FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION
OF HIGHER EDUCATION
ITMO UNIVERSITY

Report

on the practical task No. 4

Algorithms for unconstrained nonlinear optimization. Stochastic and metaheuristic algorithms

Performed by

Dmitry Grigorev, Maximilian Golovach

J4133c

Accepted by

Dr Petr Chunaev

St. Petersburg

2022

1. Goal of the work

The goal of this work consists of the following points:

- to become familiar with some stochastic and metaheuristic algorithms in the context of nonlinear optimization problems;
- to apply the methods on practical problems and compare them with each other;
- besides, to apply Nelder-Mead and Levenberg-Marquardt algorithms to these problems and compare obtained results with results of the metaheuritic algorithms.

2. Formulation of the problem

In this task here are two problems under consideration:

2.1. Problem *I*

Let $x_k = \frac{3k}{1000},\ k = 0, \ldots, 1000$ and y_k are defined as follows with i.i.d. $\delta_k \sim N(0, 1)$:

$$y_k = \begin{cases} -100 + \delta_k & \text{if } f(x_k) < -100, \\ 100 + \delta_k & \text{if } f(x_k) > 100, \\ f(x_k) + \delta_k & \text{otherwise,} \end{cases}$$

where $f(x) = \frac{1}{x^2 - 3x + 2}$ — rational function with two vertical asymptotes $x = 1$ and $x = 2$. The problem is to approximate the generated data $(x_k, y_k)_{k=0}^{1000}$ by rational function

$$F(x, a, b, c, d) = \frac{ax + b}{x^2 + cx + d}$$

which optimally fits the data in the sense of least squares

$$D(a, b, c, d) = \sum_{k=0}^{1000} (F(x_k, a, b, c, d) - y_k)^2 \to \min_{(a,b,c,d)}.$$

The solution has to be found with precision $\varepsilon = 10^{-3}$ in at most 1000 iterations.

2.2. Problem *II*

Given the data from `https://people.sc.fsu.edu/~jburkardt/datasets/cities/cities.html` with at least 15 cities having land transport connections between them, we have to tackle the corresponding Travelling Salesman Problem.

These two problems has to be tackled with both stochastic and metaheuristic algorithms such as:

- Simulated Annealing,
- Particle Swarm,
- Differential Evolution

and deterministic Nelder-Mead and Levenberg-Marquardt algorithms.

All functions which will be obtained have to be visualized with the given data and the line which generates these data. Furthermore, we have to compare the algorithms in terms of precision, number of iterations and number of function evaluations using the results.

3. Brief theoretical part

The theory is taken from [1].

Suppose that we are given with a function $f:\ G \subset \mathbb{R}^n \to \mathbb{R}$ where G is connected open set in $\mathbb{R}^n$. We need to minimize this function on G.

3.1. Simulated Annealing

The name of this algorithm comes from annealing in metallurgy: a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects.

The optimization here starts from evaluating the value of the objective function $f(x_0)$ at an initial random point $x_0 \in G$. At each step the next point x_k is obtained from the current temperature T_k which describes the mobility of moves from x_{k-1} to x_k. There are lots of rules to get to x_k from the previous point. One of them is as follows:

$$x_k^{(i)} = x_{k-1}^{(i)} + \left((x_{\max}^{(i)} - x_{k-1}^{(i)})r_{k-1}^{(i)} + (x_{\min}^{(i)} - x_{k-1}^{(i)})s_{k-1}^{(i)}\right)\frac{T_{k-1}}{T_0},\ i = 1, \ldots, n,$$

where $r_{k-1}^{(i)},\ s_{k-1}^{(i)} \sim U(0,1)$ are i.i.d (with respect to both upper and lower indices) uniformly distributed random variables and $x_{\min}^{(i)}$ and $x_{\max}^{(i)}$ define the search range for the solution. There are also lots of ways to define the rule to which the sequence $\{T_k\}$ obeys. Here we consider the following one with the **annealing coefficient** $p \geq 1$:

$$T_k = T_0\left(1 - \frac{k-1}{k_{\max}-1}\right)^p,$$

where $k_{\max}$ is the maximum number of iterations. We accept the new point x_k if $f(x_k) \leq f(x_{k-1})$ (however, the are also a number of ways to accept the new point but we limited ourselves on this one).

3.2. Differential Evolution

Let we have an initial population $x_1^{(0)}, \ldots, x_m^{(0)} \in G$ of constant volume m. At each iteration for each $j = 1, \ldots, m$ **mutant individual** $v_i^{(k)}$ is obtained from the elements of previous population $x_1^{(k-1)}, \ldots, x_m^{(k-1)}$. There are a large variety of ways to produce $v_i^{(k)}$:

$$v_i^{(k)} = x_i^{(k-1)} + K(x_a^{(k-1)} - x_i^{(k-1)}) + F(x_b^{(k-1)} - x_c^{(k-1)}); \tag{1}$$

$$v_i^{(k)} = x_i^{(k-1)} + F(x_b^{(k-1)} - x_c^{(k-1)}); \tag{2}$$

$$v_i^{(k)} = x_a^{(k-1)} + F(x_b^{(k-1)} - x_c^{(k-1)}); \tag{3}$$

$$v_i^{(k)} = x_{\text{best}}^{(k-1)} + F(x_b^{(k-1)} - x_c^{(k-1)}); \tag{4}$$

$$v_i^{(k)} = x_i^{(k-1)} + K(x_{\text{best}}^{(k-1)} - x_i^{(k-1)}) + F(x_b^{(k-1)} - x_c^{(k-1)}); \tag{5}$$

$$v_i^{(k)} = x_{\text{best}}^{(k-1)} + K(x_a^{(k-1)} - x_b^{(k-1)}) + F(x_c^{(k-1)} - x_d^{(k-1)}); \tag{6}$$

$$v_i^{(k)} = x_a^{(k-1)} + K(x_b^{(k-1)} - x_c^{(k-1)}) + F(x_d^{(k-1)} - x_e^{(k-1)}); \tag{7}$$

where $x_{best}^{(k-1)}$ is the best (in terms of value of function f) individual of $(k-1)$-th generation, a, b, c, d, e are randomly chosen different numbers from the set $\{1, \ldots, i-1, i+1, \ldots, m\}$, $0 \le K \le 1$ is the **combination factor** and $0 \le F \le 1$ is the **scaling factor**. As soon as the mutants are obtained, the **trial individuals** $u_1^{(k)}, \ldots, u_m^{(k)}$ are created in the process of **cross-over** with its constant $C \in [0, 1]$:

$$u_{ij}^{(k)} = \begin{cases} u_{ij}^{(k)} & \text{if } r_{ij}^{(k)} \le C \text{ or } j \ne s_{ij}^{(k)} \\ x_{ij}^{(k-1)} & \text{otherwise,} \end{cases}$$

where $r_{ij}^{(k)} \sim U(0, 1)$, $s_{ij}^{(k)}$ is j-th element in the vector $s_i^{(k)}$ which is a random permutation of the set $\{1, \ldots, m\}$. In other words, the trial individual has some components of the mutant individual and at least one component of its parents $x_i^{(k-1)}$. Then the new generation is produced:

$$x_i^{(k-1)} = \begin{cases} u_i^{(k)} & \text{if } f(x_i^{(k-1)}) \ge f(u_i^{(k)}), \\ x_i^{(k-1)} & \text{otherwise.} \end{cases}$$

Common choice of parameters is as follows: $C = 0.9,\ F = K = 0.8$. Moreover, the larger is m and the smaller are F and K then the more robust is the algorithm and the more expensive is the optimization process.

3.3. Particle Swarm

This method replicates the behavior of birds looking for food and following the leader. The idea is that each individual in in the swarm knows both its own best position and best

position of the whole swarm.

Let we are given with an initial population $x_1^{(0)}, \ldots, x_m^{(0)}$ of size m. At each iteration the position of the i-th individual is calculated with respect to the formula:

$$x_i^{(k)} = x_i^{(k-1)} + v_i^{(k)},$$

where $v_i^{(k)}$ is the velocity of i-th individual which is the function of previous velocity $v_i^{(k-1)}$, i-th individual's best position $\widetilde{x}_i$ and the population's best position $\widetilde{x}$:

$$v_i^{(k)} = W v_i^{(k-1)} + C_1 r_1 (\widetilde{x}_i - x_i^{(k-1)}) + C_2 r_2 (\widetilde{x} - x_i^{(k-1)}),$$

where $r_1, r_2 \sim U(0,1)$ are the random variables, C_1 is the **cognitive factor**, C_2 is the **social factor** and W is the **inertia factor**.

4. Results

5. Data structures and design techniques used in algorithms

In the implementation of the algorithms and for random variables generation the Python's package Numpy is used since it allows to vectorize calculations what accelerates the evaluations.

6. Conclusion

As the result of this work, we considered direct methods of optimization in the cases of one and two variables. The methods were compared by the number of iterations required to obtain the solution with given precision, the number of function evaluations and precision. All of the considered methods except Nelder-Mead method were implemented in Python programming language. Discussion for data structures and design techniques which were used in the implementations is provided. The work goals were achieved.

7. Appendix

Algorithms implementation code is provided in [2].

Bibliography

1. Cavazzuti M. Optimization Methods. — Springer Berlin, Heidelberg, 2013. — ISBN: 9783642311864.
2. Grigorev D., Golovach M. Code repository. — `https://github.com/dmitry-grigorev/AlgoAnalysisDevelopment`. — 2022.
3. Deisenroth M. P., Faisal A. A., Ong C. S. Mathematics for Machine Learning. — Cambridge University Press, 2020.