CHAPTER 4, SECTIONS 3–4

LOCAL SEARCH ALGORITHMS
Chapter 4, Sections 3-4

- Local search in continuous spaces (very briefly)
- Genetic algorithms (briefly)
- Simulated annealing
- Hill-climbing

Outline
Iterative improvement algorithms

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

Then state space = set of "complete" configurations;

find optimal configuration, e.g., TSP

find configuration satisfying constraints, e.g., timetable

Chapter 4, Sections 3-4

Constantspace, suitable for online as well as offline search.

In such cases, can use iterative improvement algorithms;

keep a single "current" state, try to improve it.

Or, find configuration satisfying constraints, e.g., timetable;
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

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

Move a queen to reduce number of conflicts row, column, or diagonal.

Put $n$ queens on an $n \times n$ board with no two queens on the same row, column, or diagonal.
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
make-node(initial-state(problem))
current
loop do
    neighbor ← a highest-valued successor of current
    if value(neighbor) ≥ value(current) then return state(current)
    current ← neighbor

end
```

Hill-climbing (or gradient ascent/descent)

Hill-climbing (or gradient ascent/descent)
Hill-climbing contd.

Useful to consider state space landscape

- Objective function
- State space
- Global maximum
- Local maximum
- "Flat" local maximum
- Shoulder

Random restart hill climbing overcomes local maxima—trivially complete.

Random sideways moves escape from shoulders 😊loop on flat maxima

Chapter 4, Sections 3-4
Idea: escape local maxima by allowing some “bad” moves but gradually decrease their size and frequency.

Simulated annealing
Properties of simulated annealing

At fixed "temperature" $T$, state occupation probability reaches Boltzmann distribution $p(x) \propto e^{-E(x)/kT}$, always reach best state $x^*$ because $e^{-E(x)/kT}$ decreases slowly enough 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.
Idea: keep $k$ states instead of $l$; choose top $k$ of all their successors

Local beam search

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

Searches that find good states recruit other searches to join them

Not the same as $k$ searches run in parallel
Genetic algorithms = stochastic local beam search + generate successors from pairs of states
Geneticalgorithms contd.

GAs require states encoded as strings (GPs use programs)

\[ \text{GAs \neq \text{evolution: e.g., real genes encode replication machinery!}} \]

Crossover helps if substrings are meaningful components

\[ \text{Crossover helps if substrings are meaningful components} \]

\[ = \]

\[ + \]

Genetic algorithms contd.
Suppose we want to site three airports in Romania:

**Objective Function**
\[
f(x^1, x^2, x^3) = \sum \text{sum of squared distances from each city to nearest airport}
\]

**Newton–Raphson** (1664, 1690) iterates

\[
0 = (x) f \Delta \Rightarrow \nabla f(x) = H x - x \rightarrow x
\]

Sometimes can solve for \( \nabla f(x) = 0 \) exactly (e.g., with one city).

**Gradient Methods** compute

\[

\text{empirical gradient considers change in each coordinate, e.g.,}

(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}) = \nabla f \Delta
\]

**Discretization** methods turn continuous space into discrete space,

\[
(x) f \Delta v + x \rightarrow x
\]

to increase/reduce \( f \), e.g., by

\[
x \pm \frac{\partial f}{\partial x}
\]

Sometimes can solve for \( \nabla f(x) = 0 \) exactly (e.g., with one city).

\[
0 = (x) f \Delta \Rightarrow \nabla f(x) = H x - x \rightarrow x
\]

**Chapter 4, Sections 3-413**