International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 2.102

Optimization of Ornstein-Uhlenbeck Models in Pair Trading: Heuristic and Metaheuristic Approaches

Nikola Basheski[1], Vesna Dimitrievska Ristovska, PhD [2]

[1] Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University of Skopje,
ul.Rugjer Boshkovikj 16, Skopje, North Macedonia
Email: *nikola.basheski[at]students.finki.ukim.mk*

[2]Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University of Skopje,
ul.Rugjer Boshkovikj 16, Skopje, North Macedonia
Email: *vesna.dimitrievska.ristovska[at]finki.ukim.mk*

Abstract: ***Stochastic processes are widely used in describing time dependent sequences of random changes in both physical and social phenomena, including, relevantly to this paper, random market movements of stock prices. Optimization techniques are applied in finance, for example, in Banking processes such as minimizing losses from credit risks and changes of the value of collateral assets, and in trading - such as assessing optimal position sizes and buy/sell times. This study explores heuristic and metaheuristic optimization methods for financial modeling in pair trading. Using Genetic Algorithm and Simulated Annealing, the study aims to optimize parameters of the Ornstein-Uhlenbeck process, a stochastic model used in trading decisions. Performance is evaluated on simpler test cases before applying these methods to minimize the log-likelihood function of the process. Results show the effectiveness of these algorithms in enhancing trading strategies, providing robust insights for financial optimization.***

Keywords: Genetic Algorithm (GA), Simulated Annealing (SA), Ornstein-Uhlenbeck (O-U) process

1. Introduction

We cover the application of two different mathematical optimization methods, both of which belong to the heuristic/metaheuristic class of optimization methods, Genetic Algorithm and Simulated Annealing. Heuristic optimization algorithms handle high-dimensional spaces effectively and do not rely on gradients, which are difficult to compute in high dimensions. They often apply conceptually similar procedures, inspired by real-life occurrences such as crossover and mutation (in genetic algorithms) or annealing in metallurgy.

This paper explores the application of these two optimization methods in financial modeling. First, we evaluate their performance on two basic test functions, where we can easily compare the results with our expectations: the minimization of the lower half of a sphere centered at zero and the minimization of the Rosenbrock function. These algorithms are then applied to the Ornstein-Uhlenbeck process, a stochastic process that is both a stationary Gauss-Markov process, meaning it is Gaussian, Markovian, and temporally homogeneous.

The purpose of this study is to evaluate the effectiveness of Genetic Algorithm and Simulated Annealing in optimizing the Ornstein-Uhlenbeck process parameters for financial trading strategies. This study contributes to financial modeling by demonstrating the applicability of heuristic and metaheuristic optimization techniques to enhance trading strategies, addressing challenges in stochastic parameter estimation.

2. Application

For this model, we have selected two similar companies in the same city (NYC) and industry (real estate), Blackstone Mortgage Trust (BXMT)[9] and Appolo Commercial Real Estate Inc. (ARI)[10]. Their clientele has similar demographic composition and they are subject to similar macroeconomic effects. Their businesses are fairly similar and react similarly to the changes in their environments - so there is a correlation between statistical metrics of their performance (such as earnings, price-to-earnings ratio, EV/EBITDA, etc.) and changes over time, and similarly there is correlation between their stock prices [9][10].

We detect the correlation break by expressing the long-term relationship between the two stock prices and tracking significant spikes in their volatility. We define the target variable as the ratio of the two prices, adjusted by a factor that represents the moving average of the same ratio, and then model the adjusted ratio using the Ornstein-Uhlenbeck process [14].

$$dX_t = \theta(\mu - X_t)X_t dt + \sigma dW_t$$

where:

- X_t = the current price of the stock.
- θ = a mean reversion constant (how fast is the stock usually returning to the mean).
- μ = the mean historical price of the stock.
- σ = a constant volatility.
- W_t = a Wiener process (Brownian motion).

Assuming X_t and W_t are unit normal random variables ($N(0, 1)$) and are mutually independent.

3. Problem

The desired ability of an investor is to predict future events accurately - specifically, to predict 'big' consequential movements in the market. The basic strategy is to buy low and sell high, with buying at the lowest possible price (the bottom) and selling at the highest price possible (the top) being the optimal scenario for a trader [4]. The statistical strategy outlined above makes sense taking into consideration several caveats; one, short-selling - buying high - selling low can be profitable (in a certain context) and two, the future is unpredictable, but there are patterns that may be exploited (especially by risk averse institutions such as Banks or shadow Banks, which trade large volumes of financial derivatives at low risk), such as the correlation of stock prices of similar companies and their mean reverting behavior [2][11].

Short selling is the action where an investor or a trader bets against the price of an asset. If the trader believes that BXMT will depreciate in value over the following week, the trader may borrow 100 BXMT stocks worth $10,000 from a lender, at an interest rate or a fee (as a fixed amount) and sell the borrowed stock to another trader for the same $10,000. If the trader has correctly assessed the near future of BXMT, and the price of BXMT depreciates approximately 40% of its contract value, each stock is now worth $60. Before the week ends, according to the terms of the contract, the trader is obliged to return the borrowed stock to the lender, so the trader buys back the same stock from the market at $60 a stock, which amounts to $6,000. The trader has earned $4,000, minus the fixed coupon she may have paid to the lender upfront.

If only exploiting mean reversion, the trader would only make profit after an increase of the volatility of a single stock [5]. Big movements of the stock price in one direction are followed by big movements in the opposite direction, and small movements in the stock price in one direction are followed by small movements in the opposite direction [6][7]. This, coupled with mean reversion, increases the predictability of future movements and simplifies the trader's decision-making process, but what if the whole market crashes and there are no 'movements in the opposite direction'[2][5]?

An even safer bet is statistical arbitrage/ pair trading, where the trader makes more money if both stocks perform as she predicted, or if one doesn't - she has hedged against that risk by covering the losses from one of the stocks, with the gains from the other. So statistical arbitrage represents a two sided bet, and it exploits the correlation between the prices of similar stocks and their mean reverting behavior. With this model, the mean reversion component and the volatility are taken into consideration, by using the O-U process not only to model the price movements of a single stock or each of the stocks, individually, but by modeling their relationship [14]. This is done by using strong increases in the volatility of the relationship between the prices as an indicator of a correlation break (and a threshold for buy/sell orders), and the mean reversion component as a hedging strategy [5][6][7].

We are trying to use the two aforementioned optimization techniques to find the optimal parameters which best simulate the movements of the ratio between the stock prices. We use the historical data and the derived Ornstein-Uhlenbeck mean, standard deviation and mean reversion parameter to find the Gaussian distribution with the appropriate parameters, and then we minimize the its log-likelihood to find the optimal parameters of the O-U process which models the ratio [15]. In statistical arbitrage, the optimized volatility defines the upper and lower thresholds for trading: when the ratio exceeds the threshold above the mean, we buy stock A and sell stock B; when it drops below the threshold, we buy B and sell A. The optimized mean reversion rate determines how quickly we adjust our positions [4].

4. Objective Functions

Three objective functions are used as test cases for the optimization methods. The first two functions are relatively straightforward, allowing us to easily verify whether the optimization algorithm successfully converges to the correct minima.

- The lower half of a sphere with a radius of 2: $f(x,y) = -\sqrt{4-x^2-y^2}$
- The Rosenbrock function:$f(x,y) = (a-x)^2 + b(y-x^2)^2$, where $a = 1$ and $b = 100$. These parameters are standard in the optimization literature as they create a steep, narrow "valley," posing a challenge for optimization algorithms in locating the global minimum.

The third function, designed for our optimization process, is the log-likelihood function of the Ornstein-Uhlenbeck process. Its purpose is to estimate the parameters that maximize the joint probability of the observed data. A straightforward approach involves using maximum likelihood estimation (MLE) to identify the parameter values that enable the simulated O-U process to best replicate the observed data's dynamics [8].

For independent and identically distributed random variables, the joint density function $f(X|\theta,\mu,\sigma)$ is the product of univariate density functions:[2][13]

$$f(x_1,x_2,\dots,x_n|\theta,\mu,\sigma) = f(x_1|\theta,\mu,\sigma)\dots f(x_n|\theta,\mu,\sigma) = \prod_{i=1}^{n} f(x_i|\theta,\mu,\sigma) \quad (1)$$

where $x = (x_1,x_2,\dots,x_n)$. For convenience, we typically take the natural logarithm of the likelihood function:[13]

$$l(\theta,\mu,\sigma) = \sum_{i=1}^{n} lnf\left(x_{t_i}|\theta,\mu,\sigma\right) \quad (2)$$

This function, known as the log-likelihood function, serves as the objective for our optimization process, where we seek the parameter values that minimize this function:[13]

$$\hat{\kappa}(x) = \underset{\theta,\mu,\sigma}{argmin}\, l(\theta,\mu,\sigma|x) \quad (3)$$

The closed-form solution for the Ornstein-Uhlenbeck process is given by:[2][8][12][15][16]

$$X_{t+\Delta t} = X_{te^{-\theta t}} + \mu\left(1-e^{-\theta t}\right) + \sigma\sqrt{\frac{1-e^{-2\theta t}}{2\theta}} \quad (4)$$

Consequently, the distribution of the Ornstein-Uhlenbeck process follows a Gaussian distribution, characterized by the parameters derived from the closed-form solution to the O-U stochastic differential equation: [2][8][12][15][16]

$$f(x_t|\theta,\mu,\sigma) = \sqrt{\frac{1}{2\pi\sigma^2}}\exp\left(-\frac{\left(x_t-\left(x_0e^{-\theta t}+\mu(1-e^{-\theta t})\right)\right)^2}{2\left(\frac{\sigma^2}{2\theta}(1-e^{-2\theta t})\right)}\right) \quad (5)$$

5. Algorithms

For the purposes of simplicity, we use simpler functions (the aforementioned lower half of a sphere with radius of 2, centered at 0 and the Rosenbrock function) to demonstrate the performance of these algorithms. Then we apply these algorithms in the actual code simulating the O-U process of the model [2][8].

5.1 Genetic Algorithms

Genetic Algorithms (GA) are optimization techniques inspired by processes involving genetics and inheritance in natural selection. It's a metaheuristic method of optimization and belongs to a wider class of evolutionary algorithms (EA). Genetic Algorithms incorporate randomness (in the selection of individuals as parents from the population, the selection of genes inherited in each child, from each parent, random mutations occurring each generation), which reflects the random nature of survival, crossover, inheritance, mutations and other such processes in natural selection [17][19][20].

1) Fitness Calculation:
Define a function that calculates the fitness of an individual. We are using the Sphere function, the Rosenbrock function, or the Log-Likelihood of the Gaussian function. Since we are looking for the local minimum, the fitness function is formulated appropriately for the guesses to be lower and lower.

2) Generate Initial Population:
We choose a somewhat arbitrary set of boundaries, mostly informed by expert opinion [0.001, 3.0] where we expect the initial guesses to be made. The population is made of decimal numbers between 0.001 and 3 (in a uniformly distributed manner - equal probability for each decimal between the bounds) and has a defined size of 100, which is large enough to provide genetic diversity and prevent premature convergence, yet small enough to be computationally manageable. The number of genes should be the number of parameters of the function we are trying to optimize. So in the case of the lower half of the sphere and Rosenbrock function, we have 2 functions since both of these functions are two-dimensional. In the case of the log likelihood of the Gaussian distribution, however, we are using 3 parameters (the estimated optimal mean, standard deviation, and mean reverting rate), so each individual will have 3 genes.

3) Select Individuals Based on Fitness:
We make a list of all the individuals in the population at stage i (in the form of a dataframe for convenience). We evaluate each individual according to the fitness function. We then sort them in ascending order because we treat those with the lowest scores as the best candidates (because we are looking for a minimum). After we have sorted the table according to the scores of the individuals, in ascending order, we split the population in half, only taking the upper half (with the lowest scores).

4) Crossover Function:
In the case of the sphere and the Rosenbrock function, the crossover function is simple. Initially, the children are taken as copies of their parents, but different from each other. When we construct an original new population, the children will combine one gene from the first parent and another from the other parent. They have to inherit different genes. For the log-likelihood of the Normal distribution, the simplest way is to take half of the genes from one parent with half of the genes from the other, but in order for the gene selection to be unbiased, the index of the gene inherited in one child is randomly selected from the number of genes in each individual.

5) Mutation Function:
We define some mutation rate. Then we pick a decimal point between 0.001 and the value of each gene in an individual and we subtract these values from the genes of the individuals in the original population. With this, we ensure that we properly direct the mutation, while also maintaining that the change of the mutation is still smaller than the value of the gene which is being mutated. Since we only mutate if a randomly chosen decimal point is smaller than the mutation rate, not all individuals have their genes mutated. It means we have individuals mutate at the mutation rate - meaning the mutation rate is the percentage of mutating iterations.

6) Select Parents for Crossover:
We simply select two random individuals to be the parents in the current iteration

7) Generate New Population:
We use the previous functions to define a new population. For each individual, we pick two parents, we use the crossover function to come up with children with an original set of genes and then we apply mutation to the genes to some of them, further increasing the diversity of the population in the direction of the optimal solution.

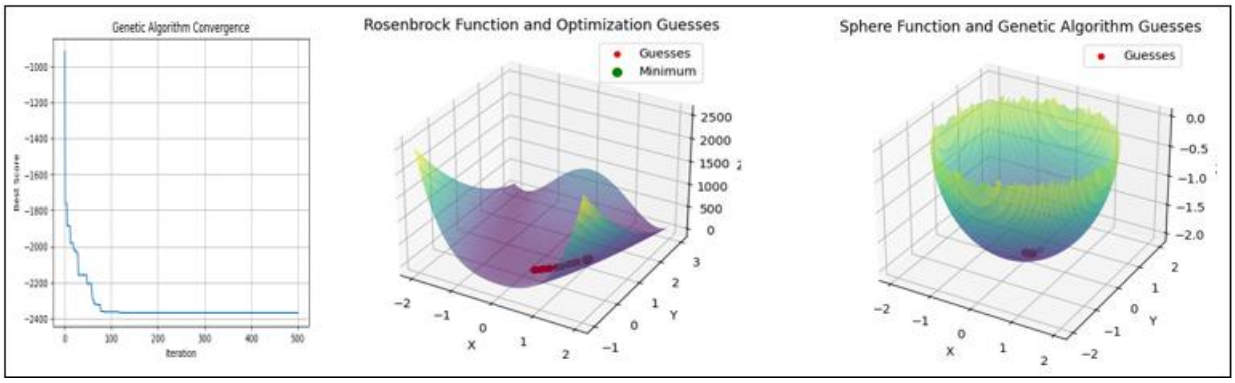
8) Main Genetic Algorithm Function:

a) We initialize 500 maximum iterations for balance between computational efficiency and convergence, providing sufficient exploration space for all the objective functions.
b) We initialize mutation rate of 0.01 which maintains diversity, preventing premature convergence while preserving the trend of the solutions towards the minimum.
c) Initialize the population at 100.
d) Initialize the best solution and its score at 0.
e) Initialize a list to store the convergence history and guesses.
f) Main optimization loop:
   - Evaluate fitness scores for each individual in the population.
   - Sort individuals based on their fitness scores.
   - Select the best individual and its score.
   - Update the best solution and its score if a better one is found.

- Append the best score to the convergence history.
- Append the current guesses to the guesses list.
- Select top individuals for reproduction.
- Generate a new population by crossover and mutation.

g) Return the best solution, its score, and the guesses.

The visualization of the guesses made by the Genetic algorithm for finding the optimum of the sphere and the Rosenbrock function are displayed on the below.



5.2 Simulated Annealing

Simulated annealing (SA) is a probabilistic technique and a heuristic optimization based on the annealing process in metallurgy which involves heating a material above a certain temperature threshold, maintaining a suitable temperature for an appropriate amount of time and then cooling. The acceptance criterion for the Simulated Annealing algorithm is constructed in such way as to allow for worse solutions to be accepted at some frequency (rate determined by set probability). This helps SA escape local optima although the method decreases the probability of accepting worse guesses as the number of iterations increases - which directs the algorithm towards the global optimum. It is often more effective to use Simulated annealing to find precise optima in a finite period of time, rather than using something like Gradient Descent which approximates optimal solutions [18][21].

1) Define the Objective function:

Objective (x, y) calculates the value of the function $-\sqrt{4-x^2-y^2}$, representing a negative half-sphere, or the Rosenbrock function $(a-x)^2+b(y-x^2)^2$ or the log-likelihood of the Standard Normal distribution.

2) Simulated Annealing:

simulated_annealing(objective, step_size, initial_temperature, n_iterations, initial_guess) implements the simulated annealing algorithm.

a) Initialization: Sets initial parameters, solutions, and temperature. It initializes the solutions by evaluating the objective function (the sphere, Rosenbrock, or log-likelihood of Gaussian function) at the current guess (which at iteration 0 is the guess for the initial parameters). The Simulated Annealing algorithm uses a step size of 0.1, an initial temperature of 10, and 10,000 iterations to optimize the negative half-sphere objective function. The step size (0.1) ensures small perturbations to explore the solution space locally, balancing exploration and exploitation. The initial temperature (10) provides a sufficiently high probability to accept worse solutions early, preventing the algorithm from being trapped in local minima.

b) Main Loop: Iterates over the specified number of iterations which in this case is 10000.

- Generates new parameters by adding a random perturbation to the current parameters. This is done by picking a random number from the current parameters and multiplying by some (small) number for the step size and then adding it to the current parameters. This is done in the line:
- new_params = current_params + randn(len(current_params)) * step_size
- Evaluates the objective function for current and new parameters.
- new_cost<best_solution: checks if the new guess is actually better than the last one (is it smaller).
- **Acceptance Criterion:** In simulated annealing, there are two options as conditions for acceptance: 1. if the new proposal follows the trend towards the extreme point or if it satisfies some rule (such as the Metropolis criterion) where it accepts the new parameters if they improve the solution. If not, it may still accept them based on a probability that decreases with temperature. So, given $\delta = cost_i - cost_{i-1}$ (change between the new and the current cost) we define a threshold for acceptable probability $P_{acceptable} = e^{-\delta/t}$ (Metropolis criterion) we pick a random number from a uniformly distributed set between 0 and 1 and we check if it is smaller than the acceptable probability $\mathcal{N}(0, 1) < P_{acceptable}$ Updates the temperature to gradually reduce (cooling) $t_i = t_0/(1 + N_{iter})$.
- Records the best solution found so far.

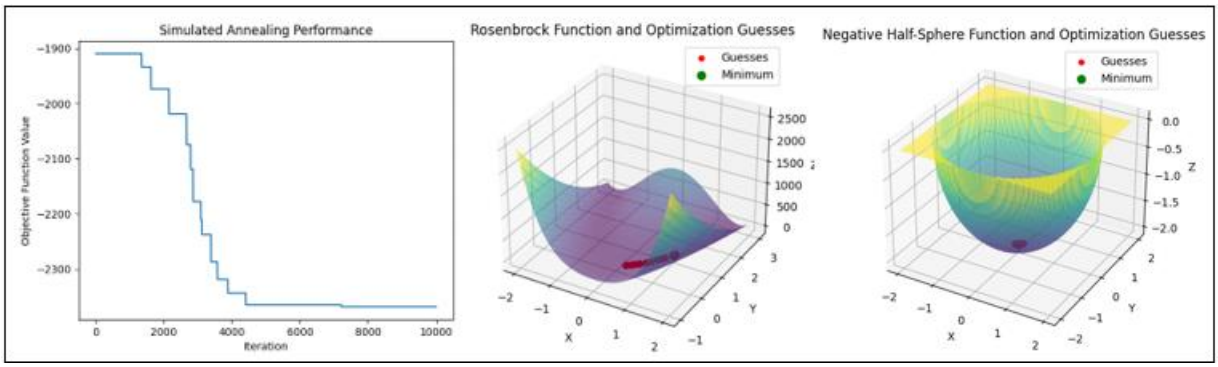
3) Parameters:

Defines the parameters for the simulated annealing algorithm: `n_iterations`, `step_size`, and `temp`.

4) Call Simulated Annealing Function:

Executes the simulated annealing algorithm and stores the best parameters and score found.

The visualization of the guesses made by the Simulated Annealing algorithm for finding the optimum of the sphere and the Rosenbrock function are displayed below.



6. Results

After successfully applying the algorithms to find the optimal points for simpler functions like the sphere and Rosenbrock, we proceed to the more complex task of optimizing the Ornstein-Uhlenbeck process by minimizing the log-likelihood of the Gaussian distribution, with parameters derived from the O-U process.

As mentioned in the process of describing the algorithms, most of the parameters are tuned by incorporating expert judgment and manual search, guided by our expectations for the minima. The expert judgment is meant to combine reasonable starting point given the objective functions and computational efficiency. For the purpose of validation, we establish a benchmark threshold for the optimized parameters using the function scipy.optimize.minimize(fun=log_likelihood_OU, x0=theta0, args=(vol,), constraints=cons_set). The results yield a mean close to the mean of the modified ratio in the historical data at 1.001, a mean reversion parameter of 0.106 and a standard deviation of 0.024. The minimum log-likelihood value at these parameters is -2369.111.

The optimized parameters using the Genetic Algorithm are as follows: 0.9979 for the mean, 0.1102 for mean reversion parameter and 0.0247 for the standard deviation. The minimum log-likelihood of the Gaussian is -2368.700.

The optimized parameters using the Simulated Annealing method are as follows: 1.000 for the mean, 0.10342319 for mean reversion parameter and 0.02465 for the standard deviation. The minimum log-likelihood of the Gaussian is -2368.865.

7. Conclusions

Optimization techniques are widely used in finance, such as in banking to minimize losses from credit risks and fluctuations in the value of collateral assets, and in trading to determine optimal position sizes and timing of buy/sell decisions. In this paper, we explore two heuristic/metaheuristic optimization algorithms—Genetic Algorithm and Simulated Annealing—aimed at maximizing profit in a pair trading or statistical arbitrage strategy. We begin by testing these algorithms on simpler cases, where the success can be easily verified and then apply them to minimize the log-likelihood function of the Ornstein-Uhlenbeck (O-U) process, which yields the optimal volatility and mean reversion parameters for informing buy/sell decisions.

This research demonstrates the effectiveness of genetic algorithm and simulation in optimizing Ornstein-Uhlenbeck process parameters for financial trading strategies. By applying these methods to stochastic modeling, the study provides a robust framework for traders to enhance decision-making and profit margins, contributing valuable insights into financial optimization.

Acknowledgements
This research was partially supported by Faculty of Computer Sciences and Engineering, Univ. Ss. Cyril and Methodius, Skopje.

References

[1] Basheski, N., & Dimitrievska Ristovska, V. (2024). Classical optimization methods for an Ornstein-Uhlenbeck process-based model in pair trading. In Proceedings of the International Scientific Conference ”MATHMODEL 2024”, Borovets, Bulgaria
[2] Basheski, N., & Bakeva, V. (2024). Stochastic processes in finance and trading. In CIIT 2024: 21st International Conference on Informatics and Information Technologies, Faculty of Computer Science and Engineering, University of Ss. Cyril and Methodius (in print).
[3] Beavis, B., & Dobbs, I. M. (1990). Static Optimization. In Optimization and Stability Theory for Economic Analysis (pp. 32–72). Cambridge University Press. ISBN 0-521-33605-8.
[4] Pervushyna, V. (2023). Optimal Stopping in Pairs Trading: Ornstein-Uhlenbeck Model. Hudson & Thames.
[5] Leung, T., & Li, X. (2015). Optimal Mean Reversion Trading: Mathematical Analysis and Practical Applications. World Scientific Publishing Company.
[6] Cont, R. (2005). Volatility Clustering in Financial Markets: Empirical Facts and Agent–Based Models. Springer.
[7] Mandelbrot, B., & Hudson, L. R. (2006). The (Mis) behavior of Markets: A Fractal View of Risk, Ruin, and Reward. Basic Books.
[8] QuantPy. (2022). Trading stock volatility with the Ornstein-Uhlenbeck process. GitHub.
[9] Yahoo! (2024, January 12). Blackstone Mortgage Trust, Inc. (BXMT) Stock Historical Prices & Data. Yahoo! Finance. https://finance.yahoo.com/quote/BXMT/history?p=BXMT.

[10] Yahoo! (2024, January 12). Apollo Commercial Real Estate Finance, Inc. (ARI) Stock Historical Prices & Data. Yahoo! Finance. https://finance. yahoo.com/quote/ARI/history?p=ARI.
[11] Vasicek, O. (1977). An Equilibrium Characterization of the Term Structure. Journal of Financial Economics, 177–188.
[12] Øksendal, B. (2014). Stochastic Differential Equations: An Introduction with Applications. Springer.
[13] Dobson, J. A., & Barnett, G. A. (2008). An Introduction to Generalized Linear Models. Chapman and Hall/CRC, 3(4), 77–79.
[14] Brus, D., & Alber, D. (2020). The Ornstein-Uhlenbeck Process and Its Applications to Pairs Trading. KTH Royal Institute of Technology.
[15] Hendricks, H. J., Ongala, J., & Ntirampeba, D. (2021). Modification of the Ornstein Uhlenbeck Process to Incorporate the Influence of Speculation on Volatility in Financial Markets. Journal of Mathematics and Statistics. https://thescipub.com/abstract/jmssp.2022.1.10.
[16] cantaro86.(2022).Financial-Models-Numerical-Methods Public. GitHub. https://github.com/cantaro86/Financial-Models-Numerical-Methods.
[17] Brownlee, J. (2021, October 12). Simple Genetic Algorithm From Scratch in Python. Machine Learning Mastery. https://machinelearningmastery.com/ simple-genetic-algorithm-from-scratch-in-python/.
[18] Brownlee, J. (2021, October 12). Simulated Annealing From Scratch in Python. Machine Learning Mastery. https://machinelearningmastery. com/simulated-annealing-from-scratch-in-python/.
[19] Mitchell, M. (1996). An Introduction to Genetic Algorithms. MIT Press. ISBN 9780585030944.
[20] Burkhart, M. C., & Ruiz, G. (2023). Neuroevolutionary Representations for Learning Heterogeneous Treatment Effects. Journal of Computational Science, 71: 102054. doi:10.1016/j.jocs.2023.102054. S2CID 258752823.
[21] What is Simulated Annealing? (2023, May 13). www.cs.cmu.edu. https://www.cs.cmu.edu/~./ph/859F09/WhatIsSimulatedAnnealing.html.