Lecture 4
Algorithms for unconstrained nonlinear optimisation. Stochastic and metaheuristic algorithms

Analysis and Development of Algorithms

УНИВЕРСИТЕТ ИТМО

Overview

Optimization problem

Problem

$f : \mathbb{R}^n \to \mathbb{R}$; $f = f(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_n)$
To solve the optimization problem $f(\mathbf{x}) \to \min_{\mathbf{x} \in Q}$ means to find $\mathbf{x}^* \in Q$, where Q is the region of acceptability, such that f reaches a minimal value at $\mathbf{x}^*$.
Notation: $\mathbf{x}^* = \arg\min_{\mathbf{x} \in Q} f(\mathbf{x})$.

Methods considered so far:

- direct
- first- and second-order

(*Recall the definitions and the methods*)

Everything was determined precisely $\to$ deterministic approaches

What if we exploit randomness or stochasticity?

Monte Carlo (stochastic) and meta-heuristic methods

Monte Carlo (stochastic) methods are a wide class of algorithms that rely on repeated *random sampling* to obtain solutions to an optimization problem. The methods are most useful when it is impossible or difficult to apply other approaches.

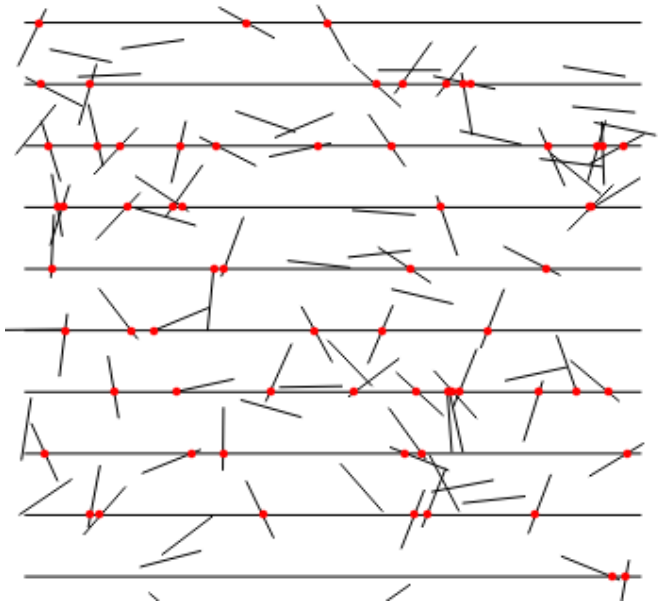
Probabilistic interpretation: by the law of large numbers, the expected value of some random variable can be approximated by taking the empirical mean of independent samples of the variable.

Demonstration: Integration

Metaheuristic methods are *nature-inspired* algorithms that discover a solution to an optimization problem *by trial and error*. Metaheuristics do not guarantee that a globally optimal solution can be found. Many metaheuristics implement some form of stochastic optimization.

Buffon's needle problem

Given a needle of length ℓ dropped on a plane ruled with parallel lines t units apart, what is the probability P that the needle will lie across a line upon landing? Suppose that $\ell \leqslant t$ ("short needle" case).



Solution:

$$P = \frac{2\ell}{\pi t}.$$

Drop n needles and find that m of those are crossing lines, so P is approximated by the fraction m/n. From this, find an experimental estimate for π:

$$\pi \approx \frac{n}{m} \cdot \frac{2\ell}{t}.$$

Demonstration: Real experiment, Simulation

Simulated annealing (Metropolis algorithm)

Simulated annealing is a metaheuristic algorithm that solves the optimization problem similar to the process of annealing in metallurgy.

Name and inspiration: heating and controlled cooling of a material to increase the size of its crystals and reduce their defects.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be *energy*. Let $T = \{T_k\}$ be a decreasing non-negative sequence s.t. $T_k = 0$ for $k > N$ (T is called *cooling schedule* or *temperature*).

Algorithm

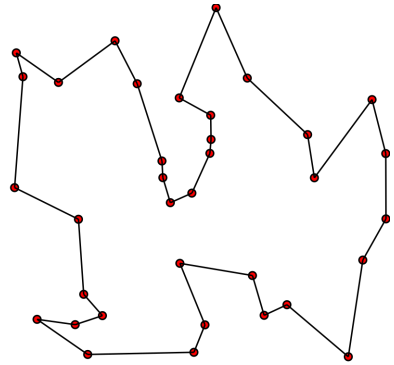
Let a_0 be an initial approximation. At each iteration $k \in \mathbb{N}_0$:

- choose $a^* \in \mathit{Neighbours}(a_k)$, where *Neighbours* is a certain rule;
- if $f(a^*) \leqslant f(a_k)$, then $a_{k+1} = a^*$; if $f(a^*) > f(a_k)$, then $a_{k+1} = a^*$ with probability
$$\exp\left(-\frac{f(a^*) - f(a_k)}{T_k}\right),$$
- Stop if $T_k \equiv 0$.

Demonstrations: 2D, 3D

SA for Travelling Salesman Problem (TSP)

TSP formulation:
Given a list of cities and their locations (usually specified as points on a plane), what is the shortest route which will visit every city exactly once and return to the point of origin?



Question: How can we define the *Neighbours* rule in this case?

Demonstration: Travelling Salesman problem

Active Learning

Friendly description

Research paper

Evolutionary algorithms

Evolutionary algorithms are generic population-based metaheuristic optimization algorithms. They use mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the function under optimization determines the quality of the solutions. Evolution of the population then takes place after the repeated application of the above-mentioned mechanisms.

Examples:

- Differential evolution
- Particle swarm optimization
- Genetic algorithm (*self-study*)

Demonstration: Animal evolution

Differential evolution

Differential evolution is an metaheuristic algorithm that solves the optimization problem by maintaining a population of *agents*, i.e. candidate solutions, creating new agents by combining existing ones and further keeping the best one.

Choose $p \in [0, 1]$, the *crossover probability*, $w \in [0, 2]$, the *differential weight*, and $N \geq 4$, the *population size*. Let $\mathbf{x} \in \mathbb{R}^n$ denote an agent in the population.

Algorithm

Until a termination criterion is met (e.g. the number of iterations performed):

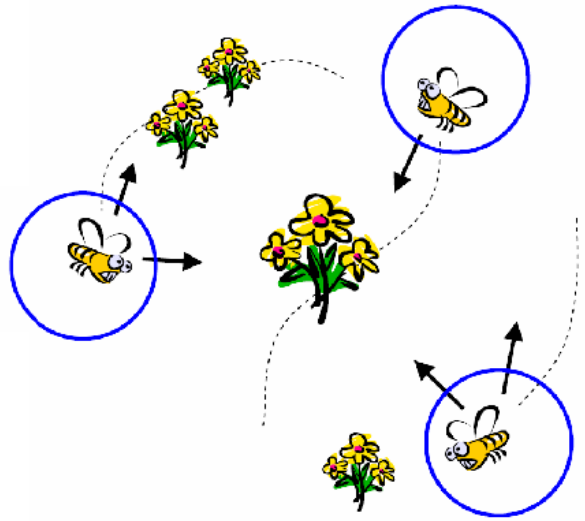
- Randomly pick N agents $\mathbf{x}$ (i.e. the population).
- Pick three distinct agents $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ from the population, different from $\mathbf{x}$.
- Compute the *trial* vector $\mathbf{y} = (y_1, \ldots, y_n)$ as follows. For $i = 1, \ldots, n$, pick $r_i \in U(0, 1)$. If $r_i < p$, then $y_i = a_i + w(b_i - c_i)$, otherwise $y_i = x_i$.
- If $f(\mathbf{y}) \leqslant f(\mathbf{x})$, then replace $\mathbf{x}$ with the trial vector $\mathbf{y}$, otherwise keep $\mathbf{x}$.

Pick the best agent from the population and return it as the best found solution.

Demonstrations: 3D, Airfoil geometry, Generative design
Question: How can we use the generative design idea in ML?

Particle swarm optimization

Particle swarm optimization is a metaheuristic algorithm that solves the optimization problem by iterative moving candidate solutions (*particles*) with certain *velocity*. Each particle's movement is influenced by its local best known position and is guided by the swarm's best known position that is updated as better positions are found by other particles.



Let s be the number of particles in the swarm, with positions x_i and velocities v_i. Let p_i be the best known position of particle i and g the one of the entire swarm. All the vectors are d-dimensional

The values b_l and b_u represents the lower and upper boundaries of the search-space. The parameters ω, ϕ_p, and ϕ_g are selected manually and control the behaviour and efficiency of the method.

The choice of parameters can have a large impact on optimization performance and is the subject of much research.

Particle swarm optimization

Algorithm

Initialize the swarm's best known position g with a random number
For each particle $i = 1, \ldots, s$ do:
  Initialize the particle's position with a uniformly distributed $x_i \sim U(b_l, b_u)$
  Initialize the particle's best known position to its initial position: $p_i \leftarrow x_i$
  If $f(p_i) < f(g)$ then update the swarm's best known position: $g \leftarrow p_i$
  Initialize the particle's velocity: $v_i \sim U(-|b_u - b_l|, |b_u - b_l|)$
Until a termination criterion is met (e.g. a certain number of iterations is reached):
  for each particle $i = 1, \ldots, s$ do
    for each $d = 1, \ldots, n$ do
      Pick random numbers: $r_p, r_g \sim U(0, 1)$
      Update the velocity: $v_{i,d} \leftarrow \omega v_{i,d} + \phi_p r_p (p_{i,d} - x_{i,d}) + \phi_g r_g (g_d - x_{i,d})$
    Update the position: $x_i \leftarrow x_i + v_i$
    if $f(x_i) < f(p_i)$ then update the particle's best known position: $p_i \leftarrow x_i$
      if $f(p_i) < f(g)$ then update the swarm's best known position: $g \leftarrow p_i$

Demonstration: Example 1, Example 2

Thank you for your attention!