it everytime by the *Eval-Fn* value of each node.
- One of the simplest Eval-Fn: **estimated cost** to reach the goal.

3

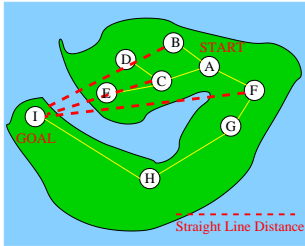
## Heuristic Function



- $h(n)$  = estimated cost of the cheapest path from the state at node  $n$  to a goal state.
- The only requirement is the  $h(n) = 0$  at the goal.
- **Heuristics** means “to find” or “to discover”, or more technically, “how to solve problems” (Polya, 1957).

4

## Heuristics: Example



- $h_{SLD}(n)$ : straight line distance (SLD) is one example.
- Start from **A** and Goal is **I**: **C** is the most promising next step in terms of  $h_{SLD}(n)$ , i.e.  $h(C) < h(B) < h(F)$
- Requires some knowledge:
  1. coordinates of each city
  2. generally, cities toward the goal tend to have smaller **SLD**.

5

## Greedy Search

```
function Greedy-Search (problem)
```

```
     $h(n)$ =estimated cost from  $n$  to goal
```

```
    return Best-First-Search(problem,  $h$ )
```

- Best-first with heuristic function  $h(n)$

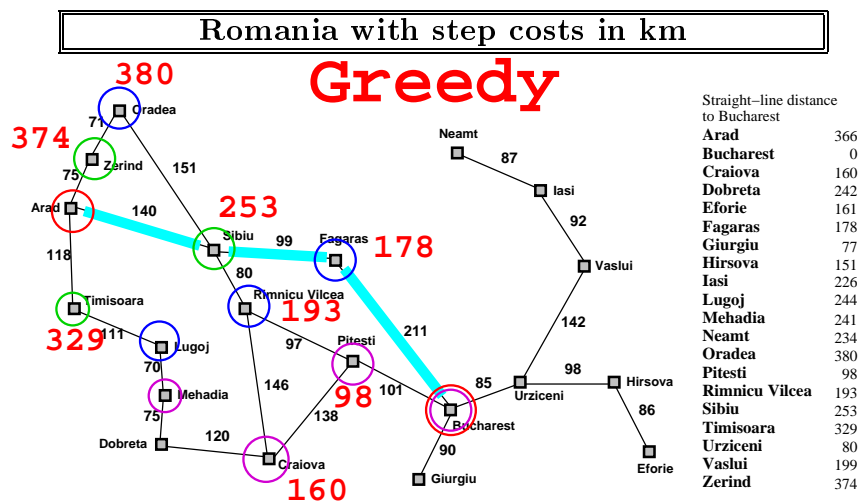
6

## Greedy Search: Evaluation

Branching factor  $b$  and max depth  $m$ :

- Fast, just like Depth-First-Search: single path toward the goal.
- Time:  $b^m$
- Space: same as time – all nodes are stored in sorted list(!), **unlike DFS**
- Incomplete, just like DFS
- Non-optimal, just like DFS

8



Total Path Cost = 450

7

## A\*: Uniform Cost + Heuristic Search

Avoid expanding paths that are already found to be expensive:

- $f(n) = g(n) + h(n)$
- $f(n)$ : estimated cost to goal through node  $n$
- **provably complete and optimal!**
- **restrictions:**  $h(n)$  should be an **admissible heuristic**
- admissible heuristic: one that **never overestimate** the actual cost of the best solution through  $n$

9

## Behavior of A\* Search

- usually, the  $f$  value never decreases along a given path:  
**monotonicity**
- in case it is nonmonotonic, i.e.  $f(Child) < f(Parent)$ , make this adjustment:  
 $f(Child) = \max(f(Parent), g(Child) + h(Child))$ .
- this is called **pathmax**

11

## A\* Search

**function** A\*-Search (*problem*)

$g(n)$ =current cost up till  $n$

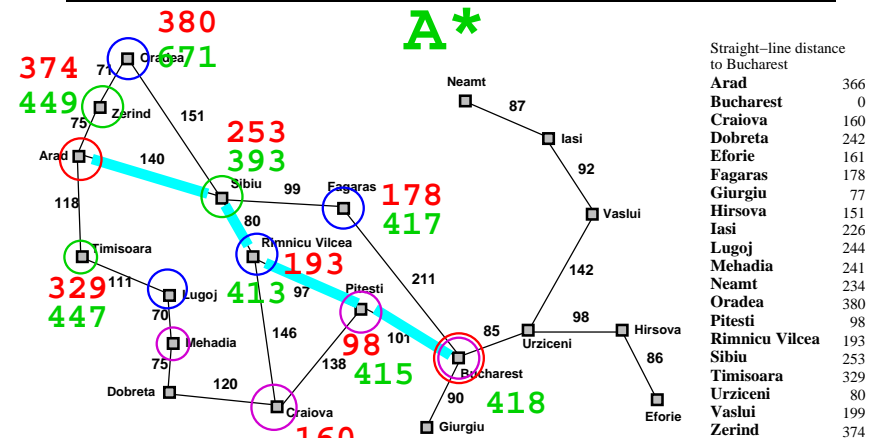
$h(n)$ =estimated cost from  $n$  to goal

**return** Best-First-Search(*problem*,  $g + h$ )

- Condition:  $h(n)$  must be an **admissible heuristic function!**
- A\* is **optimal!**

10

Romania with step costs in km



Total Path Cost = 418

12

## Optimality of $A^*$

$G_2$ : suboptimal goal in the node-list.

$n$ : unexpanded node on a shortest path to goal  $G_1$

- $f(G_2) = g(G_2)$  since  $h(G_2) = 0$
- $> g(G_1)$  since  $G_2$  is suboptimal
- $\geq f(n)$  since  $h$  is admissible

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

13

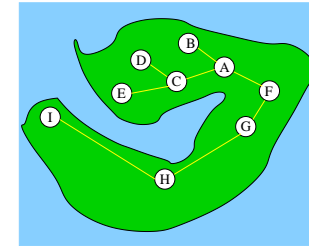
## Lemma to Optimality of $A^*$

Lemma:  $A^*$  expands nodes in order of increasing  $f(n)$  value.

- Gradually adds **f-contours** of nodes (cf. BFS adds layers).
- The goal state may have a  $f$  value: let's call it  $f^*$
- This means that all nodes with  $f < f^*$  will be expanded!

15

## Optimality of $A^*$ : Example



1. **Expansion of parent allowed:** search fails at nodes **B**, **D**, and **E**.
2. **Expansion of parent disallowed:** paths through nodes **B**, **D**, and **E** will have an inflated path cost  $g(n)$ , thus will become nonoptimal.

$$\underbrace{A \rightarrow C \rightarrow E \rightarrow C \rightarrow A}_{\text{inflated path cost}} \rightarrow F \rightarrow \dots$$

14

## $A^*$ : Evaluation

- Complete : unless there are infinitely many nodes with  $f(n) \leq f(G)$
- Time complexity: exponential in (relative error in  $h \times$  length of solution)
- Space complexity: same as time (keep all nodes in memory)
- Optimal

16

## Heuristic Functions: Example

Eight puzzle

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

- $h_1(n)$  = number of misplaced tiles
- $h_2(n)$  = total **Manhattan** distance (city block distance)

$$h_1(n) = 7 \text{ (not counting the blank tile)}$$

$$h_2(n) = 2+3+3+2+4+2+0+2 = 18$$

\* Both are admissible heuristic functions.

17

## Designing Admissible Heuristics

**Relax the problem** to obtain an admissible heuristics.

For example, in 8-puzzle:

- allow tiles to move anywhere  $\rightarrow h_1(n)$
- allow tiles to move to any adjacent location  $\rightarrow h_2(n)$

For traveling:

- allow traveler to travel by air, not just by road: **SLD**

19

## Dominance

If  $h_2(n) \geq h_1(n)$  for all  $n$  and both are admissible, then we say that  $h_2(n)$  **dominates**  $h_1(n)$ , and is better for search.

Typical search costs for depth  $d = 14$ :

- Iterative Deepening : 3,473,941 nodes expanded
- $A^*(h_1)$ : 539 nodes
- $A^*(h_2)$ : 113 nodes

Observe that in  $A^*$ , every node with  $f < f^*$  is expanded. Since  $f = g + h$ , nodes with  $h(n) < f^* - g(n)$  will be expanded, so larger  $h$  will result in less nodes being expanded.

- $f^*$  is the  $f$  value for the optimal solution path.

18

## Other Heuristic Design

- Use composite heuristics:  $h(n) = \max(h_1(n), \dots, h_m(n))$
- Use statistical information: random sample  $h$  and true cost to reach goal. Find out how often  $h$  and true cost is related.

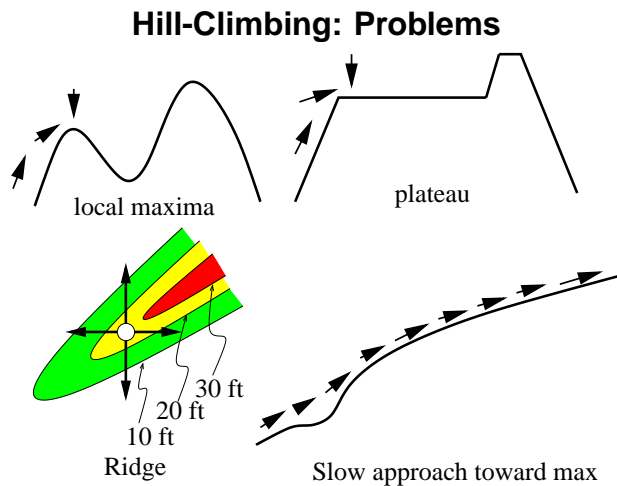
20

## Iterative Deepening $A^*$ : $IDA^*$

$IDA^*$  is complete and optimal, but the performance is limited by the available space.

- Basic idea: only search within a certain  $f$  bound, and gradually increase the  $f$  bound until a solution is found.
- More on  $IDA^*$  next time.

21



- Possible solution: **simulated annealing** – gradually decrease randomness of move to attain globally optimal solution (more on this next week).

23

## Iterative Improvement Algorithms

Start with a complete configuration (all variable values assigned, and **unoptimal**), and **gradually improve** it.

- Hill-climbing (maximize cost function)
- Gradient descent (minimize cost function)
- Simulated Annealing (probabilistic)

22

### Key Points

- best-first-search: definition
- heuristic function  $h(n)$ : what it is
- greedy search: relation to  $h(n)$  and evaluation. How it is different from DFS (time complexity, space complexity)
- $A^*$ : definition, evaluation, conditions of optimality
- designing good heuristics: several rule-of-thumbs
- Basic idea of  $IDA^*$
- Basic idea of iterative improvement algorithms

### Next Time

More on  $IDA^*$  and Simulated Annealing. Chapter 5, Game Playing.

24