Informed search

-already went through the blind searches

- depth first, breadth first

-now we’re going to the heuristic procedures

-’intelligence’ is about using those limited resources in this way

-heuristic basically means there’s a guideline to be followed

-ex of controlling the center of the chessboard for a chess playing program

Heuristics is used for

-no exact solution exists

-finding the exact solution proves to be expensive

(heuristics aren’t always right)

Hill climbing

-trying to optimize ‘x’

-going to the optimized value and stay there

-maybe reach a peak, but there exists a better answer

-’momentum’

-find a peak, but keep going a little to see what happens

-find a rise, keep going to see if there’s better.

-limited, if you run out, you go back to previous peak

-sometimes that isn’t good though, so you can do the ‘best of three’

-try three start points

-pick the best of those three

-depending on search space, best of three may still not be good.

-goal is to appear intelligent

-three problems with hill climbing

-plateau

-flat in both directions

-solution? best of three

-foothill problem

-local maximums

-ridge problem

-sometimes the maximum is very sharp

-sampling is discrete, maybe the peak is super sharp and it may be overlooked

-solution? decreasing step

Beam search

-used heuristic evaluation function

-like breadth first search but explores only m best nodes throws away nodes not explored

-not optimal, not complete

Best first search

-save all children trees

-expands tree heuristically

-nodes expand in best-first order

-can be like depth or breadth first

-keeps all nodes

-breadth first, but following priority

-looks good? follow it!

-more accurately, estimated best first search

-estimate the ‘cost’ of the next nodes from a given node.

-eventually finds the goal

- queue all the nodes in order of the estimated best first, where best is defined

by a heuristic.

-’greedy search’

-chooses least cost node first

-problem with greedy search

-doesn’t take notice of the distance it travels to get to a low cost node and can end up with sub optimal solutions

-if a longer path is taken initially, ends up with a shorter total path in the end

-Time: O(b^m), good heuristic improved performance

-Space: O(b^m), keeps all nodes in memory

-Optimal? no

Arad to Bucharest example

-shortest initial path leads to a long path around the top, when a shorter total path would come from taking a longer one to begin with

Tile puzzle example

-possible heuristics

-count number of tiles out of place

-sum all the distances the tiles must move

-recognize very hard to swap adjoining tiles

-evaluation function

-can combine multiple heuristics

-force exploration of shallower solutions by adding a path-length cost

Admissibility

-a search algorithm is admissible if it is guaranteed to find a minimal path when it exists

-breadth first search is admissible

-best-first search is not

-a heuristic is admissible when it finds the shortest path

-heuristic used in the tile puzzle problem

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

-takes into account path length and ‘goodness’

-’goodness’ being an estimated distance to goal

-as long as the heuristic does not overestimate distance, it is admissible

-overestimate might cause the best path to be ignored.

A\* algorithm

-best first search with an admissible heuristic

-optimality of A\*

-suppose some suboptimal goal G2 has been generating and is in the queue

Let n be an unexpanded node on a shortest path to an optimal goal G1

f(G2) = g(G2) since h(G2) = 0

> g(G1) since G2 is suboptimal

>= f(n) since h is admissible

since f(G2) > f(n), A\* will never select G2 for expansion