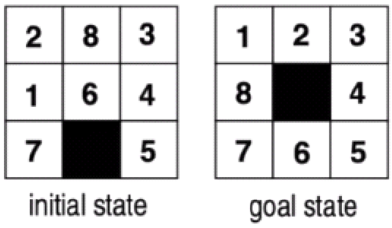
Heuristics

04 November 2020 20:32

A **heuristic function** $h(n)$, takes a node n and returns a non-negative real number that is an estimate of the cost of the least-cost path from node nn to a goal node. The function $h(n)$ is an **admissible heuristic** if $h(n)$ is always less than or equal to the actual cost of a lowest-cost path from node n to a goal.

How do we design our heuristics to make the best search algorithm possible?

Admissible heuristics: Example



- 8-puzzle is just hard enough to be interesting
- Branching factor 3(ish), typical solution around 20 steps
- Exhaustive search: 3^{20} states $(= 3.5 \times 10^7)$
- Eliminating repeating states: $9! = 362880$ states
- Need a decent heuristic.

Possible heuristics

H(n) = number of misplaced tiles
H(n) = Manhattan distance

Characterising Heuristics

- If A* tree-search expands N nodes and solution is depth d, then the effective branching factor * is the branching factor a uniform tree of depth d would have to contain N nodes.
- The closer the effective branching factor is to 1, then the larger the problem that can be solved.
- Can estimate b* experimentally (usually fairly consistent over problem instances)
- Q) is h2 always better than h1? If h(n) is bigger than another h(n) for all n then that heuristic is said to dominate the other heuristic

Deriving Heuristics

Can derive admissible heuristics of the exact solution cost of a relaxed version of the problem
A problem with less restrictions on operators is a relaxed problem
E.g. 8 puzzle, we can relax it and say that a tile can move from A to B if B is blank

If one dominates, use that. Else, calculate the h value for each state and use maximum value. If all options are admissible, then the chosen one will be admissible

Subproblems

Derive admissible heuristics from solution cost of a subproblem of given problem
Cost of subproblem = lower bound on cost of complete problem

We can store exact solution costs in a database which we can use to lookup values

We can combine pattern databases

Disjoint pattern databases

If we can divide up the problem so moves only affect a single subproblem

Statistical approach

Run search over training problems and gather statistics