

## Week 20: Search III

Martha Lewis  
(based on slides from Raul Santos Rodriguez)

- $A^*$
- Heuristics
- Relaxation

## Best of both: UCS + Greedy

### Idea

$A^*$  takes into account the cost from the root node to the current node and estimates the path cost from the current node to the goal node

$$f(n) = \underline{g(n)} + \underline{h(n)}$$

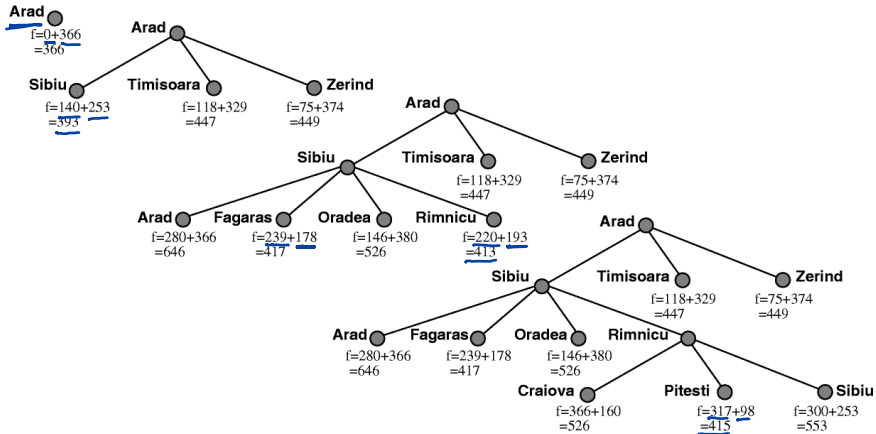
$g(n)$ : path cost from the start node to node  $n$

$h(n)$ : estimated cost of the cheapest path from  $n$  to the goal

$f(n)$ : estimated cost of the cheapest solution through  $n$

$A^*$  distorts costs to favour goal states

# A\*: example



Algorithm: A\* search [Hart/Nilsson/Raphael, 1968]

Run uniform cost search with modified edge costs:

$$Cost'(s, a) = Cost(s, a) + h(Succ(s, a)) - h(s)$$

if  $h(n)$  satisfies **certain conditions**, is A\* search

- complete? **YES**
- optimal? **YES**

The algorithm is identical to UCS, using  $g + h$  instead of  $g$

## $A^*$ : conditions for optimality

Will any heuristic work? **NO**

### Admissibility

A heuristic  $h(n)$  is said to be an admissible heuristic if it never overestimates the cost to reach the goal. For every node  $n$ ,

$$\underline{h(n)} \leq h^*(n),$$

where  $h^*(n)$  is the true cost to reach the goal state from  $n$ .

If  $h(n)$  is **not admissible**, the method is called  $A$ .

Consider the heuristic from our example (straight line distance).

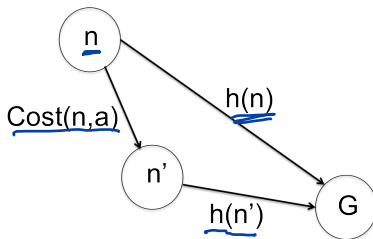
Is the heuristic admissible?

- Yes
- No

## Consistency or monotonicity

A heuristic  $h(n)$  is consistent if, for every node  $n$  and every successor  $n'$  of  $n$  generated by any action  $a$ , the estimated cost of reaching the goal from  $n$  is no greater than the step cost of getting to  $n'$  plus the estimated cost of reaching the goal from  $n'$ :

$$h(n) \leq \text{Cost}(n, a) + h(n')$$

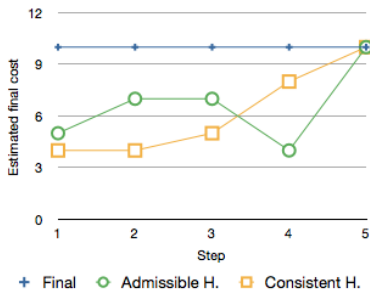




# $A^*$ : conditions for optimality

## Corollary

Every consistent heuristic is also admissible.



Comparison of an admissible but inconsistent and a consistent heuristic evaluation function.

# Optimality of $A^*$

- 1 If  $h(n)$  is consistent, then the values of  $f(n)$  along any path are nondecreasing.

## Proof

Suppose  $n'$  is a successor of  $n$ ; then  $\underline{g(n')} = \underline{g(n)} + \underline{\text{Cost}(n, a)}$  for some action  $a$ , and we have

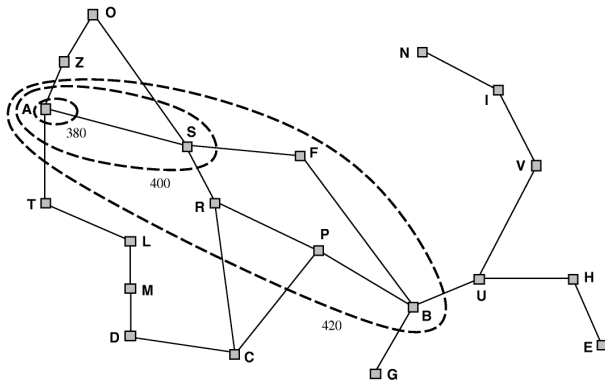
$$\underline{f(n')} = \underline{g(n')} + \underline{h(n')} = \underline{g(n)} + \underline{\text{Cost}(n, a)} + \underline{h(n')} \geq \underline{g(n)} + \underline{h(n)} = f(n)$$

- 2 Whenever  $A^*$  selects a node  $n$  for expansion, the optimal path to that node has been found.

## Proof

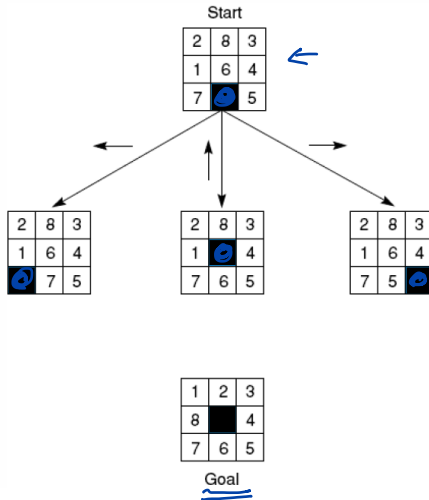
Were this not the case, there would have to be another frontier node  $n'$  on the optimal path from the start node to  $n$ , because  $f$  is nondecreasing along any path,  $n'$  would have lower  $f$ -cost than  $n$  and would have been selected first.

$f$ -costs are nondecreasing along any path  $\rightarrow$  contours in the state space



## Example: 8-puzzle

Slide the tiles horizontally or vertically into the empty space until the configuration matches the goal configuration.



## Example: 8-puzzle

- $h_1(n)$  = number of misplaced tiles
- $h_2(n)$  = total Manhattan distance (i.e., the number of squares from desired location of each tile)
- $h_3(n)$  =  $2 \times$  number of direct tile reversals

---

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

- $h_1 = ?$
- $h_2 = ?$
- $h_3 = ?$

## Example: 8-puzzle

- $h_1(n)$  = number of misplaced tiles
- $h_2(n)$  = total Manhattan distance (i.e., the number of squares from desired location of each tile)
- $h_3(n) = 2 \times$  number of direct tile reversals

---

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

- $h_1 = 8$
- $h_2 = \underline{3} + \underline{1} + \underline{2} + 2 + 2 + 3 + 3 + 2 = 18$
- $h_3 = 0$

## Example: 8-puzzle

<table><tr><td>2</td><td>8</td><td>3</td></tr><tr><td>1</td><td>6</td><td>4</td></tr><tr><td></td><td>7</td><td>5</td></tr></table>	2	8	3	1	6	4		7	5	<b>5</b>	<b>6</b>	<b>0</b>
2	8	3										
1	6	4										
	7	5										
<table><tr><td>2</td><td>8</td><td>3</td></tr><tr><td>1</td><td></td><td>4</td></tr><tr><td>7</td><td>6</td><td>5</td></tr></table>	2	8	3	1		4	7	6	5	<b>3</b>	<b>4</b>	<b>0</b>
2	8	3										
1		4										
7	6	5										
<table><tr><td>2</td><td>8</td><td>3</td></tr><tr><td>1</td><td>6</td><td>4</td></tr><tr><td>7</td><td>5</td><td></td></tr></table>	2	8	3	1	6	4	7	5		<b>5</b>	<b>6</b>	<b>0</b>
2	8	3										
1	6	4										
7	5											
	Tiles out of place	Sum of distances out of place	2 x the number of direct tile reversals									

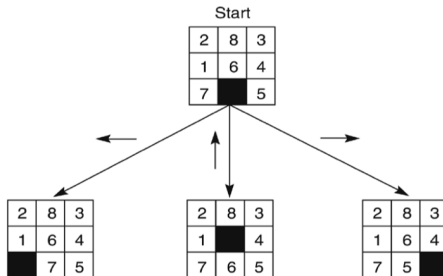
1	2	3
8		4
7	6	5

Goal

# Example: 8-puzzle

$g(n) = 0$

$g(n) = 1$



Values of  $f(n)$  for each state,

6

4

6

where:

$$f(n) = g(n) + h(n),$$

$g(n)$  = actual distance from  $n$   
to the start state, and

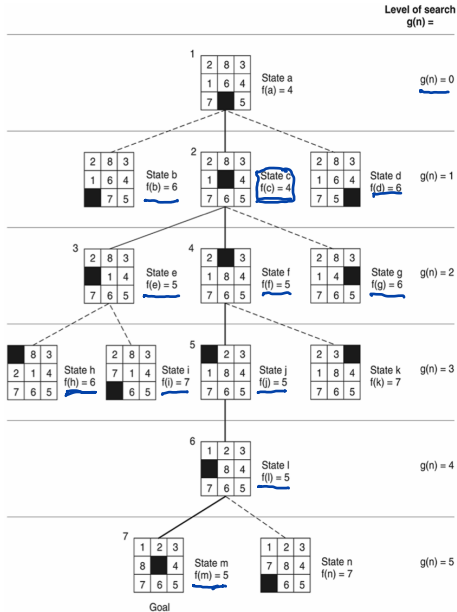
$h(n)$  = number of tiles out of place.

1	2	3
8		4
7	6	5

Goal



# Example: 8-puzzle



## Example: 8-puzzle

	Search Cost (nodes generated)			Effective Branching Factor		
$d$	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	—	539	113	—	1.44	1.23
16	—	1301	211	—	1.45	1.25
18	—	3056	363	—	1.46	1.26
20	—	7276	676	—	1.47	1.27
22	—	18094	1219	—	1.48	1.28
24	—	39135	1641	—	1.48	1.26

**Figure 3.29** Comparison of the search costs and effective branching factors for the ITERATIVE-DEEPENING-SEARCH and  $A^*$  algorithms with  $h_1, h_2$ . Data are averaged over 100 instances of the 8-puzzle for each of various solution lengths  $d$ .

Let  $h_1$  and  $h_2$  be two admissible heuristics. if  $h_2(n) \geq h_1(n)$  for all  $n$ , then  $h_2$  **dominates**  $h_1$ .

## Question

Is it possible for a computer to invent such a heuristic mechanically?

$h_1$  and  $h_2$  are estimates of the remaining path length for the 8-puzzle, but they are also perfectly accurate path lengths for **simplified versions** of the puzzle.

- 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.



**Idea:** Constraints make life hard. Get rid of them.



*"Due to TV network constraints, our 5 year mission has been reduced to 13 weeks, with a possible renewal."*

<table><tr><td>2</td><td>8</td><td>3</td></tr><tr><td>1</td><td>6</td><td>4</td></tr><tr><td>■</td><td>7</td><td>5</td></tr></table>	2	8	3	1	6	4	■	7	5	5	6	0
2	8	3										
1	6	4										
■	7	5										
<table><tr><td>2</td><td>8</td><td>3</td></tr><tr><td>1</td><td>■</td><td>4</td></tr><tr><td>7</td><td>6</td><td>5</td></tr></table>	2	8	3	1	■	4	7	6	5	3	4	0
2	8	3										
1	■	4										
7	6	5										
<table><tr><td>2</td><td>8</td><td>3</td></tr><tr><td>1</td><td>6</td><td>4</td></tr><tr><td>7</td><td>5</td><td>■</td></tr></table>	2	8	3	1	6	4	7	5	■	5	6	0
2	8	3										
1	6	4										
7	5	■										
	Tiles out of place	Sum of distances out of place	2 x the number of direct tile reversals									

1	2	3
8	■	4
7	6	5

Goal

- If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then  $h_1(n)$  gives the exact solution!
- If the rules are relaxed so that a tile can move to any adjacent square, then  $h_2(n)$  gives the exact solution!

If a problem is written down in a formal language, it is possible to construct heuristics automatically. Consider the following rule:

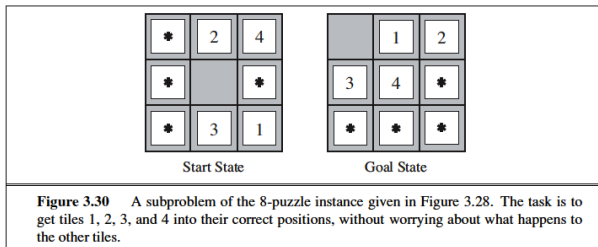
## 8-puzzle

A tile can move from square A to square B if A is horizontally or vertically adjacent to B and B is blank

We can generate three heuristics by removing one or both of the conditions from the above rule:

- a) A tile can move from square A to square B
- b) A tile can move from square A to square B if A is adjacent to B
- c) A tile can move from square A to square B if B is blank

**Subproblems:** relax original problem into independent subproblems



**Learning from experience**

E.g., solving lots of 8-puzzles  $\rightarrow$  training data

Use ML to predict  $h(n) \rightarrow h(n) = c_1x_1(n) + c_2x_2(n)$



## Relaxed search problem

A relaxation  $P'$  of a search problem  $P$  has costs that satisfy:

$$\underline{Cost'(n, a)} \leq \underline{Cost(n, a)}$$

Removing constraints  $\rightarrow$  Reducing edge costs

## Relaxed heuristic

Given a relaxed search problem  $P'$ , define the relaxed heuristic  $h(n) = \underline{h'^*(n)}$ , the minimum cost from  $n$  to a goal state using  $Cost'(n, a)$ .

## Question

If a collection of admissible heuristics  $h_1, \dots, h_m$  is available for a problem and none of them dominates any of the others, which should we choose?

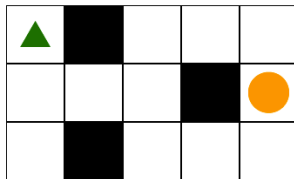
## Question

If a collection of admissible heuristics  $h_1, \dots, h_m$  is available for a problem and none of them dominates any of the others, which should we choose?

$$h(n) = \max\{h_1(n), \dots, h_m(n)\}$$

# Problem

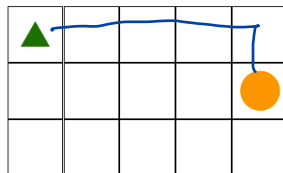
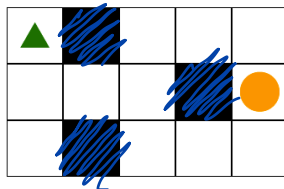
Goal: move from triangle to circle



Find a good heuristic!

# Problem

Goal: move from triangle to circle



$$h(n) = \text{ManhattanDistance}(n, (2, 5))$$

$$\text{e.g., } h((1, 1)) = 5$$

- Informed search:  $A^*$  expands nodes with minimal  $\underline{f(n)} = \underline{g(n)} + \underline{h(n)}$ .
- Consistent and admissible heuristics.
- How to construct heuristics?
  - Relaxation
  - Subproblems
  - Learning