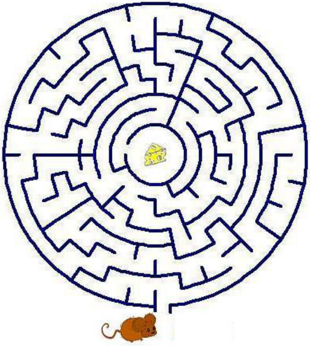
COMP 34120 — Artificial Intelligence and Games

How “heuristics” can speed-up search on a graph

School of Computer Science

Start by considering Puzzles



Not games, no interaction between strategies.

Skip

Motivation

- Find the shortest path from current location to a goal.
- I will show how use of *heuristics* can increase the efficiency of search on a graph.
- The A* algorithm — an informed and efficient way of finding shortest paths on graphs.
- This algorithm is widely used in AI (including game AI) for planning.
- Cannot be used directly for games. Later, heuristics will be used to search game trees and make strong players.

Motivation

- A* algorithm an important algorithm in artificial intelligence.
- Good starting point for games.

What is a heuristic?

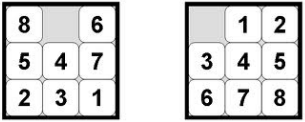
Eureka!

I have found it (translation)

What is a heuristic?

- Rules which are applied because they are *found* to have worked.
- By *trial and error*.

Eight Puzzle



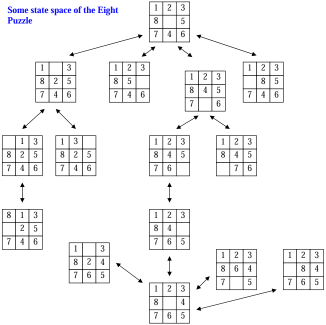
Starting point.   Goal

Some eight-puzzle applets:

- Most have died
- One from Helpful games

State spaces abstracted as weighted graphs

Eight-Puzzle



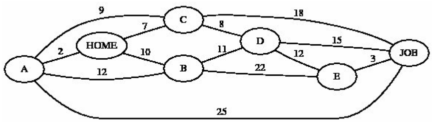
General notation for graphs

Nodes: represent the states of the puzzle.

Edges: An edge from nodes i to j represent a direct move from state i to state j.

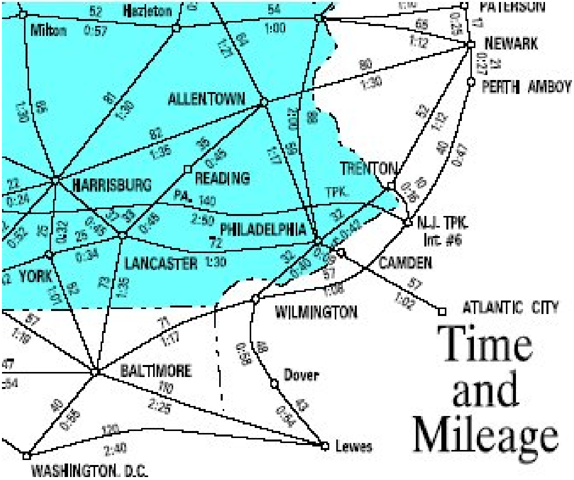
Weights: $c(i, j)$ represents positive distances (time or costs) associated with the move from i to j.
Otherwise, use

$$c(i,j) = \begin{cases} 1; & \text{if there is a link from } i \text{ to } j \\ 0; & \text{otherwise.} \end{cases}$$



Note: Not drawn to scale.

Task: Find the shortest path from Home to Job.



Get from Lewes to Milton in the shortest time (or the shortest distance). continue

Dijkstra's algorithm

- Finds the shortest path from source to all other nodes in graph.
- Searches nodes nearest to source node first (priority queue)
- When a node is popped from the priority queue, the shortest path to it has been found.
- Home2Work

Dijkstra's algorithm (properties)

- Finds the shortest path from a source s to all the nodes in the graph.
- The algorithm is *complete*; if there is a path from the source to the target, this will find it.
- The algorithm finds the *optimal* or shortest path.
- It is *uninformed*; i.e. it does not take into account any information you might have about the location of the goal.

How would a human solve these problems?

- Move to the available state x seemingly "closest to the goal".
- Requires a "heuristic" function:

$$h(x) = \text{ estimated distance from node } x \text{ to the goal.}$$

- Repeatedly move to the available node x' connected to state x with the lowest value of $h(x')$.

Lewes to Milton

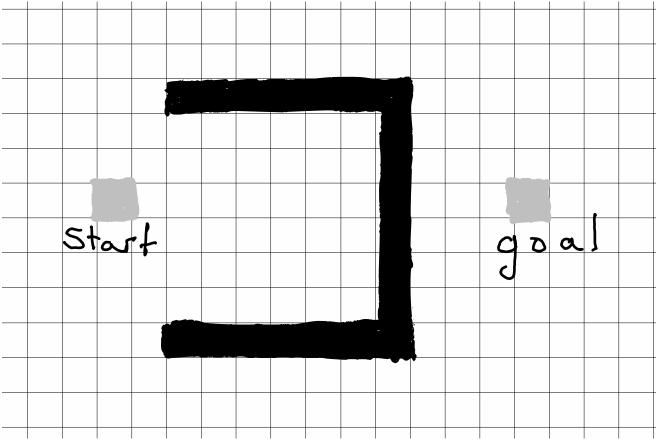
Greedy heuristic search

- Uses a heuristic $h(x)$ approximation for the distance to the goal.
- “Greedily” searches — always move to the unvisited available state nearest to the goal, according to the heuristic.

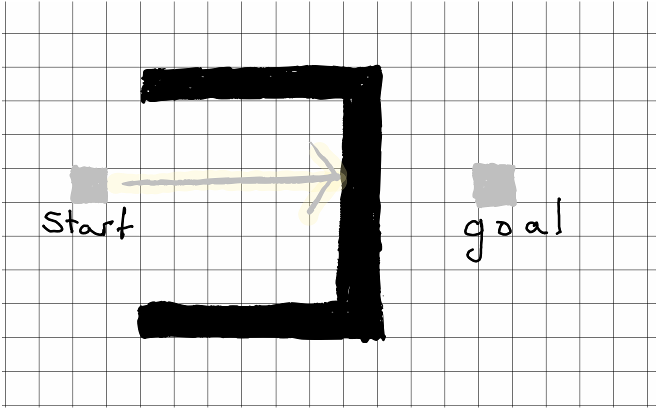
Greedy heuristic search properties

- Not *complete* — will not necessarily find a path even if one exists.
- Does not necessarily find the optimal path.
- Can be very fast.
- It is *informed* search: domain knowledge required to produce and evaluate a good heuristic.
- Can be made complete with backtracking.

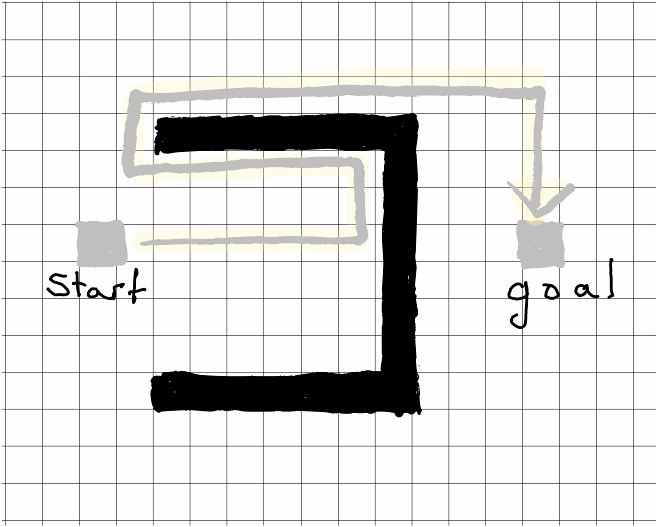
Without backtracking, it can get stuck



Without backtracking, it can get stuck



Does not always find the shortest path



Two Algorithms

Dijkstra's algorithm: Prioritizes nodes closest to the source node.

Greedy heuristic search: Prioritizes nodes closest to the goal, using a heuristic function to estimate the distance to the goal.

A* combines the two algorithms

$g(x)$: The distance from the source s to the node x.

- Dijkstra's algorithm searches ordered on $g(x)$.

$h(x)$: The heuristic. An estimate of the distance of node x to the goal t.

- Greedy heuristic searches ordered on $h(x)$.

A* : searches ordered on their sum,

$$f(x) = g(x) + h(x).$$

- The total estimated distance from source to goal through node x.
- A* is *essentially* Dijkstra with $f(x)$ replacing $g(x)$.

What makes a good heuristic — three properties

Admissible: The heuristic must *underestimate* the true distance. Essential!

Monotonic: Satisfies a triangle inequality (see later slide).

Informative: The closer $h(x)$ is to the true distance to the goal from x, the more informative it is.

What makes a good heuristic — admissible

- A heuristic is called *admissible*, if, for all nodes x, it is no longer than the true shortest distance to the goal t,

$$h(x) \leq d^*(x, t); \text{ for all nodes } x, \tag{1}$$

$$d^*(x, t) = \text{ true shortest distance from node } x \text{ to goal } t. \tag{2}$$

- i.e. *h underestimates* the distance to the goal.
- $h(x)$ should be optimistic.

Theorem: If the heuristic is admissible, when the goal is popped from the priority queue, the shortest path to the goal has been found.

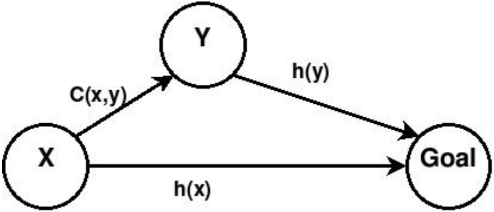
Note: When *non-goal* states are popped, the shortest path to them is not necessarily found (unless the heuristic is *monotonic*, see below). Thus, unlike with Dijkstra's algorithm, it may be necessary to reconsider already expanded nodes.

What makes a good heuristic — monotonic

- A heuristic is called *monotonic* or *consistent* if for all children y of x,

$$h(x) \le c(x, y) + h(y).$$

- A kind of "triangle inequality"



Theorem: If a heuristic is monotonic, then when a node is popped from the priority queue, the shortest path to it has been found.

What makes a good heuristic — informative

- A heuristic should be *informative*, i.e. as close to the true distance without exceeding it.
- A heuristic $\tilde{h}$ is *more informative* than another heuristic $h(x)$ if $h\tilde{(}x) \geq h(x)$ for all nodes x.

Extreme informative and uninformative heuristics

Most informative: $h(x)$ is the true *shortest* distance to the goal t. This is the perfect heuristic. The shortest path will be found with no backtracking.

Least informative: $h(x) = 0$ for all x. (Then A* is the same as Dijkstra's algorithm.)

Eight Puzzle

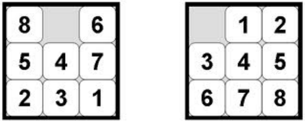
Starting point. Goal

`https://www.helpfulgames.com/subjects/brain-training/sliding-puzzle.html`

Can you think of good heuristics?

- $h_1(x) =$ the number of misplaced tiles.
- $h_2(x) =$ the sum of the (Manhattan) distances of the tiles to their goal positions.
- **Q**uestions: Are the heuristics admissible? Are they monotonic? Which of the two heuristics is more informative?

Eight Puzzle



Starting point. Goal

- $h_1 = 7$. (Only the 4 is in the correct location.)
- $h_2 = 3 + 4 + 2 + 0 + 2 + 4 + 2 + 4 = 24$. (Distance of tile #1, #2, . . . , # 8.)

Eight Puzzle Results

Some typical search costs: (d is the path length, IDS = iterative deepening search, a form of depth-first search).

$d = 14$	IDS = 3,473,941 nodes
	$A^*(h_1)$ = 539 nodes
	$A^*(h_2)$ = 113 nodes
$d = 24$	IDS $\approx$ 54,000,000,000 nodes
	$A^*(h_1)$ = 39,135 nodes
	$A^*(h_2)$ = 1,641 nodes

See "Artificial Intelligence: A Modern Approach (2nd Edition)", S. Russell and P. Norvig (2003) p107 for a more complete table.

[Wikimedia animated gif]

Infinite Mario AI competition

- A contest to control to find the best AI to control Mario.
- Mario AI Competition
- Controllers which reached the highest levels always used A* ; also the slowest.
- Winning (2009) heuristic: “get to the right-hand border of the screen as fast as possible. Avoid being hurt”.

A* Mario (red lines show plan)

What do you need to know about A* search

1. That it exists
2. Where you might use Dijkstra's algorithm, consider using A* search. Could be much faster.
3. Requires a creative step — choice of heuristic.

Can we apply these ideas to games?

No! But, . . .

- The heuristic $h(x)$ is a “state evaluation” function.
- In games, we use *board evaluation* functions.

But the search algorithms must be modified (to take into account the other players).

Future work

Reading: There are some sources in the week 1, Lecture 2 section on Blackboard.

Problem session: Work on problems on A* search from Problem Sheet 1.

Next topic: "Representation of Games". Read Chapter 1 of "Lecture Notes on Game Theory". Pages 6–18 for Lecture 3; pages 19–29 for Lecture 4 Pay particular attention to **Definition** 6 and **Theorem** 1.10.

Example classes

- Start on Monday, 10am—11am.
- Zoom room `https://zoom.us/j/91351820545`