Learning Heuristics

Want to use solutions we found for a problem to come up with good heuristics for A*. We'll consider an inductive learning approach.

That is, the heuristic we're learning is a linear function

$h(n) = c_1x_1 (n) + c_2x_2(n) + + c_mx_m(n)$ (ci's need to be learned)

Example: a feature of an 8 puzzle might be number of adjacent tiles which are in sequence.

h(start board) should be = cost of optimal solution. for that board.

So we choose ci's such that the difference square of h from the actual value is as small as possible.

In order to calculate ci's in the above equation, take the partial derivative of each ci, set all the equations to 0 in order to maximize distance, and give sample points to solve the system of equations.

Local Search and Optimization

In many search problems the path to the goal doesn't matter.

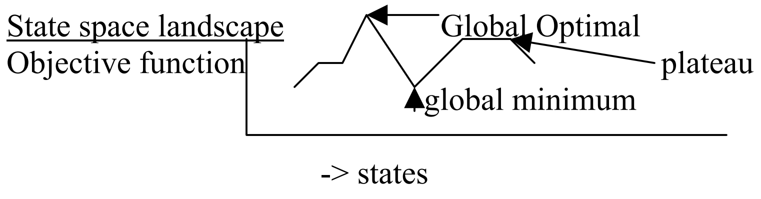
Example – **n queens problem.** (placing n non-attacking queens on a blank chessboard)

By keeping around path info, we're wasting memory.
Instead we should use

Local Search Algorithm. These use a single state and look at neighboring stsates to try to search for a more optimal solution. If a neighbor is more optimal, usually switch to it.

A complete-local search algorithm, finds a globally optimal solution if it exists. (i.e. satisfies goal)

State space landscape



Local Search Techniques

1. Hill climbing move in the direction of increasing objective value until no neighbor has higher value. Not complete. Can get stuck at local optima, plateaux, etc…

Variants:

a Stochastic hill climbing

- choose at random from among uphill moves to do
  (suffers same problem as 1)

b Random Restart Hill climbing

- Do hill climbing several times from randomly selected starting positions.
  Choose best result returned.
  (This works well for n queens)

Simulated Annealing

Idea from metallurgy where annealing means to heat up metal then slowly cool it to a low energy state.

Algorithm.

```
Loop:
        Pick a neighbor of current state at random.
        If neighbor is more optimal
                then switch to it.
        Otherwise
                Switch with probability $\exp(-k\Delta E/\tau)$
                        ($k$ is the constant fudge factor)
                        ($\Delta E$ is how much worse the neighbor is)
                        ($\tau$ is the current temperature)
```