Heuristics

1 Heuristics for TSP

Some heuristics for A* search on the TSP (when formulated so that a state z is a partial tour)

h(z)=

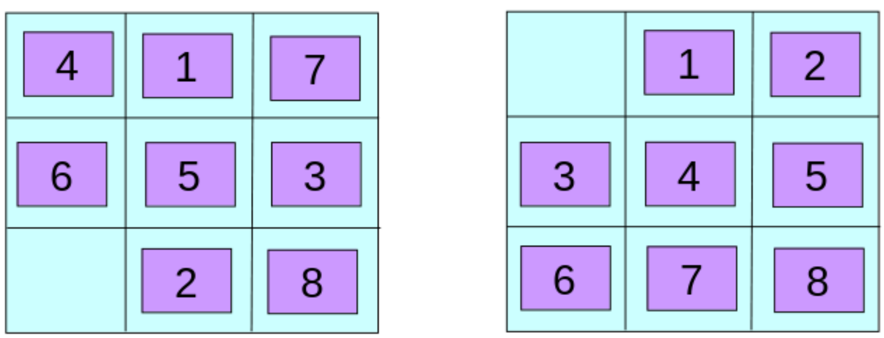
- Distance to the nearest unvisited city (from the current end-city)
- Shortest path of length k through unvisited cities(from the current end-city)
- Distance of the "greedy completion" (from the current end-city)
- Distance of any "partial" completion, only through k unvisited cities (from the current end city and back to the start)
- Distance of the partial tour formed by "inserting" an unvisited city somewhere into the current partial tour
- Distance of the partial tour formed by "inserting" a path of k unvisited cities somwehere into the partial tour
- ...

2 Heuristic functions for the 8-Puzzle Problem

We shall investigate heuristic functions through the 8-Puzzle Problem

- A matrix of 9 cells, 8 of which contain tiles named 1,2,...,8
- Have to move the tiles horizontally and vertically utilising the empty cell
- So that the tiles should end up in a specific configuration

A particular start state and goal state is shown below:



2.1 Common Heuristics

- $h_1(x)$ = the number of misplaced tiles in the corresponding state
  - So if z is a node of the search tree whose corresponding state is configuration shown then $h_1(z)$=6
- $h_2(x)$ = the sum of the distanced of the tiles from their goal positions where the distance is the Manhattan distance
  - i.e., the sum of the horizontal and vertical distanced
  - So if z is a node of the search tree whose corresponding state is the configuration shown then

$$h_2(x) = 0 + 3 + 2 + 2 + 1 + 1 + 3 + 0 = 12$$

Both heuristics are admissible
Experimental evidence suggests that h_2 is better than h_1

2.1.1 Domination

In fact, h_2 is always better than h_1

Notice that for every node z, $h_2(z) \geqslant h_1(z)$

- In general, whenever this is true we say that h_2 dominates h_1

Theorem 5
If:

- h is admissible
- there is a fixed $\epsilon > 0$ such that all step-costs exceed ϵ
- The branching factor is bounded by b

the A* search necessarily expands all nodes z for which $f(z) < c*$, where $c*$ is the optimal path-cost to a goal node

Equivalent to "all nodes z for which $g(z) < c * -h(z)$"

But if $h_2(z) \geqslant h_1(z)$ and $g(z) < c * -h_2(z)$ then $g(z) < c * -h_1(z)$

- so, any node expanded with heuristic h_2 will also be expanded with heuristic h_1
- moreover, A* search with h_1 might expand other nodes too

Thus, it is always best to use a heuristic function with higher values, provided it is admissible and efficiently computable

3 Inventing heuristic functions

How do these heuristics arise?

Whilst h_1 and h_2 are estimates of remaining path length, they are also accurate estimates for simplified versions of the problem

- Suppose that there rules of the problem were changed so that a tile could move to any location and not just to an adjacent vacant cell
- Then h_1 would be the optimal number of steps to a goal node
- Suppose that the rules of the problem were changed so that a file could move one cell up, down, left or right, regardless as to whether the adjacent cell were vacant
- Then h_2 would be the optimal number of steps to a goal node

Definition: Relaxed problem

A problem with fewer restrictions on the actions

Any rule in the original problem should be a rule in the relaxed problem but not necessarily vice versa

The cost of an optimal solution in the relaxed problem is an admissible heuristic for the original problem

- Any solution for the original problem is a solution for the relaxed problem

3.1 Sub-problem heuristics

Heuristics can also be derived from the solution-cost of a sub-problem

Consider the 8-puzzle problem where the cost is defined as just getting tiles 1,2,3 and 4 to their correct positions (without worrying about the other tiles)

The cost of an optimal solution to this sub-problem is used as a heuristic for the main problem

- it is necessarily less than the cost of an optimal solution to the original -problem and so the resulting heuristic is admissible

However, a relaxed problem or a sub-problem cannot be so difficult to solve that the time taken to compute the heuristic values is excessive

4 Automatic heuristic derivation

If a problem is written in a formal language then one can often automatically derive relaxed problems

For example, if the 8-Puzzle Problem actions are defined via

- a tile can move from cell A to cell B if
  - cell A is horizontally or vertically adjacent to cell B and cell B is blank

  then we can generate 3 related problems by removing one or both of the conditions in the conjunction

In order for such heuristics to be practically usable the relaxed problems must be efficiently solvable

5 More than one heuristic

When one generates new heuristic functions, one often fails to obtain a heuristic that is clearly the best from those generated

- i.e., no function dominates any other function

However, one can compose a new heuristic function using all the heuristic functions generated to obtain a dominating heuristic function

Suppose $h_1, h_2, ..., h_m$ are admissible (resp. consistent) heuristic functions

Then the heuristic function h defined via

$$h(x) = \max\{h_1(x), h_2(x), ..., h_m(x)\}$$

is admissible (resp. consistent) and dominates each of $h_1, h_2, ..., h_m$

There is a cost to h

- in order to compute h(x), one needs to compute each of $h_1(x), h_2(x), ..., h_m(x)$