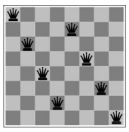
Optimization Problems and Local Search

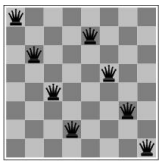
Russell and Norvig 4.3, 4.4

Optimization Problems

- Previously: systematic exploration of search space.
  - Path to goal is the solution
- For some problems path is irrelevant.
  - Example: 8-queens



8 Queens



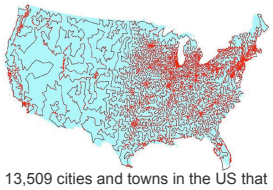
Stated as an optimization problem:

- State space: a board with 8 queens on it
- Objective/cost function: Number of pairs of queens that are attacking each other (quality of the state).

The Traveling Salesman Problem (TSP)

TSP: Given a list of cities and their pairwise distances, find a shortest possible tour that visits each city exactly once.

An optimal TSP tour through Germany's 15 largest cities (one out of 14!/2)



13,509 cities and towns in the US that have more than 500 residents

http://www.tsp.gatech.edu/

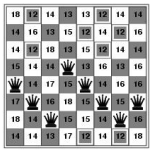
The Traveling Salesman Problem (TSP)

TSP: Given a list of cities and their pairwise distances, find a shortest possible tour that visits each city exactly once.

States?

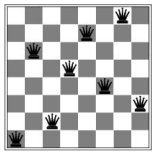
Cost function?

Local Search



- Keep a current state, try to improve it by "locally" exploring the space of solutions
- Improve state by moving a queen to a position where fewer queens attack each other (neighboring state)
- Neighbors: move a queen in its column

Greedy local search



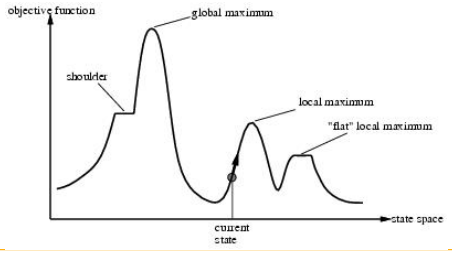
- Problem: can get stuck in a local minimum (happens 86% of the time for the 8-queens problem).

Local minima vs. local maxima

- Local search: find a local maximum or minimum of an objective function (cost function).
- local minima of a function $f(n)$ are the same of the maxima of $-f(n)$. Therefore, if we know how to solve one, we can solve the other.

Hill-climbing

Apply successor function, and keep moves that improve the objective function



Hill-climbing

function HILL-CLIMBING(*problem*) **return** a state that is a local maximum

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

This flavor of hill-climbing is known as steepest ascent (steepest descent when the objective is minimization).

Local search

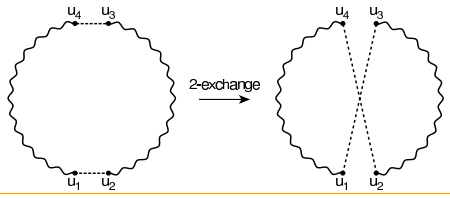
Need to consider:

- Choice of initial state
- Successor function

Solving TSP

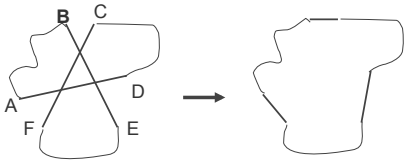
Need to design a successor function that yields valid tours

A 2-opt move:



3-opt

- Choose three edges from tour
- Remove them, and combine the three parts to a tour in the cheapest way to link them



Source: University of Utrecht, www.cs.uu.nl/docs/vakken/na/na2-2005.ppt

Performing a 3-opt move

- Note that the six nodes come in pairs that are already connected: {A,B}, {C,D}, {E,F}
- Node 'A' can connect to:
  - Anything but 'B'
  - Connecting it to D makes it a 2-Opt (OK)
  - 4 choices
- Node 'B' can connect to:
  - Cannot connect to 'A' or what 'A' connects to
  - Cannot connect to the other half of the pair that 'A' connects to.
  - 2 choices
- The final two nodes connect to each other
  - Eight legal possibilities (7 of which are novel)

Solving TSP (cont.)

- 3-opt moves lead to better local minima than 2-opt moves.
- The Lin-Kernighan algorithm (1973): a λ-opt move - constructs a successor that changes λ cities in a tour
- Often finds optimal solutions.
- The best algorithm for TSP until 1989.

The Art of Local Search

- Hill-climbing (like A*) is a simple algorithm
- The art is in defining the successor function
  - More commonly called a neighborhood
  - Goal: avoid local minima

Variations

- Steepest ascent: Successor is the neighbor with the largest increase in objective function.
- Stochastic hill-climbing
  - Random selection among the uphill moves.
  - The selection probability can vary with the steepness of the uphill move.
- First-choice hill-climbing
  - Stochastic hill climbing, generating successors randomly until a better one is found.
- Random-restart hill-climbing
  - Choose best among several hill-climbing runs, each from a different random initial state.

Random Restart

- Suppose that the probability of failure in a single try is P_f
- The probability of failure in k trials:

$$P_f(k \text{ trials}) = (P_f)^k$$

$$P_s(k \text{ trials}) = 1 - P_f(k \text{ trials}) = 1 - (P_f)^k$$

- The probability of success can be made arbitrarily close to 1 by increasing k.
- Example: For the eight queens problem

$$P_s(100 \text{ trials}) = 0.9999997$$

Hill climbing for NP-complete problems

- NP-complete problems can have an exponential number of local minima.
- But, a reasonably good local maximum can often be found after a small number of restarts.