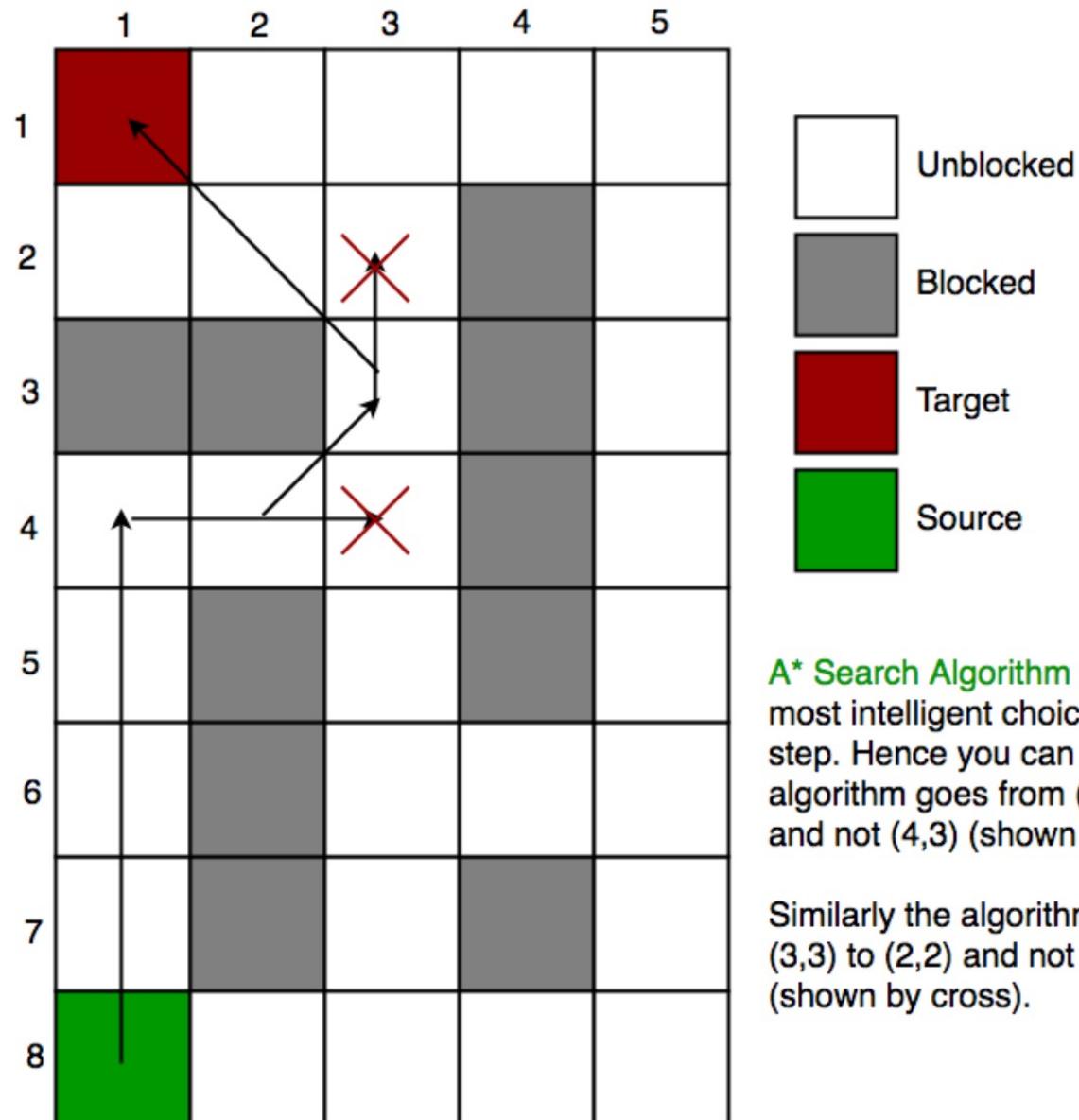


CSCI 3202: Intro to Artificial Intelligence

Lecture 10: A* Search and Heuristics

Rhonda Hoenigman
Department of Computer Science



A* Search Algorithm makes the most intelligent choice at each step. Hence you can see that algorithm goes from (4,2) to (3,3) and not (4,3) (shown by cross).

Similarly the algorithm goes from (3,3) to (2,2) and not (2,3) (shown by cross).

[Source](#)

A* Search

Uniform-cost search:

$$f(n) = g(n) \quad (\text{cost to get to } n)$$

Greedy:

$$f(n) = h(n) \quad (\text{estimated cost to get from } n \text{ to goal})$$

A*:

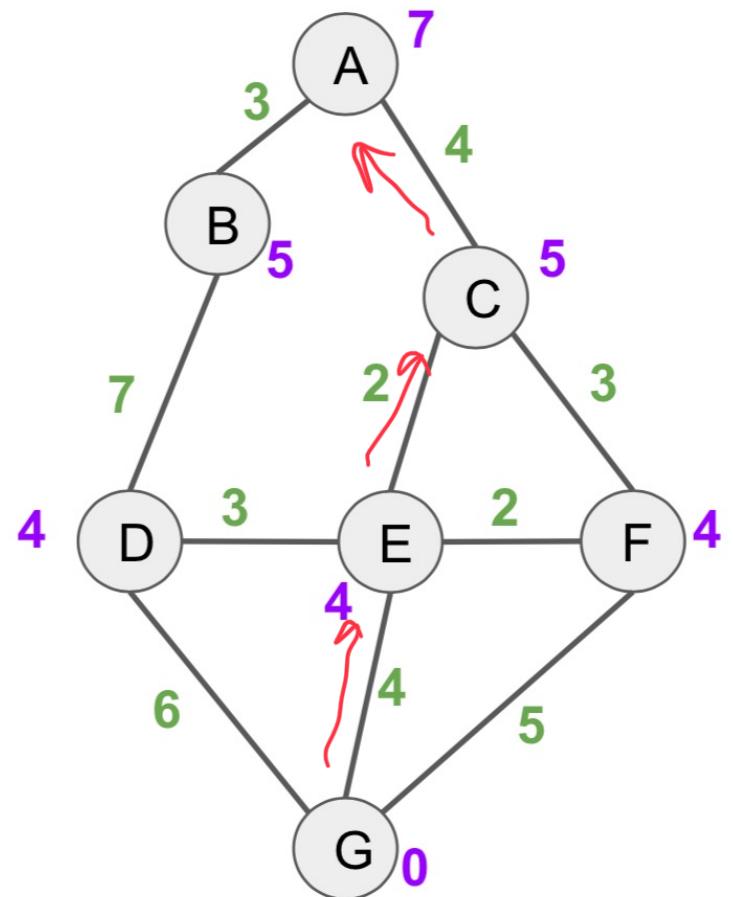
$$f(n) = g(n) + h(n) \quad (\text{estimated total cost of cheapest solution through } n)$$

A* Search

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

A* Search:

- Find the minimum cost path from A to G
- $h(n)$ values are given in purple
- Step costs are given in green



Exp
A
B
C
E
G

A \Rightarrow C \Rightarrow E \Rightarrow G

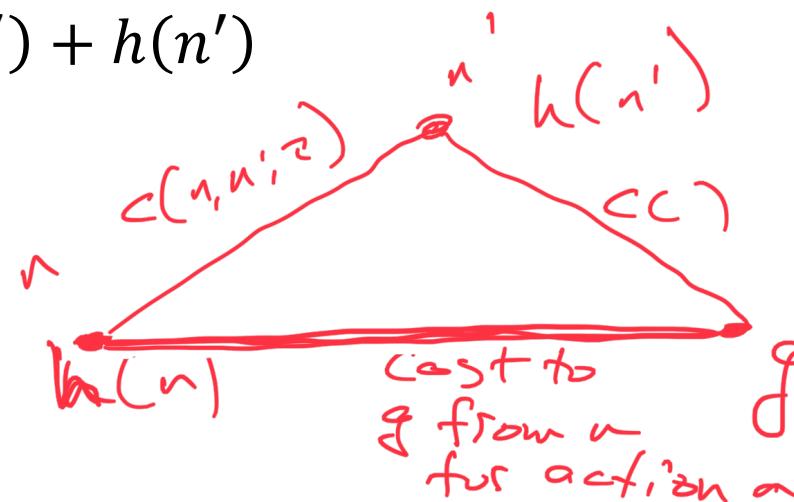
Frontier
(B, 8), (C, 9)
(C, 9), (D, 14)
(E, 10), (F, 11), (D, 14)
(D, 13), (F, 11), (G, 10)

A* is optimal if $h(n)$ is admissible and consistent

Conditions for Optimality: Admissibility & Consistency

- $h(n)$ must be **admissible** - an admissible heuristic is one that never overestimates the cost to reach the goal.
- $h(n)$ 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' :

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



Optimality of A* Search

A* is **optimally efficient** for any given heuristic: No other optimal algorithm is guaranteed to expand fewer nodes than A*

- Recall: A* expands all nodes with $f(n) < C^*$, where C^* is the cost of the optimal solution path.
- Any algorithm that does not expand all nodes with $f(n) < C^*$ risks missing a better solution path.

Optimality of A* Search

So A* is **optimal, complete, and optimally efficient.**

Why do we even care about other search algorithms?

- Number of nodes to expand along the goal contour is still **exponential** in depth of solution/length of solution path.
- Absolute error: $\Delta := h^* - h$
 - h^* = actual cost from root to goal
 - h = heuristic you used
- Relative error: $\epsilon := (h^* - h)/h^*$

Heuristics that is as close to actual cost as possible, we will minimize error.

Want ∞ to be as small as possible

A* Search

Complexity depends strongly on state space characterization

- Single goal, tree, reversible actions $\rightarrow O(b^\Delta)$, or $O(\underline{b}^{\epsilon d})$ with constant step costs (d is solution depth)

Δ typically is proportional to the path cost h^* , so ϵ is pretty much constant (or growing with d), and we can rewrite: $O((b^\epsilon)^d)$

- The effective branching factor is really b^ϵ .
- Important to choose as good of a heuristic as we can.

- Many goal states/near-goal states can be a problem -- need to expand a *lot* of branches.

Back to heuristics ...

- Educated guess about solution quality using domain knowledge
 - *Chess* -
- Using a heuristic can help solve a problem more quickly.
- There is an "art" to deciding on a heuristic function.
- We want $h(n)$ to be admissible. But we need to keep in mind that the lower $h(n)$ is, the more nodes A* expands (making it slower.)
 - *Need to be easy and fast to calculate.*



Heuristics

2	4	8
7	1	
5	6	3

Is it better to
move 1 \rightarrow , 8 \downarrow ,
3 \uparrow



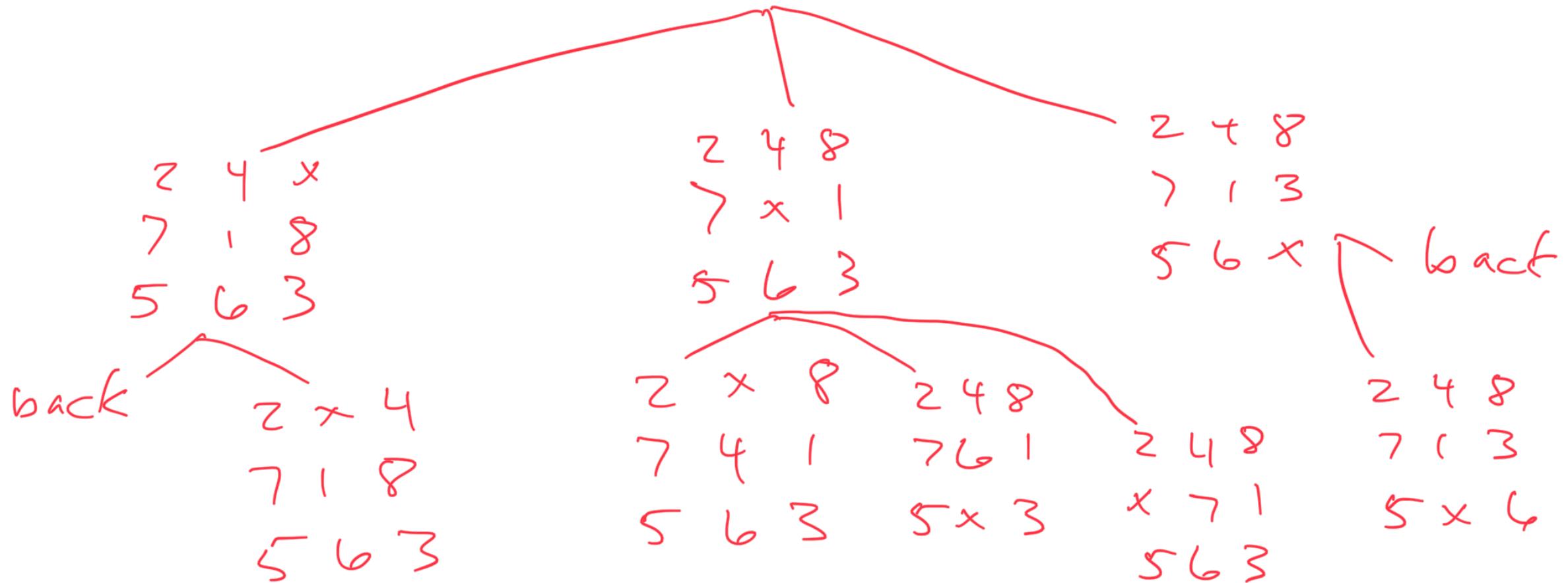
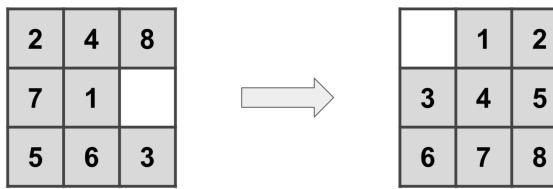
	1	2
3	4	5
6	7	8

Example: solve the 8-tile problem

Solution with fewer
moves is better than soln.
with more moves.

8-tile problem search tree

Is any state closer to goal than other states?



Heuristics

2	4	8
7	1	
5	6	3



	1	2
3	4	5
6	7	8

Branching factor $b \approx 3$

Average solution depth = 22

➤ BFS might expand around $3^{22} \approx 3.1 \times 10^{10}$ nodes (tree)

➤ Graph version: $\frac{9!}{2} \approx 180,000$ distinct reachable states

Iterating
through all
states not
possible

Heuristics

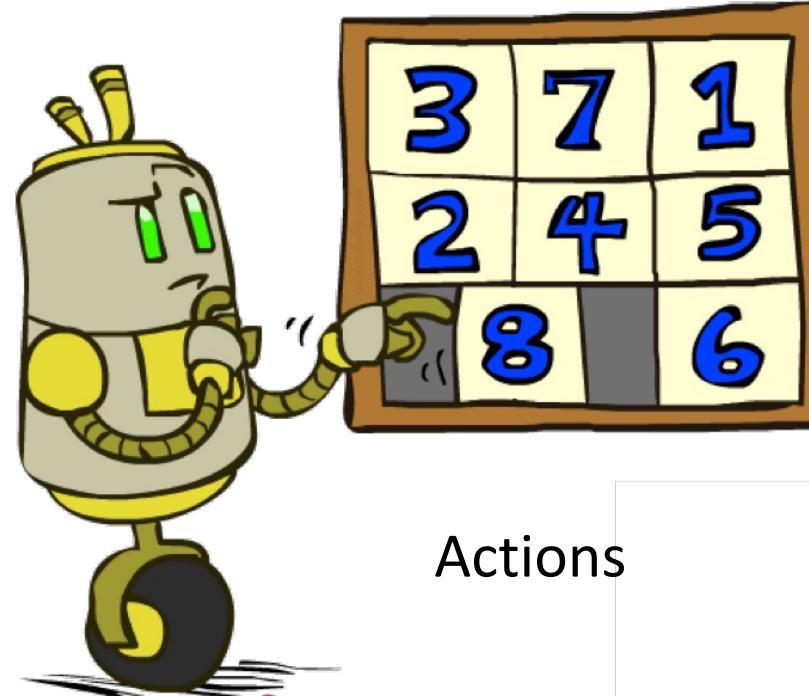
How do we come up with heuristics?

- 1) Generate heuristics from relaxed problems. *Same problem but with fewer constraints*
- 2) Generate heuristics from sub-problems. *Smaller version of same problem.*
- 3) Learning heuristics from experience. *Solve the problem thousands of times and look for patterns.*

Heuristics

7	2	4
5		6
8	3	1

Start State



Actions

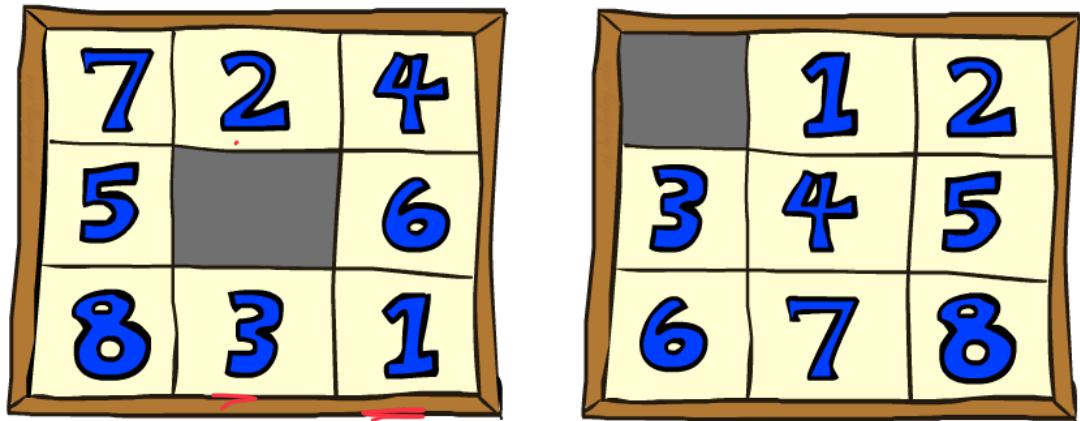
	1	2
3	4	5
6	7	8

Goal State

- What are the states? Positions of tiles
- How many states? $9!$
- What are the actions? slide tile
- How many successors from the start state? 4
- What should the costs be? Cost = 1. Every action needs a cost.
Every uniform cost.

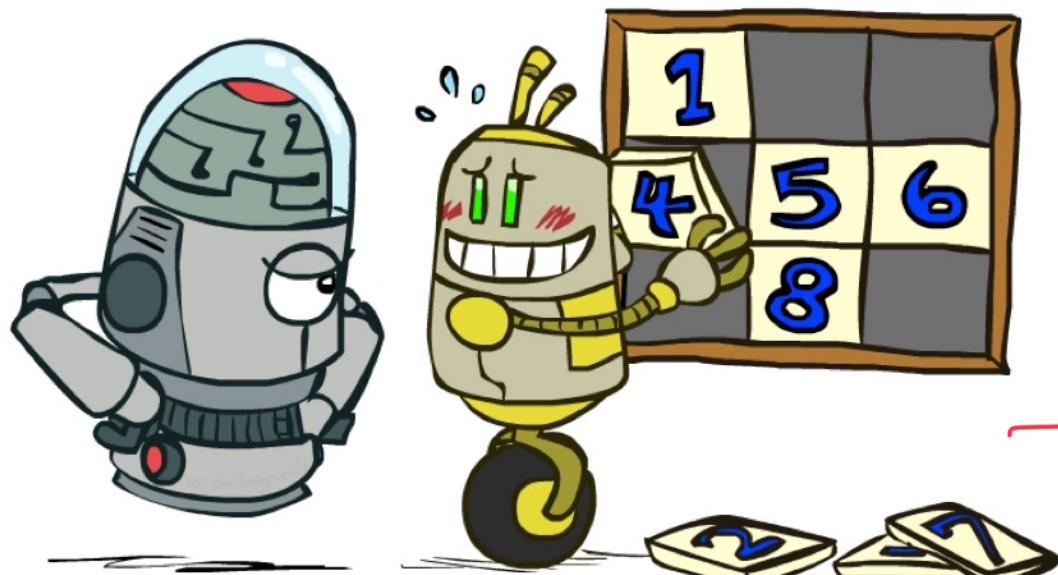
Heuristics – relaxed problem example

- Heuristic: Number of tiles misplaced
- Why is it admissible? *cost will be lower than actual cost.*
- $h(\text{start}) = ?$ 8
- This is a *relaxed-problem* heuristic



Start State

Goal State



Puzzle description

A tile can move from Square A to Square B if B is adjacent to A and B is empty,

3 relaxed problems

A tile can move A to B if A + B are adjacent

A tile can move if B is empty.

Relaxed problems

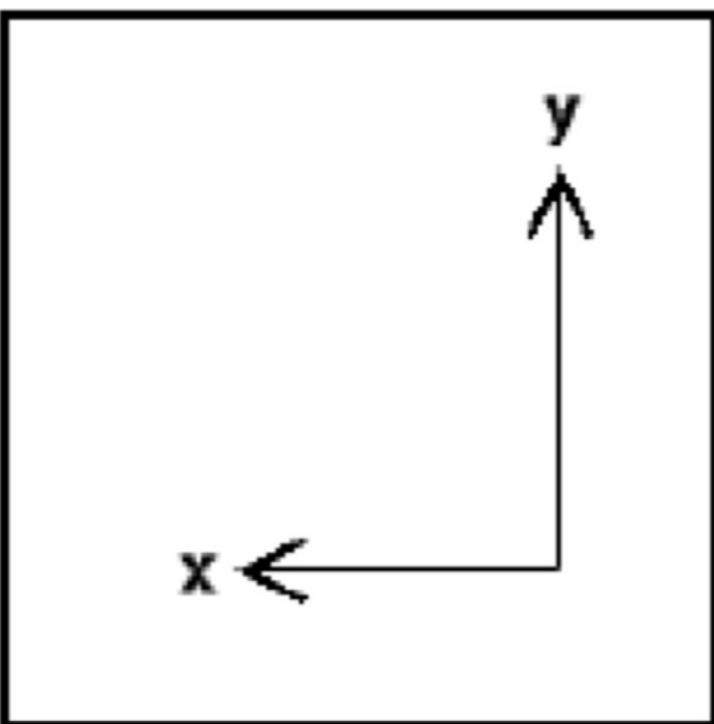
→ A tile can move A to B.

⇒ Any optimal solution to the original problem is also a solution to the relaxed problem.

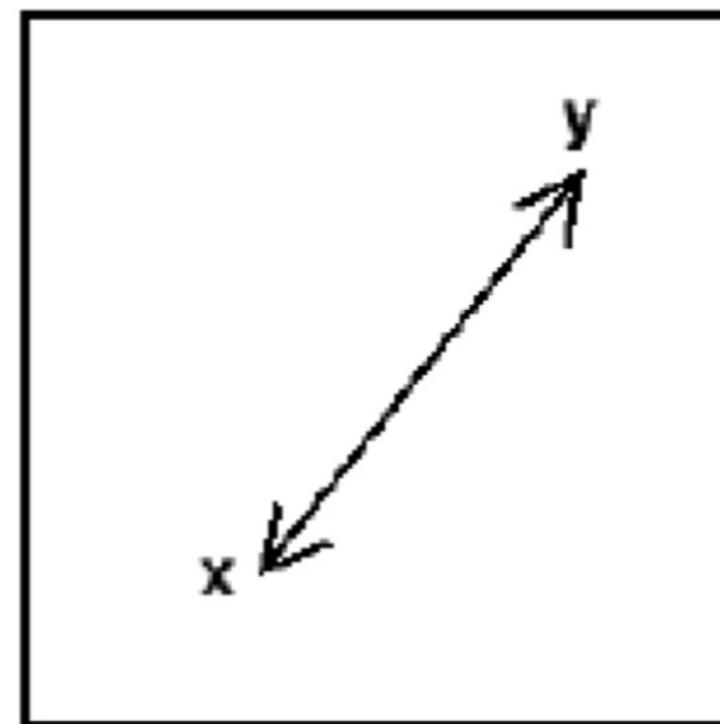
Relaxed problems include “short-cuts” of the original problem – they will be cheaper solutions than the full problem.

⇒ Optimal solutions of the relaxed problem are admissible heuristics

Heuristics – a distance example



Manhattan



Euclidean

Euclidean distance ~~both~~ can be heuristic for
Manhattan distance problems.

Heuristics – relaxed problem example

- What if we had an easier 8-puzzle where any tile could slide any direction at any time, ignoring other tiles?

- Total *Manhattan* distance

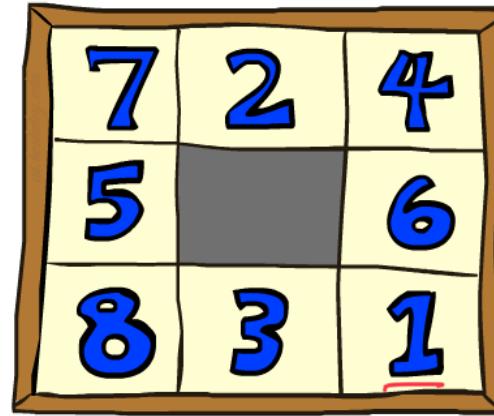
Moves from current state to goal state,

- Why is it admissible?

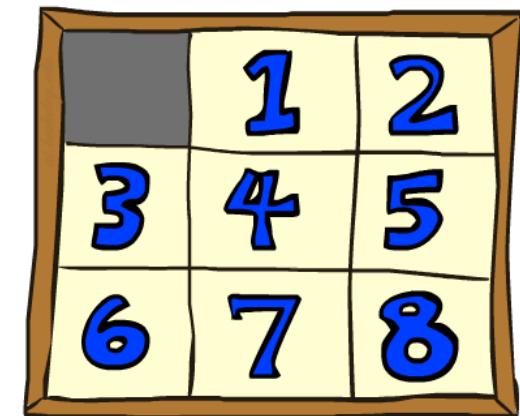
Cost to move
other tiles out of the way
not included.

- $h(\text{start}) =$

$$\begin{array}{r} 1 - 3 \quad 5 - 2 \\ 2 - 1 \quad 6 - 3 \\ 3 - 2 \quad 7 - 3 \\ 4 - 2 \quad 8 - 2 \end{array} \quad 18$$



Start State



Goal State

Average nodes expanded when the optimal path has...

TILES	...4 steps	...8 steps	...12 steps
MANHATTAN	12	25	73
heuristic	13	39	227

where Sars

Nodes expanded

using tiles

cost to move

other tiles out of the way

not included.

Start State

Goal State

Slide source: CS 188 Berkeley

Heuristics

Solutions

- How about using the *actual cost* as a heuristic?

- Would it be admissible? Yes
- Would we save on nodes expanded? No
- What's wrong with it?

We'd
need to solve
the problem first,



- With A*: a trade-off between quality of estimate and work per node
 - As heuristics get closer to the true cost, you will expand fewer nodes but usually do more work per node to compute the heuristic itself

Heuristics – which one is better?

2	4	8
7	1	
5	6	3



	1	2
3	4	5
6	7	8

We want good heuristics!

- h_1 = number of misplaced tiles 8
- h_2 = sum of distances of the tiles from their goal positions (8)

Heuristics

Depending on which heuristic we use, h_1 or h_2 , the search cost (nodes expanded) and b^ϵ will be different.

Performance comparison

d	Search Cost (nodes)			Effective Branching Factor		
	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

- h_2 (Manhattan distance) dominates h_1 (misplaced tiles)

Heuristics

- A* using h_2 will never expand more nodes than A* using h_1

Every node with $f(n) < C^*$ will be expanded

$\Rightarrow f(n) = g(n) + h(n)$, so every node with $h(n) < C^* - g(n)$ will be expanded

But $h_1(n) \leq h_2(n)$, which means any node expanded by A* using h_2 will be expanded by A* using h_1

So it's best to use a heuristic with higher values

- Makes sense, because those are almost necessarily more accurate:

Admissible \rightarrow Can't overestimate \rightarrow The higher they are, the better they are.

8-tile puzzle

$g(a) = \text{moves so far}$

$h(a) = \text{# tiles out of place, Mahattan dist.}$

Heuristics – example

Activity

choose checkers or chess

Generate game rules – ^{Doesn't need to be all} rules.

Are there relaxed problems in the game that could be used in generating heuristics?

Checkers

move one space on diagonal
opponent on diagonal, have to jump

Chess

horse moves in L, jump
King has to move out of check
one piece moves per player per turn
if King ~~is~~ can be attacked

Heuristics

Min # of pieces. # of moves to a capture and can't move out of check, game is over

Heuristics from sub-problems

Arranging the tiles {1, 2, 3, 4} into the proper slots is a subproblem of the general 8-tile problem.

- The cost of an optimal solution to this subproblem is cheaper than the cost of the optimal solution to the full problem.
- Construct a pattern database:
 - Solve each possible configuration of the subproblem
 - Store the cost of the optimal solution
 - Use this as a heuristic
 - Even better: Do this for multiple subproblems and combine the heuristics

Heuristic

Sum ranks of pawns,
ranks of opponents

*Eliminate
of
moves
King can
make.
Assign each
piece a
point value.*

2	4	*
*	1	
*	*	3

Heuristics from sub-problems – example

2	4	*
*	1	
*	*	3

Heuristics from learning

Suppose we solved thousands of 8-tile puzzles

- Then we have a gigantic sample of initial states and of optimal solution paths.

trial₁ *trial₂* *trial₃*

n_1	n_2	n_3	...
-------	-------	-------	-----

misplaced tiles ($x_1(n)$)

	2	8	5	...
--	---	---	---	-----

adjacent tiles that shouldn't be adjacent in goal state ($x_2(n)$)

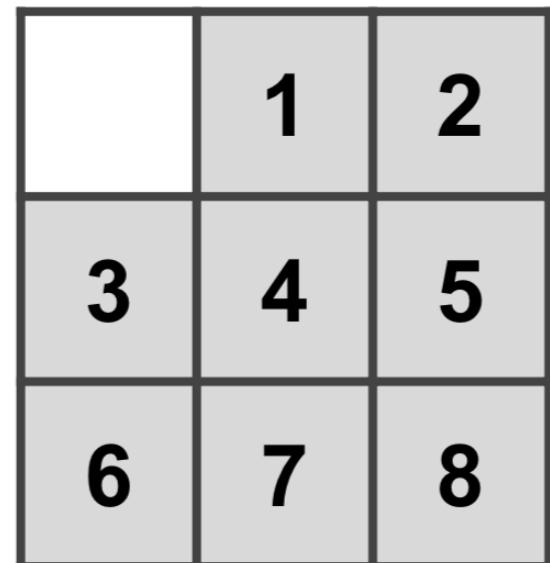
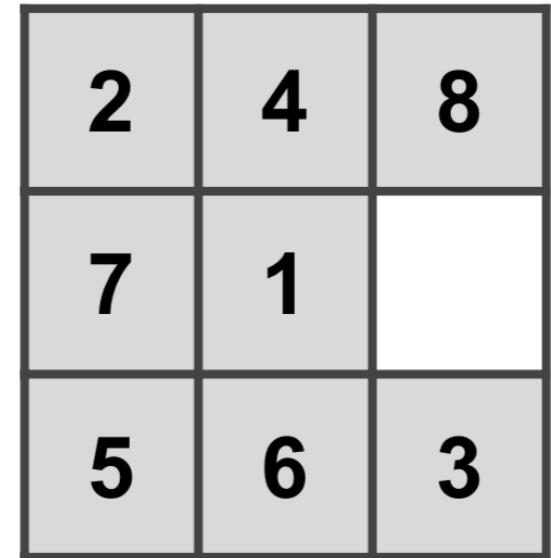
	3	6	4	...
--	---	---	---	-----

Manhattan distance to goal ($x_3(n)$)

	8	14	11	...
--	---	----	----	-----

Cost

	12	24	17	...
--	----	----	----	-----



Heuristics from learning

Predict cost from features of the initial states:

$$h(n) = c_1x_1(n) + c_2x_2(n) + c_3x_3(n) + \dots$$

	n_1	n_2	n_3	...
# misplaced tiles ($x_1(n)$)	2	8	5	...
# adjacent tiles that shouldn't be adjacent in goal state ($x_2(n)$)	3	6	4	...
Manhattan distance to goal ($x_3(n)$)	8	14	11	...
Cost	12	24	17	...

Potential issue:

- Not necessarily admissible/consistent
- Could be, depending on the features and regression constants

