Informed Search and Exploration

Chapter 4 (4.3)

Overview

- 4.1-4.2: Heuristic Search
  - Best-First Search Approach
  - Greedy
  - A*
  - Heuristic Functions
- 4.3: Local Search and Optimization
  - Hill-climbing
  - Simulated Annealing
  - Local Beam
  - Genetic Algorithms

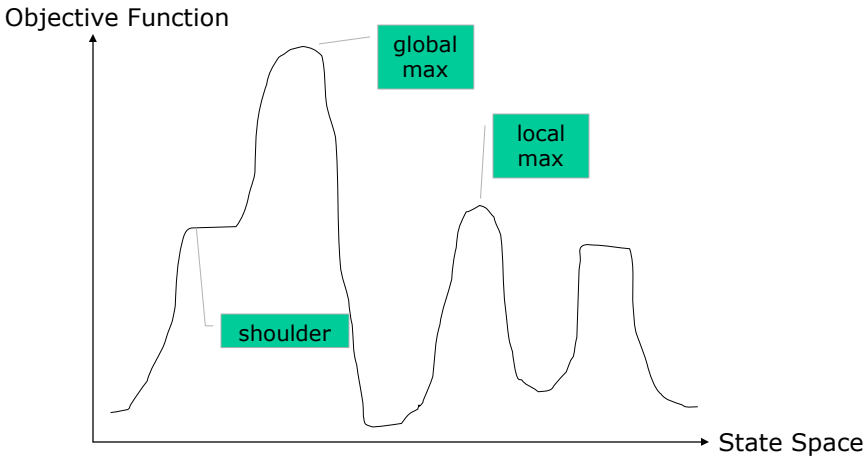
Local Search / Optimization

- Idea is to find the *best* state.
- We don't really care how to get to the best state, just that we get there.
- The best state is defined according to an *objective function*
  - Measures the "fitness" of a state.
- Problem: Find the optimal state
  - The one that maximizes (or minimizes) the objective function.

Properties

- Search the space of "complete" configurations
- Take advantage of local moves
  - Make "local" changes to "complete" configurations
- Keep track of just one state (the current state)
  - No memory of past states
  - No search tree is necessary!

State Space Landscapes



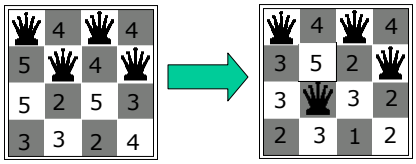
Problem Formulation

- Complete-state formulation
  - Start with an approximate solution and perturb
- n-queens problem
  - Place n queens on a board so that no queen is attacking another queen.
    - The quality of a board configuration = the number of constraints violated (examples in class)
    - Solving = minimize the number of constraint violations

Problem Formulation

- Initial State: n queens placed randomly on the board, one per column.
- Successor function: States that obtained by moving one queen to a new location in its column.
- Heuristic/objective function: The number of pairs of attacking queens.

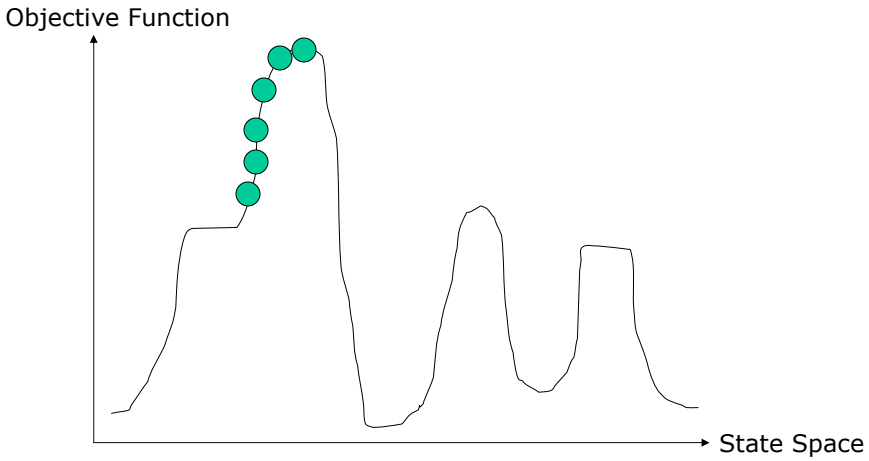
n-Queens



Local Search Algorithms

- Hill climbing
- Simulated annealing
- Local beam search
- Genetic Algorithms

Hill Climbing (or Descent)



Hill Climbing Pseudo-code

•"Like climbing Everest in thick fog with amnesia"

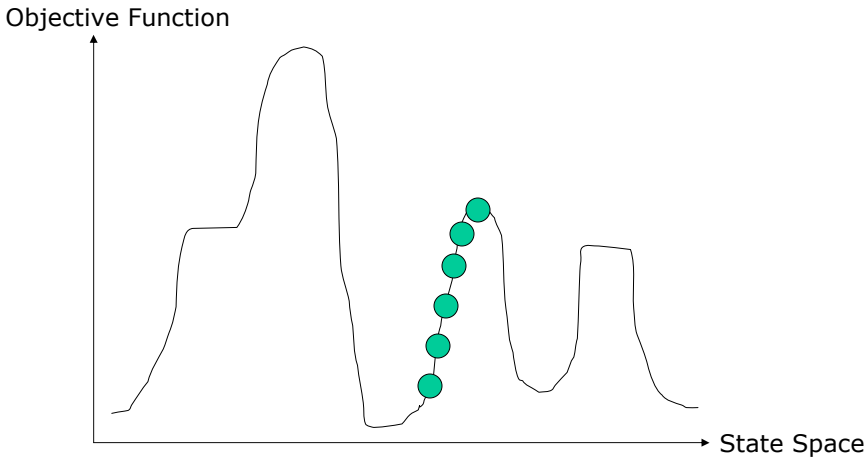
```
function HILL-CLIMBING(problem) returns a state that is a local maximum
   inputs: problem, a problem
   local variables: current, a node
                    neighbor, a node

   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
      neighbor ← a highest-valued successor of current
      if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
      current ← neighbor
```

•Always choose the next best successor state

•Stop when no improvement possible

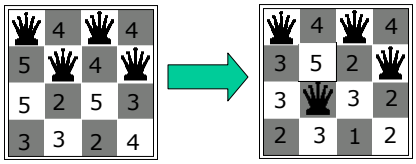
Hill Climbing Problems



Hill Climbing Problems (cont.)

- Local Optimum
- Plateaus
- Shoulders

n-Queens



What happens if we move 3rd queen?

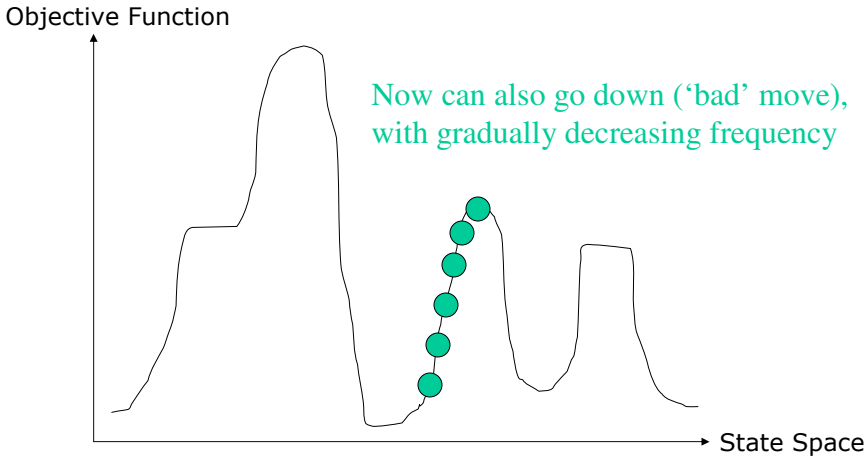
Possible Improvements

- Stochastic hill climbing
  - Choose at random from uphill moves
  - Probability of move could be influenced by steepness
- First-choice hill climbing
  - Generate successors at random until one is better than current.
- Random-restart
  - Execute hill climbing several times, choose best result.
  - If p is probability of a search succeeding, then expected number of restarts is 1/p.

Simulated Annealing

- Similar to stochastic hill climbing
  - Moves are selected at random
  - If a move is an improvement, accept
  - Otherwise, accept with probability less than 1.
- Probability gets smaller as time passes and by the amount of "badness" of the move.

Simulated Annealing



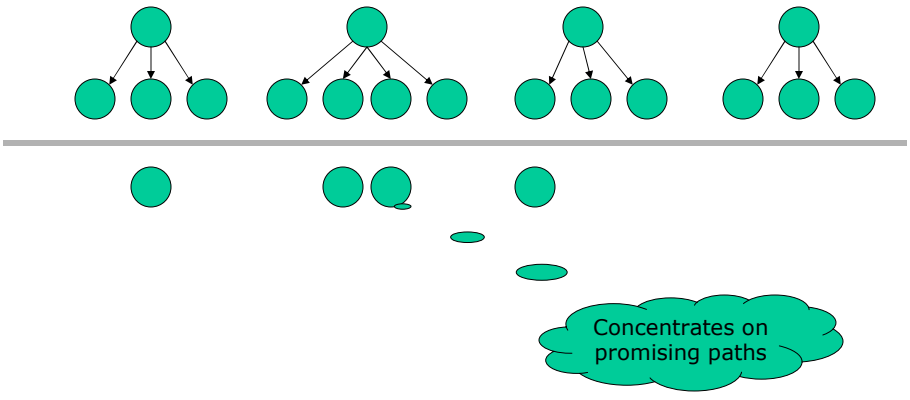
Simulated Annealing

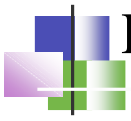
- Developed originally for modeling physical processes
  - temperature parameter T
- If T is decreased slowly enough the best configuration (state) is always reached
- Applications
  - VLSI design, Airline scheduling
- Limitation of pursuing one state configuration

Local Beam Search

- Keep k states in memory instead of just one
- Generate successors of all k states
- If one is a goal, return the goal
- Otherwise, take k best successors and repeat.

Local Beam Search





Local Beam Search

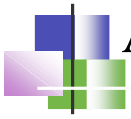
- Initial k states may not be diverse enough
  - Could have clustered around a local max.
- Improvement is *stochastic beam search*
  - Choose k states at random, with probability of choice an increasing function of its value.

Genetic Algorithms

- Can we do better?
  - Assume we have two configurations with good values that are quite different
  - Perhaps the combination of the two individual configurations may lead to a configuration with higher value
  - So grow a population of individual combinations
- Like Beam Search due to multiple states, but less local

Genetic Algorithms

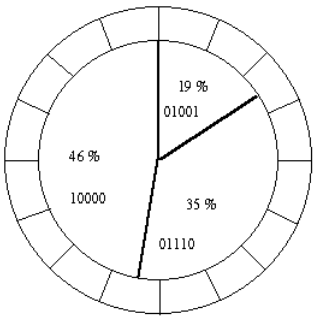
- Variant of stochastic beam search
- Successor states are generated by combining two parent states
  - Hopefully improves diversity
- Start with k states, the *population*
- Each state, or *individual*, represented as a string over a finite alphabet (e.g. DNA)
- Each state is rated by a *fitness function*
  - Measures the quality of a state in the population
- Select parents for reproduction using the fitness function



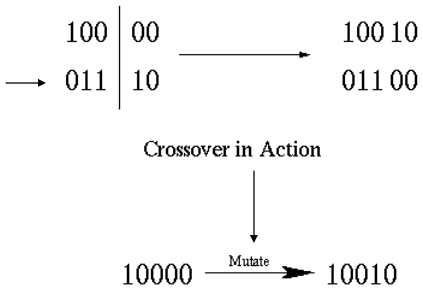
Algorithm Idea

- Create a population of random configurations
- Create a new population through:
  - Biased selection of pairs of configurations from the previous population (via fitness function)
  - Crossover (combination) of pairs
  - Mutation of resulting individuals
- Evolve the population of multiple generation cycles
- Reproduction Process Example – in class

Genetic Algorithms



Roulette Wheel Selection



Taken from http://www.cs.qub.ac.uk/~M.Sullivan/ga/ga_index.html