Optimization of Automated Trading Strategies through Nature-Inspired Computing

1st Bulat Akhmatov
Innopolis University
Innopolis, Russia
b.akhmatov@innopolis.university

2nd Alexandra Vabnits
Innopolis University
Innopolis, Russia
a.vabnits@innopolis.university

3rd Sofia Shulyak
Innopolis University
Innopolis, Russia
s.shulyak@innopolis.university

Abstract—**The field of automated trading strategies is continuously evolving, with researchers and practitioners exploring new techniques and approaches, such as nature-inspired computing, to improve trading performance. Our proposal presents an innovative approach to optimize automated trading strategies by harnessing the power of nature-inspired computing techniques, namely genetic algorithms (GA). This research aims to enhance the performance of the Freqtrade open-source crypto trading bot by developing a program for the optimization of hyperparameters of trading strategies.**

Index Terms—**trading, trading automation, stock prices, nature-inspired computing, evolutionary algorithm, genetic algorithm**

I. Introduction

Stock prices exhibit chaotic, noisy, and non-stationary characteristics, making them difficult to predict and analyze. This complexity limits the effectiveness of classical programming approaches in addressing the problem. However, statistical tests have demonstrated that time series, such as daily FOREX currency rates, are not randomly distributed [1]. This provides an example of a field where nature-inspired and ML approaches can be applied effectively.

Our primary objective is to discover genetic algorithms application to the domain of trading through an attempt to enhance the performance of the Freqtrade open-source crypto trading bot, namely by developing a program for optimization of hyperparameters and parameter search space.

The proposed research will not only investigate another possible evolutionary-based approach for trading automation but also contribute to the Freqtrade project by providing tools that will enable traders to create and optimize custom trading strategies with improved performance and adaptability. Hereby, we aim to provide a more accessible and efficient trading environment for users, ultimately promoting more informed and profitable trading decisions.

In the course of this study, we will conduct comprehensive experiments to evaluate the effectiveness of the proposed approach. Our experimental setup will involve testing the performance of various predefined trading strategies using both the Freqtrade built-in optimizer and our nature-inspired computing-based optimizer. By comparing the results obtained from both optimizers, we aim to assess the improvement in performance and adaptability achieved by our proposed approach. The evaluation process will involve rigorous back-testing, statistical analysis, and comparison of key performance metrics, such as profitability, sharpe ratio, and drawdown, to ensure the robustness and effectiveness of our optimizer in enhancing trading strategy performance. This comparative analysis will provide valuable insights into the potential benefits of incorporating nature-inspired computing techniques, specifically genetic algorithms, into the optimization of automated trading strategies.

II. Related Work

Evolutionary computation-based approaches are not new for the field of trading. However, researchers are still constantly finding new techniques as well as ways to apply the existing ones more efficiently for stock trading optimization and automation. The main state-of-the-art trends are:

1) Hybrid models: Combining evolutionary algorithms with other machine learning and artificial intelligence techniques has become popular in the field. Hybrid models leverage the strengths of multiple techniques to achieve better performance and adaptability. For instance, researchers might use EAs to optimize the parameters of a neural network-based trading model [1], [7].
2) Genetic Programming: Genetic programming (GP) has been applied to evolve trading strategies and predict stock prices [2]–[4]. GP can generate complex and non-linear models that can capture intricate market dynamics, making it a suitable approach for trading strategy optimization.
3) Genetic Algorithms: Genetic algorithms (GA) have been widely used for optimizing trading strategies, rule discovery, and feature selection in the trading domain [5], [6], [8]. GAs can efficiently search large parameter spaces and identify optimal solutions,

making them an attractive choice for trading strategy optimization.

4) Coevolutionary algorithms: Coevolutionary algorithms model the complex interactions between different market participants, such as traders and investors. This approach can help identify more robust trading strategies that can adapt to various market conditions and competitive environments [9].
5) Memetic algorithms: Memetic algorithms combine evolutionary algorithms with local search methods to improve the convergence speed and overall performance of EAs in stock trading applications [4].
6) Multi-objective optimization: Researchers are employing multi-objective optimization techniques to simultaneously optimize multiple performance metrics, such as profit, risk, and drawdown. This approach leads to more balanced and robust trading strategies that cater to various investment objectives [10].

III. Methodology

Strategies in the Freqtrade framework are represented as Python classes with pre-defined parameters such as ROI (return on investment), stoploss (limit of loss in percentage to stop trading and withdraw funds), etc, and a set of trading rules for interaction with cryptocurrency exchange (Binance, in our case) written as a class functions. The variation and a possibility for optimizing lays in a user defined parameters such as CategoricalParameter, DecimalParameter, IntParameter, and BooleanParameter classes of freqtrade.strateg. These parameters customize trading rules and usually introduced in a strategy as a range of values to choose from for user.

Our proposed approach for optimizing automated trading strategies using genetic algorithms (GA) involves several key steps. In this section, we outline the methodology employed in our research, which includes strategy representation, initial population generation, fitness evaluation, genetic operators, and the overall optimization process.

1. Strategy Representation: Strategies in our approach are represented simultaneously as a Python class and a set of user defined parameters. Hyperparameters are genes, and a candidate in a population is represented as a set of parameters, that is how the strategies are treated in mutation and crossover functions, as well as generally for storing to optimize memory usage. Only at the stage of evaluating the candidate's fitness, a Python class representation is generated from a set of parameters by extending the base class with new values with help of script in 'strategy_text_generator.py'.

2. Initial Population Generation: To create an initial population of trading strategies, we generate a set of candidate strategies by randomly sampling hyperparameters from their respective search spaces. The 'generate_initial_population' function in the 'genetic_optimizer.py' module is responsible for creating this initial population.

3. Fitness Evaluation: We evaluate the fitness of each candidate strategy in the population by backtesting it on historical market data. An obtained profit is used as the loss function. The backtesting process is performed using the Freqtrade framework, and the fitness of a strategy is determined by its profitability. The 'evaluate_population' function in the 'genetic_optimizer.py' module handles the fitness evaluation of the entire population.

4. Genetic Operators: We employ two main genetic operators in our approach: mutation and crossover. The mutation operator introduces random changes to the hyperparameters of a candidate strategy, while the crossover operator combines hyperparameters from two parent strategies to create a new offspring strategy. The 'mutate_candidate' and 'crossover_candidates' functions in the 'genetic_optimizer.py' module implement these genetic operators.

5. Population Composition: Our GA-based optimization process involves iteratively evolving the population of candidate strategies over multiple generations. The first generation starts by creating a random population. At the beginning of each generation, classes representing strategies are created for evaluation. Then their profits on the backtest are evaluated. The top one-third of candidates are always retained. Mutations and crossover are produced in numbers equal to one-third of the population size, with the ability to mutate and reproduce remaining in all candidates and distributed evenly. The 'genetic_algorithm' function in the 'genetic_optimizer.py' module drives this optimization process.

By following this methodology, we aim to demonstrate the effectiveness of our GA-based approach in optimizing automated trading strategies and compare its performance to the built-in optimizer provided by the Freqtrade framework, which is based on the ExtraTreesRegressor from skikit-learn

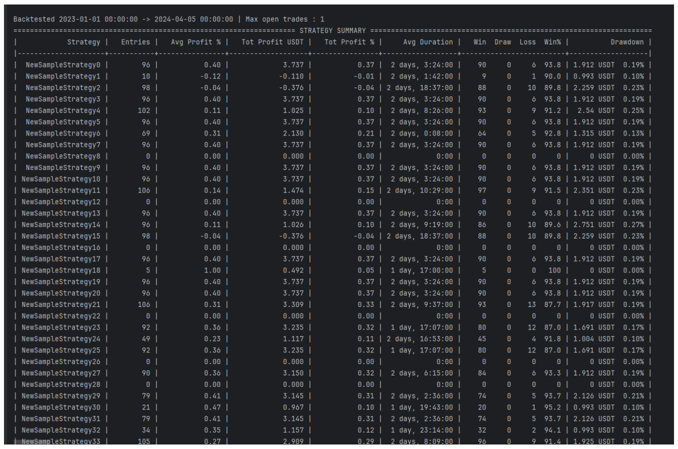
IV. GitHub link

Link to the GitHub repository

V. Experiments and Evaluation

Our GA optimizer is tested and evaluated primarily on three strategies: "SampleStrategy", "Diamond" and "Strategy005" taken from official strategy examples from Freqtrade GitHub [11] via backtesting functionality of the framework. Backtesting is a modeling technique for testing a predictive model or financial strategies on historical data. It is an attempt to simulate real behavior as closely as possible. When we talk about cryptocurrency trading, the main drawback is the lack of information about the movement between time frames. Despite the shortcomings, backtesting remains one of the most used and useful methods for evaluating trading strategies. This functionality includes showing a table describing performance of the strategies being tested on every iteration (epoch) of the GA (Fig. 1).

Then the fitness function value is parsed from "Tot Profit %" column, so that the fittest candidate of a population is defined as the individual having the maximal "Tot Profit %" value. In addition to every backtesting table, we consider the corresponding array of sorted fitness values and teh average value of fitness for the population, the fittest candidate's



```
Backtested 2023-01-01 00:00:00 -> 2024-04-05 00:00:00 | Max open trades : 1
=================================================================== STRATEGY SUMMARY ==================================================================
|            Strategy |   Entries |   Avg Profit % |   Tot Profit USDT |   Tot Profit % |      Avg Duration |   Win  Draw  Loss  Win% |          Drawdown |
|---------------------+-----------+----------------+-------------------+----------------+-------------------+-------------------------+-------------------|
|  NewSampleStrategy0 |        96 |           0.40 |             3.737 |           0.37 |  2 days, 3:24:00 |    90     0     6  93.8 | 1.912 USDT  0.19% |
|  NewSampleStrategy1 |        10 |          -0.12 |            -0.110 |          -0.01 |  2 days, 1:42:00 |     9     0     1  90.0 | 0.993 USDT  0.10% |
|  NewSampleStrategy2 |        98 |          -0.04 |            -0.376 |          -0.04 | 2 days, 18:37:00 |    88     0    10  89.8 | 2.259 USDT  0.23% |
|  NewSampleStrategy3 |        96 |           0.40 |             3.737 |           0.37 |  2 days, 3:24:00 |    90     0     6  93.8 | 1.912 USDT  0.19% |
|  NewSampleStrategy4 |       102 |           0.11 |             1.025 |           0.10 |  2 days, 8:26:00 |    93     0     9  91.2 |  2.54 USDT  0.25% |
|  NewSampleStrategy5 |        96 |           0.40 |             3.737 |           0.37 |  2 days, 3:24:00 |    90     0     6  93.8 | 1.912 USDT  0.19% |
|  NewSampleStrategy6 |        69 |           0.31 |             2.130 |           0.21 |  2 days, 0:08:00 |    64     0     5  92.8 | 1.315 USDT  0.13% |
|  NewSampleStrategy7 |        96 |           0.40 |             3.737 |           0.37 |  2 days, 3:24:00 |    90     0     6  93.8 | 1.912 USDT  0.19% |
|  NewSampleStrategy8 |         0 |           0.00 |             0.000 |           0.00 |             0:00 |     0     0     0     0 |      0 USDT  0.00% |
|  NewSampleStrategy9 |        96 |           0.40 |             3.737 |           0.37 |  2 days, 3:24:00 |    90     0     6  93.8 | 1.912 USDT  0.19% |
| NewSampleStrategy10 |        96 |           0.40 |             3.737 |           0.37 |  2 days, 3:24:00 |    90     0     6  93.8 | 1.912 USDT  0.19% |
| NewSampleStrategy11 |       106 |           0.14 |             1.474 |           0.15 | 2 days, 10:29:00 |    97     0     9  91.5 | 2.351 USDT  0.23% |
| NewSampleStrategy12 |         0 |           0.00 |             0.000 |           0.00 |             0:00 |     0     0     0     0 |      0 USDT  0.00% |
| NewSampleStrategy13 |        96 |           0.40 |             3.737 |           0.37 |  2 days, 3:24:00 |    90     0     6  93.8 | 1.912 USDT  0.19% |
| NewSampleStrategy14 |        96 |           0.11 |             1.026 |           0.10 |  2 days, 9:19:00 |    86     0    10  89.6 | 2.751 USDT  0.27% |
| NewSampleStrategy15 |        98 |          -0.04 |            -0.376 |          -0.04 | 2 days, 18:37:00 |    88     0    10  89.8 | 2.259 USDT  0.23% |
| NewSampleStrategy16 |         0 |           0.00 |             0.000 |           0.00 |             0:00 |     0     0     0     0 |      0 USDT  0.00% |
| NewSampleStrategy17 |        96 |           0.40 |             3.737 |           0.37 |  2 days, 3:24:00 |    90     0     6  93.8 | 1.912 USDT  0.19% |
| NewSampleStrategy18 |         5 |           1.00 |             0.492 |           0.05 |  1 day, 17:00:00 |     5     0     0   100 |      0 USDT  0.00% |
| NewSampleStrategy19 |        96 |           0.40 |             3.737 |           0.37 |  2 days, 3:24:00 |    90     0     6  93.8 | 1.912 USDT  0.19% |
| NewSampleStrategy20 |        96 |           0.40 |             3.737 |           0.37 |  2 days, 3:24:00 |    90     0     6  93.8 | 1.912 USDT  0.19% |
| NewSampleStrategy21 |       106 |           0.31 |             3.309 |           0.33 |  2 days, 9:37:00 |    93     0    13  87.7 | 1.917 USDT  0.19% |
| NewSampleStrategy22 |         0 |           0.00 |             0.000 |           0.00 |             0:00 |     0     0     0     0 |      0 USDT  0.00% |
| NewSampleStrategy23 |        92 |           0.36 |             3.235 |           0.32 |  1 day, 17:07:00 |    80     0    12  87.0 | 1.691 USDT  0.17% |
| NewSampleStrategy24 |        49 |           0.23 |             1.117 |           0.11 | 2 days, 16:53:00 |    45     0     4  91.8 | 1.004 USDT  0.10% |
| NewSampleStrategy25 |        92 |           0.36 |             3.235 |           0.32 |  1 day, 17:07:00 |    80     0    12  87.0 | 1.691 USDT  0.17% |
| NewSampleStrategy26 |         0 |           0.00 |             0.000 |           0.00 |             0:00 |     0     0     0     0 |      0 USDT  0.00% |
| NewSampleStrategy27 |        90 |           0.36 |             3.150 |           0.32 |  2 days, 6:15:00 |    84     0     6  93.3 | 1.912 USDT  0.19% |
| NewSampleStrategy28 |         0 |           0.00 |             0.000 |           0.00 |             0:00 |     0     0     0     0 |      0 USDT  0.00% |
| NewSampleStrategy29 |        79 |           0.41 |             3.145 |           0.31 |  2 days, 2:36:00 |    74     0     5  93.7 | 2.126 USDT  0.21% |
| NewSampleStrategy30 |        21 |           0.47 |             0.967 |           0.10 |  1 day, 19:43:00 |    20     0     1  95.2 | 0.993 USDT  0.10% |
| NewSampleStrategy31 |        79 |           0.41 |             3.145 |           0.31 |  2 days, 2:36:00 |    74     0     5  93.7 | 2.126 USDT  0.21% |
| NewSampleStrategy32 |        34 |           0.35 |             1.157 |           0.12 |  1 day, 23:14:00 |    32     0     2  94.1 | 0.993 USDT  0.10% |
| NewSampleStrategy33 |       105 |           0.27 |             2.909 |           0.29 |  2 days, 8:09:00 |    96     0     9  91.4 | 1.925 USDT  0.19% |
```

Fig. 1. Backtesting table example

parameter values and its fitness score. This data is being saved in the 'reports' directory as a json file named the same as a strategy optimized. After that, the stored data is used to get an objective function plot. The effectiveness of our algorithm is evaluated in accordance with the strategy suggested by the GA utilizing randomly selected initial parameter values. Subsequently, the algorithm is compared to the built-in Freqtrade hyperopt optimizer in terms of attained profit and computational complexity.

In our experiments, we compared the performance of three different predefined trading strategies without optimization, with the default Freqtrade hyperopt, and with our GA-based optimizer. The evaluation was conducted using backtesting on a market data from 2024-01-01 to 2024-04-05 (Table 1). This interval was chosen as a balance between the speed of the algorithm and the results obtained. For the same purpose, a population size of 30 candidates and a number of generations of 20 were used as parameters of the genetic algorithm.

An important note is that all of these results were impaired by being forced out of position at the end of the backtesting period with a loss of more than 5 percent on all strategies

TABLE I
OPTIMIZATION TECHNIQUES COMPARISON RESULTS

Strategy name	Best of initial population profit	Profit with GA	GA time	Profit with hyper-opt	Hyperopt time
Sample Strategy	0.113%	0.185%	17 min	0.184%	9 min
Diamond	0.00%	0.297%	23 min	0.34%	7 min
Strategy 005	0.133%	0.273%	16 min	0.268%	7 min

Our program is fully compatible with the Freqtrade platform and provides a convenient interface for users to interact with the bot (Fig. 2).

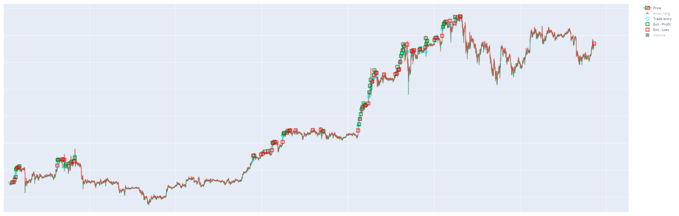
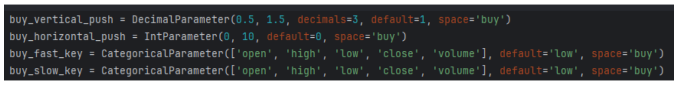


Fig. 2. Freqtrade: An overview of the actions performed by the bot for a strategy optimized and passed to it by our program ("Diamond" strategy)

To perform optimization, it parses strategy parameters from the files where the strategies are defined. It is designed to work seamlessly with the Freqtrade platform, requiring minimal user input. Users only need to specify the strategy to optimize and provide the corresponding strategy file. Our program supports the optimization of all parameters that implement CategoricalParameter, DecimalParameter, IntParameter, and BooleanParameter classes of freqtrade.strategy (Fig. 3).



```
buy_vertical_push = DecimalParameter(0.5, 1.5, decimals=3, default=1, space='buy')
buy_horizontal_push = IntParameter(0, 10, default=0, space='buy')
buy_fast_key = CategoricalParameter(['open', 'high', 'low', 'close', 'volume'], default='low', space='buy')
buy_slow_key = CategoricalParameter(['open', 'high', 'low', 'close', 'volume'], default='low', space='buy')
```

Fig. 3. Example of parameters to be optimized

Our GA-based optimizer determines the search space for parameters based not only on their predefined ranges but also on the number of parameters. This adaptive approach allows for a more comprehensive exploration of the parameter space. We also attempted to design a search space optimizer; however, we decided to postpone this feature due to time constraints, as testing the prototype indicated that it required more than 10 hours to achieve significant improvement.

During the optimization process, losses and candidate solutions are logged on every iteration as JSON files in the 'reports' folder for further analysis and plotting. This logging feature enables users to track the progress of the optimization process and analyze the performance of different candidate solutions.

VI. Analysis and Observations

Objective function plots for Sample Strategy (Fig. 4), Diamond Strategy (Fig. 5), and Strategy 005 (Fig. 6) strategies reveal that the genetic algorithm, due to its high randomness, proved to be an effective solution for selecting parameters in such a chaotic trading environment. We observed that even from the first generation, which is randomly generated population, the GA produced acceptable results, which continued to improve over time. This indicates the high applicability of the randomness of this method. The growth rate of these results executed on a single processor thread was comparable with ones of Freqtrade's built-in ExtraTreesRegressor hyperparameter optimizer running on 12 threads. Even though there is a time difference of about 2 times, the reason for this is the lack of multithreading.

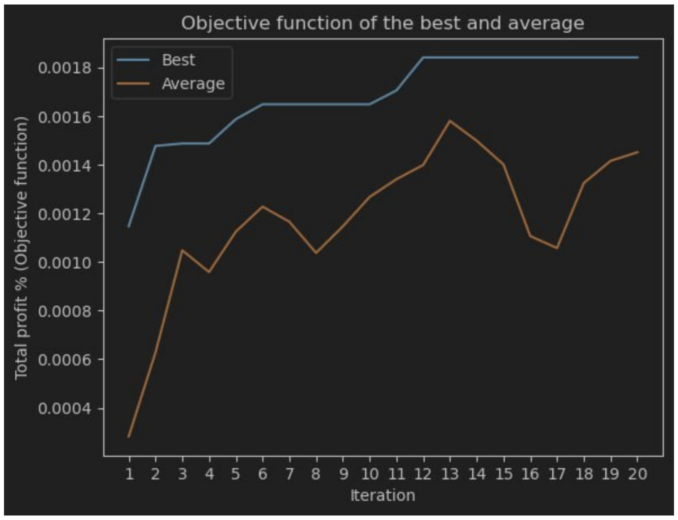


Fig. 4. GA objective function for "SampleStrategy" strategy

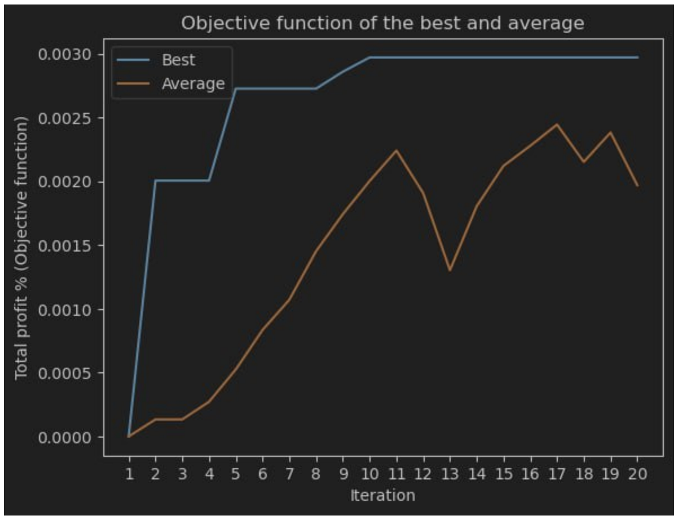


Fig. 5. GA objective function for "Diamond" strategy

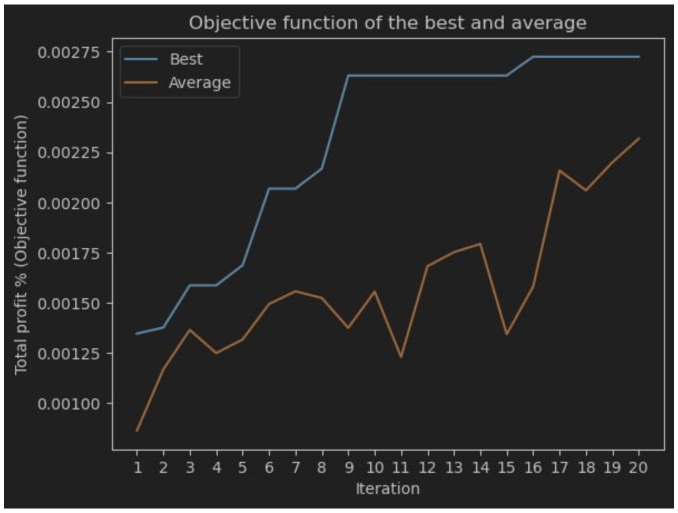


Fig. 6. GA objective function for "Strategy005" strategy

It is important to acknowledge that algorithmic trading strategies relying solely on technical indicators for decision-making are not expected to yield exceptional results. The experiments conducted on these strategies serve as a demonstration of the potential of optimization using a genetic algorithm. This approach can be particularly valuable when applied to more complex algorithms that incorporate news, sentiment analysis, and time series analysis using neural networks.

VII. Conclusion

In conclusion, our GA-based optimizer demonstrated promising results in enhancing the performance of predefined trading strategies. As future work, we plan to further improve the user experience by adding a UI or CLI for our program, enabling the optimization of additional parameters, and incorporating more loss functions. It is also an important step to implement multithreading, to speed up the process. These enhancements will contribute to the development of more efficient and adaptive trading strategies, ultimately promoting more informed and profitable trading decisions.

References

[1] C. Evans, K. Pappas, and F. Xhafa, "Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation," Math. Comput. Model., vol. 58, pp. 1249–1266 , September 2013.

[2] M. A. Kaboudan, "Genetic Programming Prediction of Stock Prices," Comput. Econ., vol. 16, pp. 207–236, December 2000.

[3] M. Dempster and C. Jones, "A real-time adaptive trading system using genetic programming," Quantit. Fin., vol. 1, pp. 397–413, April 2001.

[4] S. Y. Yang, S. Y. Mo, A. Liu, and A. Kirilenko, "Genetic programming optimization for a sentiment feedback strength based trading strategy," Neurocomp., vol. 264, pp. 29–41 , November 2017.

[5] P. Kroha and M. Friedrich, "Comparison of Genetic Algorithms for Trading Strategies," SOFSEM 2014: Theor. Pract. Comput. Sci., pp. 383–394 , 2014.

[6] Y. Kim, W. Ahn, K.J. Oh, and D. Enke, "An intelligent hybrid trading system for discovering trading rules for the futures market using rough sets and genetic algorithms," Appl. Soft Comput., vol. 55, pp. 127-140, June 2017.

[7] C. Evans, K. Pappas, and F. Xhafa, "Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation", Math. Comput. Model., vol. 58, pp. 1249-1266, September 2013.

[8] J. Straßburg, C. Gonzàlez-Martel, V. Alexandrov, "Parallel genetic algorithms for stock market trading rules," Procedia Comp. Sci., vol. 9, pp. 1306-1313, 2012

[9] K. Adamu and S. Phelps, "Coevolutionary Grammatical Evolution for Building Trading Algorithms," Electr. Eng. Appl. Comput., vol. 90, pp. 311-322, January 2011

[10] G. Zandi, R. Torabi, M.A. Mohammad, and L. Jia, "Research on stock portfolio based on time series prediction and multi-objective optimization," Advanc. Math. Sci. J., vol. 3, pp. 1131-1136, March 2021

[11] freqtrade/freqtrade-strategies repository containing free buy/sell strategies for Freqtrade. Link: https://github.com/freqtrade/freqtrade-strategies