# Artificial Intelligence: Search
# Part 3: Objective optimization

Thomas Trappenberg

January 16, 2009

Based on the slides provided by Russell and Norvig, Chapter 4, Section 3–4

# Outline

- ♢ Hill-climbing
- ♢ Simulated annealing
- ♢ More formally on local search in continuous spaces

# Iterative improvement algorithms

In many optimization problems, **path** is irrelevant;
the goal state itself is the solution

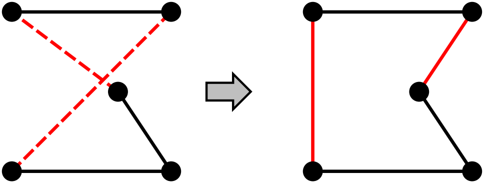
Then state space = set of "complete" configurations;
find **optimal** configuration, e.g., TSP
or, find configuration satisfying constraints, e.g., timetable

In such cases, can use iterative improvement algorithms;
keep a single "current" state, try to improve it

Constant space, suitable for online as well as offline search

# Example: Travelling Salesperson Problem

Start with any complete tour, perform pairwise exchanges

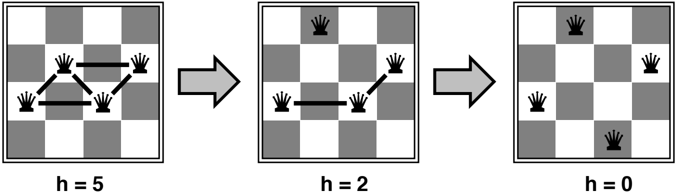


Variants of this approach get within 1% of optimal very quickly with thousands of cities

# 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




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

# Hill-climbing (or gradient ascent/descent)

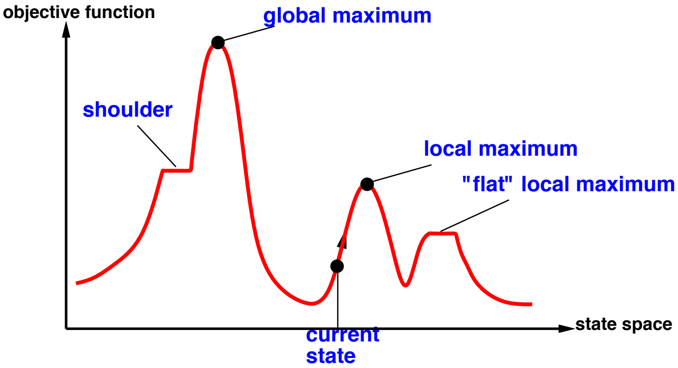
"Like climbing Everest in thick fog with amnesia"

**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] $\leq$ VALUE[current] **then return** STATE[*current*]
    *current* ← *neighbor*
  **end**

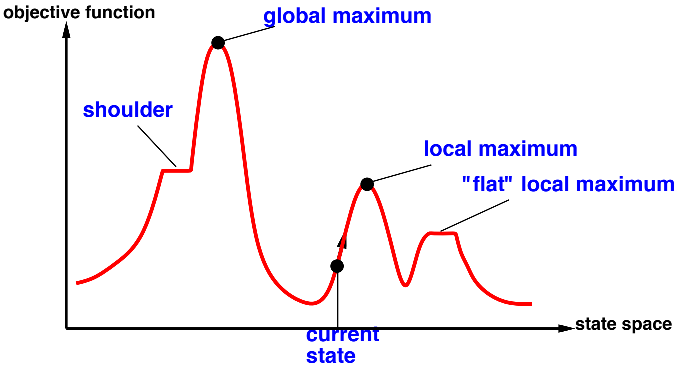
# Hill-climbing contd.

Useful to consider state space landscape



# 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

# Simulated annealing

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

```
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**

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

Widely used in VLSI layout, airline scheduling, etc.

# Continuous state spaces

Suppose we want to site three airports in Romania:
– 6-D state space defined by $(x_1, y_2)$, $(x_2, y_2)$, $(x_3, y_3)$
– objective function $f(x_1, y_2, x_2, y_2, x_3, y_3)$ =
sum of squared distances from each city to nearest airport

Discretization methods turn continuous space into discrete space, e.g., empirical gradient considers $\pm\delta$ change in each coordinate

Gradient methods compute

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3}\right)$$

to increase/reduce $f$, e.g., by $\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x})$

Sometimes can solve for $\nabla f(\mathbf{x}) = 0$ exactly (e.g., with one city).
Newton–Raphson (1664, 1690) iterates $\mathbf{x} \leftarrow \mathbf{x} - \mathbf{H}_f^{-1}(\mathbf{x}) \nabla f(\mathbf{x})$
to solve $\nabla f(\mathbf{x}) = 0$, where $\mathbf{H}_{ij} = \partial^2 f / \partial x_i \partial x_j$