Planning and search

Lecture 3: Local search algorithms

Outline

- ♢ Hill-climbing
- ♢ Tabu search
- ♢ Simulated annealing
- ♢ Local beam search

Optimization problems

Finding the best state according to some objective function

Timetable of classes (objective function looks at clashes, awkward hours, unsuitable rooms . . .)

Route for a rubbish truck (visiting all the bins without driving around too much)

Organism (objective function is reproductive fitness)

Often no clear goal test and path (or its cost) to solution does not matter

Solution space

Assuming the objective function gives a single numerical value, we can plot solutions against this value;

local search explore this 'landscape' (location is the solution and elevation is the objective function value)

assuming the bigger the value of the function the better: we are looking for the global maximum

complete local search: finds a solution if it exists

optimal local search: finds global maximum

Iterative improvement algorithms

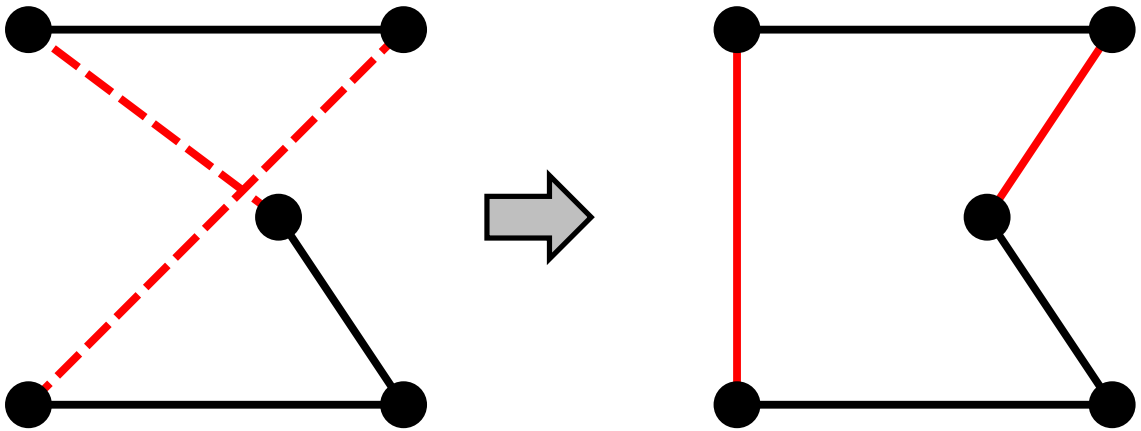
State space = set of "complete" configurations (e.g. for 8-queen problem, with all 8 queens on the board, not a partially constructed solution with fewer pieces);

Iterative improvement algorithms: keep a single "current" state, try to improve it (replace it with a better neighbour)

Not systematic, but use constant amount of memory and often find reasonable solutions in large or infinite state spaces

Example: Travelling Salesperson Problem

Start with any complete tour, perform pairwise exchanges



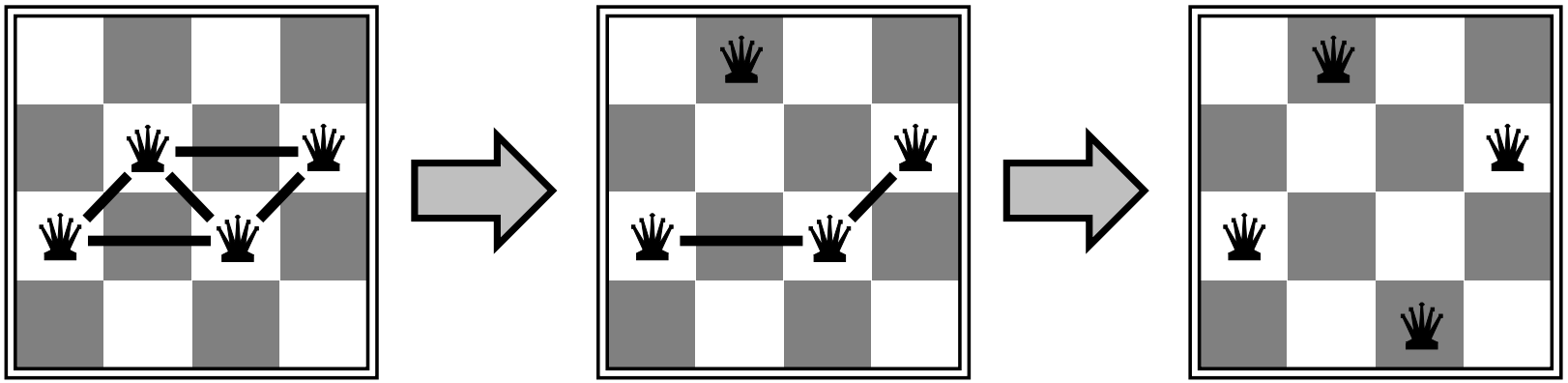
Variants of this approach get within 1% of optimal very quickly with thousands of cities

Example: n-queens

Put n queens on an $n \times n$ board with no two queens on the same row, column, or diagonal

Move a queen to reduce number of conflicts

Heuristic h: number of 'attacks'



h = 5 **h = 2** **h = 0**

Almost always solves n-queens problems almost instantaneously for very large n, e.g., $n = 1\ million$

Hill-climbing (or gradient ascent/descent)

"Like climbing Everest in thick fog with amnesia"

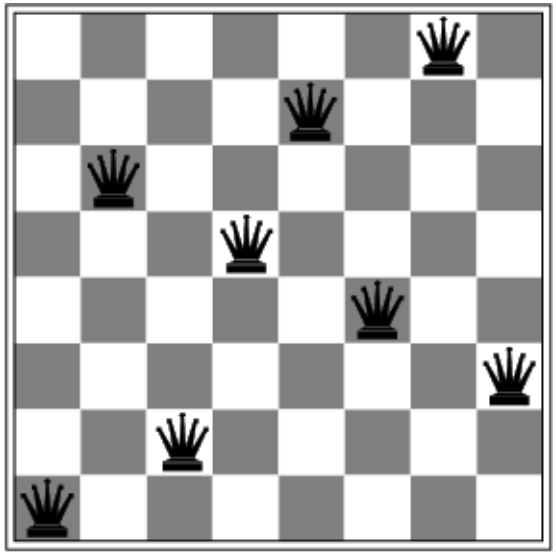
```
function HILL-CLIMBING( problem) returns a state that is a local maximum
   inputs: problem, a problem
   local variables: current, a node
                    neighbor, a node

   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
   end
```

Hill-climbing contd.

Hill climbing is greedy: grabs the next best neighbour without looking ahead

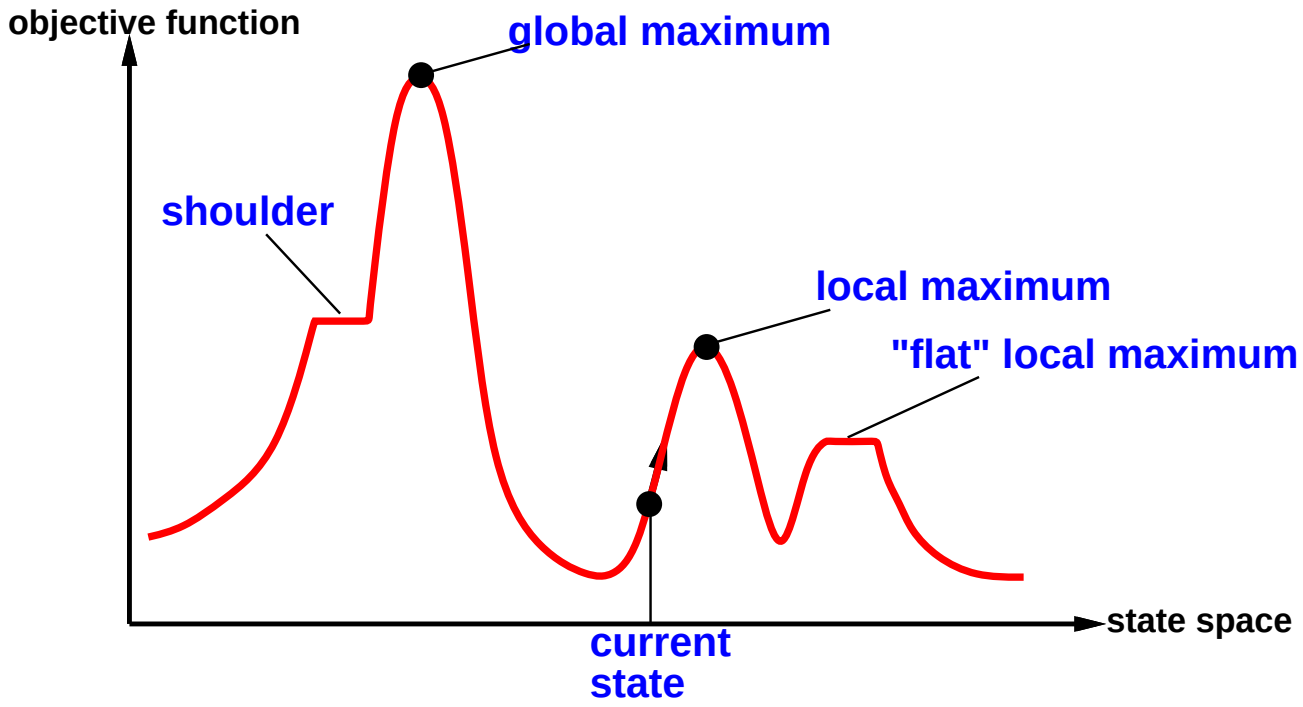
not complete and not optimal; starting from randomly generated 8-queen state, gets stuck 86% of the times

gets stuck in 'local maxima' (below, $h = 1$ - check col. 4 and 7, white diagonal - and every change will create a worse state)



Hill-climbing contd.

Useful to consider state space landscape



Random-restart hill climbing overcomes local maxima—trivially complete

Random sideways moves escape from shoulders loop on flat maxima

Tabu search

Same as hill climbing, but has a 'tabu list' of k previously visited states which cannot be revisited (are 'forbidden', or tabu)

improves efficiency

helps escape local maxima

Simulated annealing

Idea: escape local maxima by allowing some "bad" moves
but gradually decrease their size and frequency

In thermodynamics, the probability to go from a state with energy E_1 to a state of energy E_2 is given by:

$$p = e^{\frac{-(E_1 - E_2)}{kT}},$$

where e is Euler's number (approx 2.72) and k is Boltzmann's constant (relating energy and temperature; with appropriate choice of units it will be equal to 1).

The idea is that probability decreases exponentially with $E_2 - E_1$ increasing, and also the probability gets lower as temperature decreases.

Devised by Metropolis et al., 1953, for physical process modelling

Widely used in VLSI layout, airline scheduling, etc.

```
function SIMULATED-ANNEALING( problem, schedule) returns a solution state
   inputs: problem, a problem
           schedule, a mapping from time to "temperature"
   local variables: current, a node
                    next, a node
                    T, a "temperature" controlling prob. of downward steps

   current ← MAKE-NODE(INITIAL-STATE[problem])
   for t ← 1 to ∞ do
       T ← schedule[t]
       if T = 0 then return current
       next ← a randomly selected successor of current
       ΔE ← VALUE[next] – VALUE[current]
       if ΔE > 0 then current ← next
       else current ← next only with probability e^(ΔE/T)
```

Properties of simulated annealing

At fixed "temperature" T, state occupation probability reaches Boltzman distribution

$$p(x) = \alpha e^{\frac{E(x)}{kT}}$$

T decreased slowly enough $\Longrightarrow$ always reach best state x^*
because $e^{\frac{E(x^*)}{kT}} / e^{\frac{E(x)}{kT}} = e^{\frac{E(x^*)-E(x)}{kT}} \gg 1$ for small T

Is this necessarily an interesting guarantee??

(very, very slow! may be worse than exhaustive enumeration of the search space)

Local beam search

Idea: keep k states instead of 1; generate all their successors and choose top k

Not the same as k searches run in parallel!
Searches that find good states recruit other searches to join them

Problem: quite often, all k states end up on same local hill

Idea: choose k successors randomly, biased towards good ones

Observe the close analogy to natural selection!

Self-test exercise 1

Trace simulated annealing for the 8-queen problem:

VALUE(s)$= -h(s)$

schedule(1)=2, schedule(2)=1 and schedule(n)=0 for $n \geq 3$

start with a solution with $h = 1$

assume that random selection always returns the same solution with $h = 2$

for probabilistic choice, assume that if probability is greater that $1/2$ the solution is chosen, otherwise not.

Self-test exercise 2

Give the name of the algorithm which results from

a. Local beam search with $k = 1$

b. Local beam search with one initial state and no limit on the number of states retained

c. Simulated annealing with $T = \infty$ at all times

Next lecture

Next lecture: Genetic algorithms

Reading: Russell and Norvig, chapter 4