N-Queens and Local Search

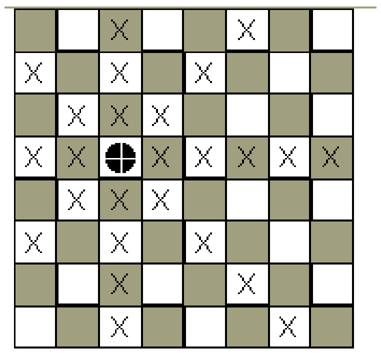
Fall Semester

Topic Outline

- Statement of the Problem
- Hill Climbing
  - Complete state formulation
  - Random restart
  - Plateaus
- Stochastic Search
  - First choice
  - Weighting
  - Simulated annealing
- Beam Search
  - Parallel
- Genetic Algorithms
  - Chromosomes
  - Population
  - Fitness function
  - Breeding
  - Crossover
- Up Next: GHOST

Statement of the Problem

- Consider an $N \times N$ chessboard. Place N queens on the board such that no two queens are attacking each other. The queen is the most powerful piece in chess and can attack from any distance horizontally, vertically, or diagonally. Thus, a solution must place the queens such that no two queens are in the same row, the same column, or along the same diagonal.



Complete State Formulation

- Number the rows and columns from 0 to $N - 1$. Use a one-dimensional list where the index represents the column value. Store the corresponding row value in each column's slot in the list. Thus, the queens are forced to occupy different columns!
- Eight queens along the main diagonal:

```
[0,1,2,3,4,5,6,7]
```

- Four queens solution:

```
[1,3,0,2]
```

- Six queens at random:

```
[4,1,2,5,3,2]
```

Hill Climbing (1)

- Start with a random configuration.
- How close is it to a solution?

$$h\left([4,1,2,5,3,2]\right) = 4$$

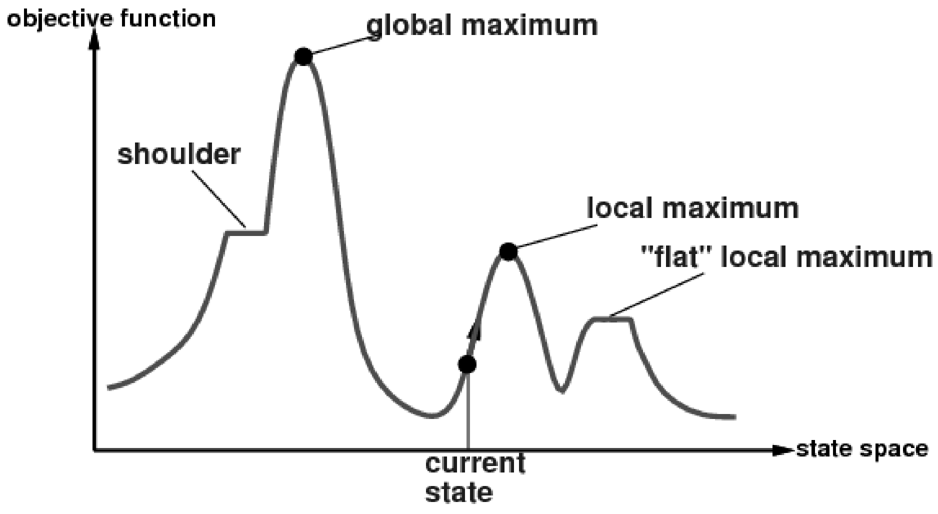
- Calculate $h(board)$ as the number of *pairs* of attacking queens.
- Worst case for $N = 8$ is $h = 28$, if all queens are in the same row or along a main diagonal.
- Note. This is not admissable for A-star, $h\left([2,3,0,2]\right) = 3$ but the board is only one step away from a solution, `board[0]=1`.

Hill Climbing (2)

- Consider all 56 (for $N = 8$) possible next boards formed by moving any one queen to *any* other row in its column.
- Evaluate $h(board)$ for each of these possibilities.
- If none is better than the current h, quit (local min). Else, take the best as the new current board and repeat. This will fail 86% of the time for $N = 8$.
- See also, Russell and Norvig page 111.

Random Restart

- If straight hill climbing fails, just start over with a new random board. (If at first you don't succeed, try, try again.)
- Russell and Norvig: This solves $N = 3 \times 10^6$ in under one minute, and the number of boards is N^N, wow! Russell's slide:



Plateaus

- Allow sideways moves (i.e., next board has *same h* as current board, not better).
- This fails only 6% of the time for $N = 8$. (But for how many sideways moves allowed?)
- Balance the effort in making more sideways moves with the hope that a solution is near versus giving up and doing a random restart.

Also...

- Starting with queens in different rows and *swapping* two columns each time seems to work 50% of the time for $N = 8$.

Stochastic Search: First Choice

- Analogy: punch bowl, dented, ping-pong ball, jiggling.
- Generate neighbors in a random order and take the first board whose h is less than the current h.

Stochastic Search: Weighting

- Of the neighbors whose h is better, weight each board by how much better the new h would be.
- Thus, boards with smaller h would be more likely to get selected but without any guarantee; a neighbor with a better h, but not the best available h, may get selected.

Simulated Annealing

- Possibly allow uphill moves with some small probability, based on “temperature” and decreasing over time (i.e., uphill moves are most likely early in the search).
- See also: Carr, Roger. “Simulated Annealing.” From MathWorld–A Wolfram Web Resource, created by Eric W. Weisstein. http://mathworld.wolfram.com/SimulatedAnnealing.html

Beam Search

- Use k hill climbs or stochastic searches in parallel.
- But, at each step take the next k best boards from *all* the searches, so that some of the searches may die-off by not contributing any boards to the next generation. (Alternative would be for each separate search to take its own next best board.)

Genetic Algorithms (1)

- Initialize a *population* of random boards.
- Rule of thumb, use $O(2N)$ members for solving N-Queens.
- The values in each board state are treated like *chromosomes*.
- Evaluate h, here called the *fitness function*, for each board.
- Typically we want to maximize fitness so use $-h$, if you like.
- Implement *natural selection* by discarding the worst M boards.
- But, we want the population size to be constant over time so...

Genetic Algorithms (2)

- Pair up, for *mating* purposes, the boards that survived. One board could potentially pair up with more than one other board.
- Pairs of boards will *breed* by *crossover* and *mutation*.
- For crossover, pick a point at random and swap all chromosomes (list elements) in the two parent boards before (or after) that point. For mutation, pick an element at random (perhaps maybe) in the child and change it randomly.
- Perform enough mating to return the population size to its initial level, or keep (?) some of the better parent boards from the previous generation, too.

Genetic Algorithms (3)

- This procedure can be applied to any parameterized optimization problem (good) but it is very expensive (bad) as many generations are required for convergence.
- If the landscape of the state space is particularly rough then hill climbing and other steepest descent-type searches are not good and this becomes an even better option.

Genetic Algorithms (4)

- For $N = 8$ this should not take more than ~ 6500 generations because 8^8 boards divided by 25 members in each population divided by 92 possible solutions.
- Try: mutation is a hill climb.
- See also, Russell and Norvig page 111.

Lab Assignment: N-Queens and Local Search

- This bullet intentionally left blank.