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
Carlos Andres Daza Rachen

Academic group
Accepted by
Dr Petr Chunaev

St. Petersburg
2022

Goal

The use of stochastic and metaheuristic algorithms (Simulated Annealing, Differential Evolution, Particle Swarm Optimization) in the tasks of unconstrained nonlinear optimization and the experimental comparison of them with Nelder-Mead and Levenberg-Marquardt algorithms

Formulation of the problem

Compare stochastic and metaheuristic algorithms, solve travelling salesman problem applying the simulated annealing method.

Brief theoretical part [1]

Stochastic methods use randomization strategically to help explore the design space for an optimum. Randomness can help escape local optima and increase the chances of finding a global optimum. Stochastic methods typically use pseudo-random number generators to ensure repeatability. A large amount of randomness is generally ineffective because it prevents us from effectively using previous evaluation points to help guide the search.

Simulated annealing borrows inspiration from metallurgy. Temperature is used to control the degree of stochasticity during the randomized search. The temperature starts high, allowing the process to freely move about the search space, with the hope that in this phase the process will find a good region with the best local minimum. The temperature is then slowly brought down, reducing the stochasticity and forcing the search to converge to a minimum. Simulated annealing is often used on functions with many local minima due to its ability to escape local minima.

Metaheuristic methods are based on a heuristic method that does not rely on the type of the problem. The metaheuristic method can be distinguished into two which are metaheuristic with single-solution based (local search) and metaheuristic based on population (random search). Population methods involve optimization using a

collection of design points, called individuals. Having a large number of individuals distributed throughout the design space can help the algorithm avoid becoming stuck in a local minimum.

Differential evolution algorithm attempts to improve each individual in the population by recombining other individuals in the population according to a simple formula.5 It is parameterized by a crossover probability p and a differential weight w, tipically between 0.4 and 1. For each individual x.

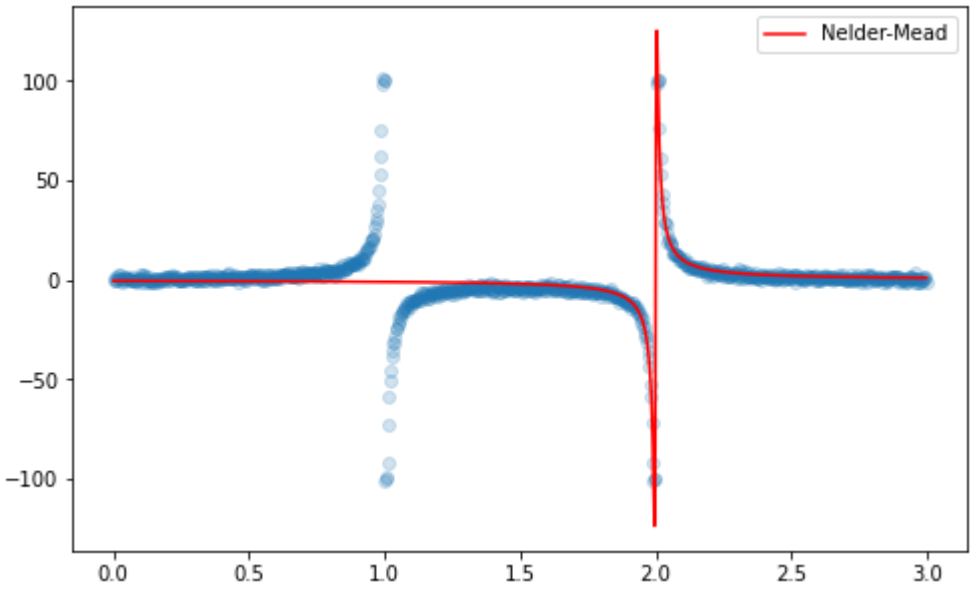
Particle swarm optimization introduces momentum to accelerate convergence toward minima. Each individual, or particle, in the population keeps track of its current position, velocity, and the best position it has seen so far. Momentum allows an individual to accumulate speed in a favorable direction, independent of local perturbations.

Results

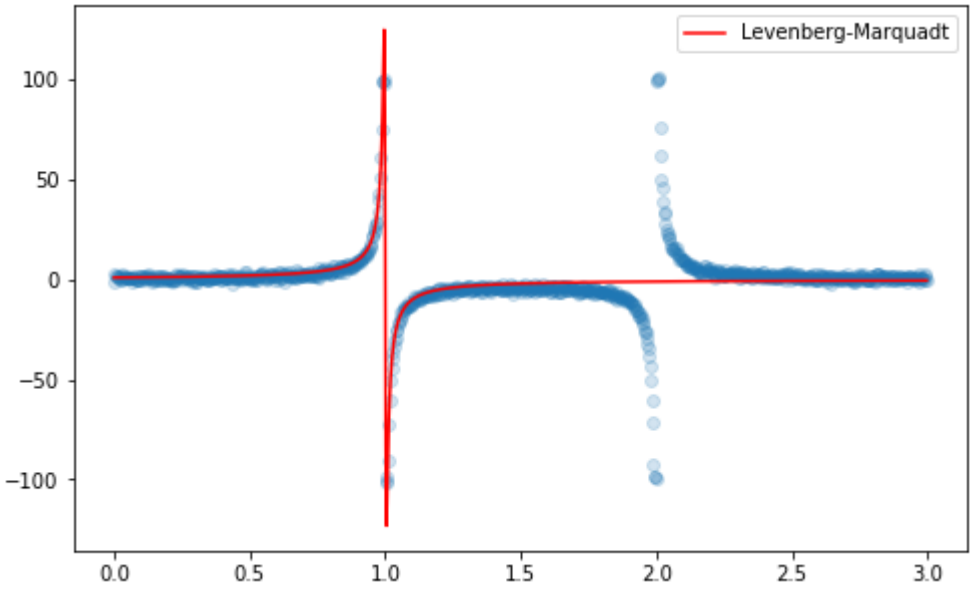
1. Minimization

	a	b	c	d	Fx value	Iterations	Fx calls
Differential Evolution	0.997252	-1.994	-3.99912	3.99826	135595	197	11950
Simulated Annealing	-0.999652	1.00016	-2.00101	1.00102	135520	1000	8811
Nelder-Mead	-0.219791	0.425709	-3.95278	3.90626	255831	354	604
Levenberg-Marquadt	-1.00079	1.00127	-2.00091	1.00093	67762.6	215	215

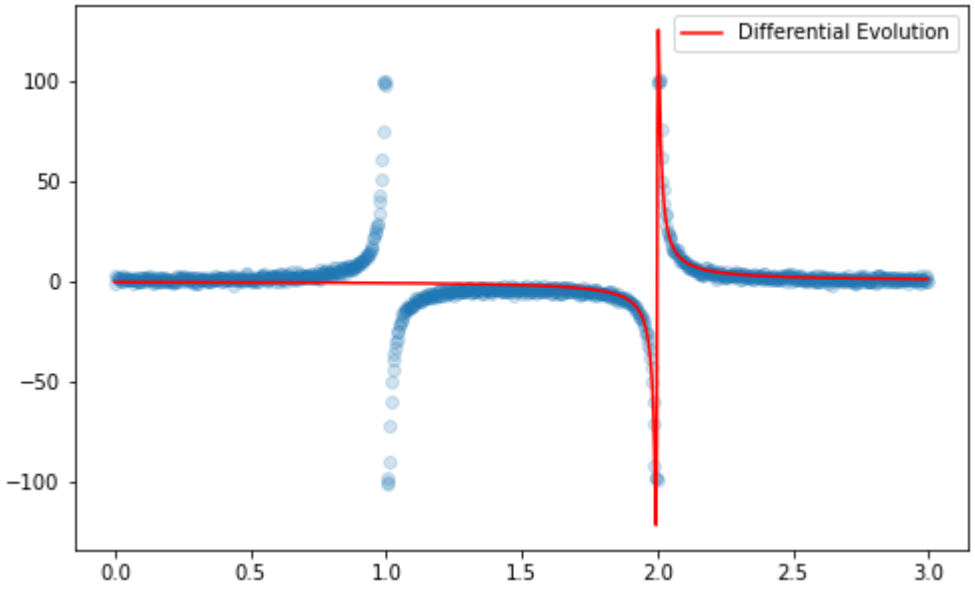
1.1 Nelder - Mead



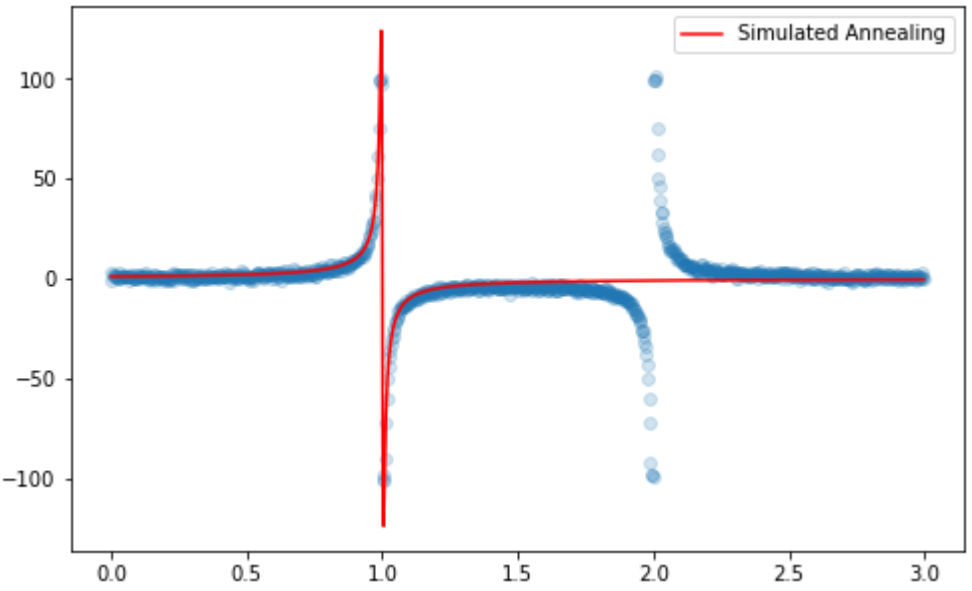
1.2 Levenberg-Marquadt



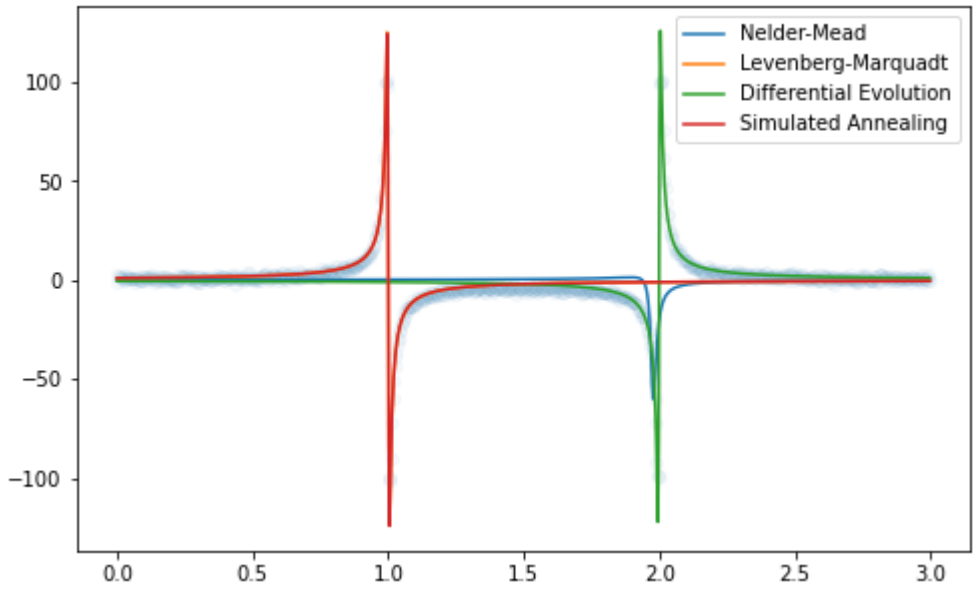
1.3 Differential Evolution



1.4 Simulated Annealing

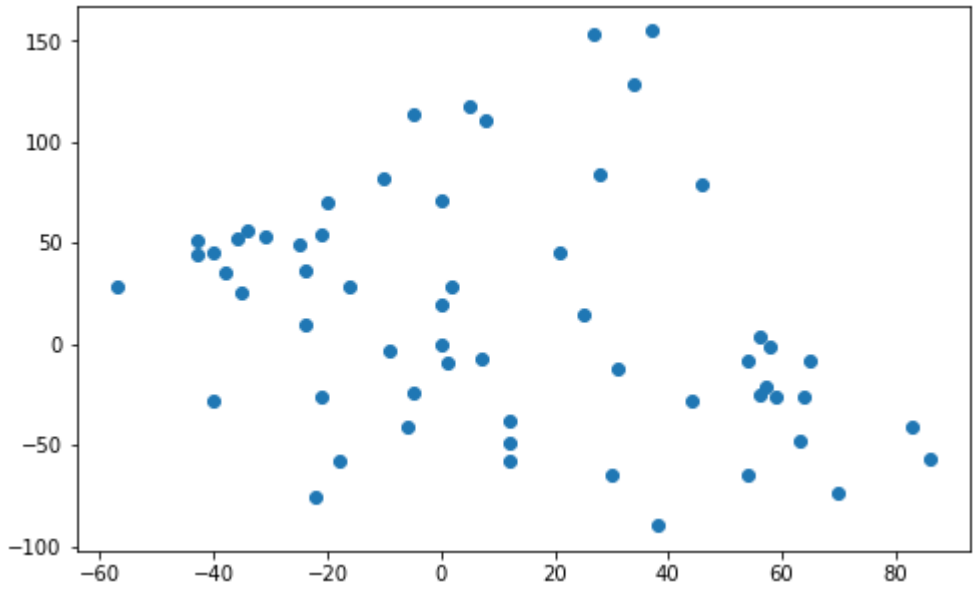


1.5 Comparison

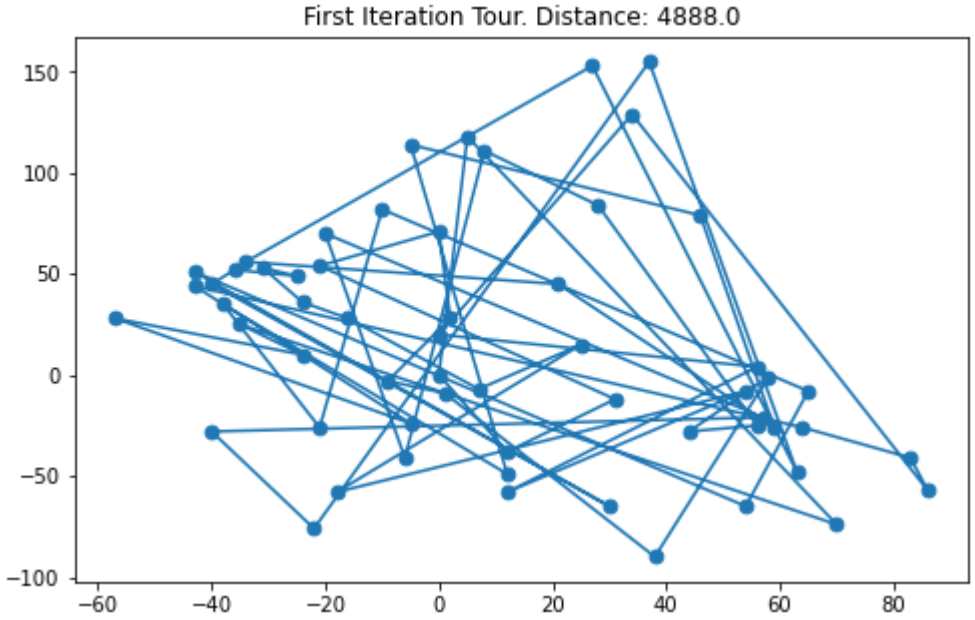


2. Travelling Salesman Problem.

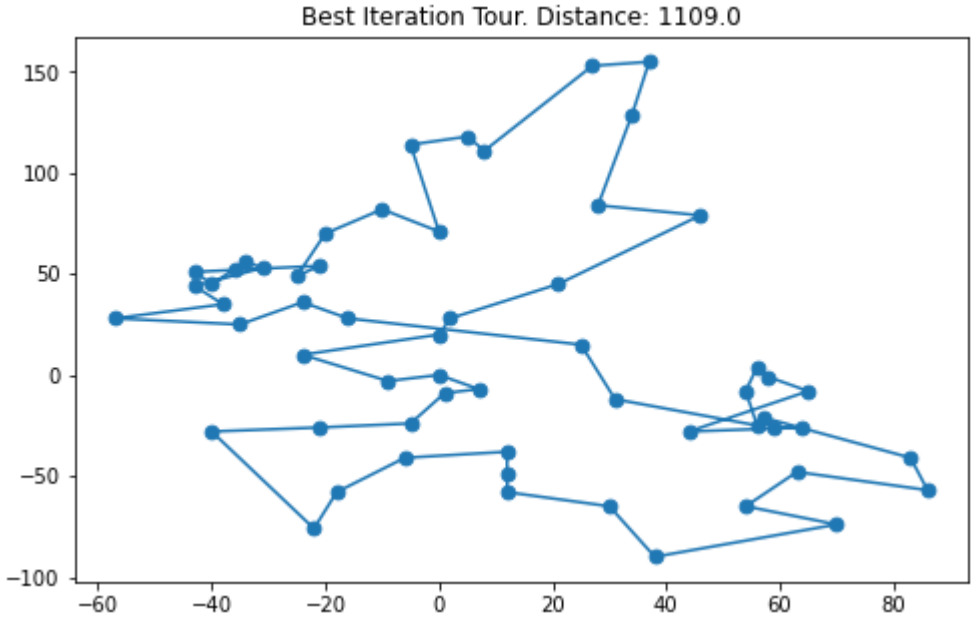
2.1 Travel points



2.2 First iteration



2.3 Best iteration tour distance



Conclusions

- Differential evolution method had the best performance with fairly good accuracy, with the other methods you can get errors with “maximun number of function evaluations”
- Levenberg-Marquadt y Simulated Annealing give similar results, but Simulated Annealing has longer number of iterations.
- It’s impressive the difference between the first iteration and the last one, that shows the ability of the algorithm to scape local minima and find best aproximations

Bibliography

[1] Kochenderfer, Algorithms for optimization