Multi-Objective Genetic Algorithm for Financial Indicator Optimization in NASDAQ Trading

Francesco di Massimo Márton László Hévizi
Bio-Inspired Artificial Intelligence – Final Project
University of Trento, Academic Year 2024/2025
https://github.com/fradimassimo/FinGA

***Abstract*—This project presents the development of a multi-objective GA (Genetic Algorithm) designed to maximize profits while minimizing the number of transactions in the NASDAQ market. The study is motivated by the increasing interest in quantitative finance, where the complexity of financial systems has highlighted the potential of evolutionary algorithms as decision-making tools. The proposed approach encodes both the parametrization of financial indicators and the confidence assigned to them within the chromosome representation, allowing the evolutionary process to optimize trading strategies. The algorithm was implemented in Python using the DEAP [1] library and tested on 20 NASDAQ-listed companies over the period 2005–2015, a time frame that includes both bull and bear markets, such as the global financial crisis and subsequent recovery. Stock data were retrieved via the Yahoo Finance API. The algorithm demonstrated significant improvement over generations. The findings suggest that multi-objective genetic algorithms, implemented with Python and DEAP, are a promising tool for financial indicator selection and automated trading strategy design.**

***Index Terms*—Genetic Algorithm, Multi-Objective Optimization, Financial Indicators, Stock Trading, NASDAQ, DEAP, Yahoo Finance**

I. Introduction

In recent years, quantitative approaches to finance have gained increasing attention due to the complexity and volatility of modern financial markets. Traditional trading strategies often struggle to adapt to rapid market changes, motivating the use of computational intelligence methods that are capable of learning, adapting, and optimizing under uncertainty. Among these methods, evolutionary algorithms have emerged as a promising tool for tackling the challenges of financial decision-making.

Genetic Algorithms (GAs), in particular, provide a robust framework for optimization problems where multiple conflicting objectives must be considered. In financial trading, one can consider a trade-off between maximizing profit while minimizing transaction costs, as frequent trading can reduce overall gains despite generating short-term profit opportunities. Multi-objective optimization therefore offers a natural approach to designing more balanced trading strategies.

We developed a multi-objective GA for the parametrization and weighting of financial indicators that generate buy and sell signals for NASDAQ-listed stocks. The chromosome representation encodes both the chosen indicators parameters and the confidence assigned to them, enabling the evolutionary process to refine strategies over time. The algorithm is evaluated on 20 stocks over the period 2005–2015, which includes both bull and bear market phases such as the global financial crisis and following rebounds. This allows us to assess the robustness of the approach across different market conditions.

The remainder of this report is organized as follows: Section II reviews relevant literature and background on evolutionary computation in finance. Section III describes the methodology and the genetic algorithm design. Section IV presents the experimental setup and dataset. Section V discusses the results, and Section VI concludes the report with final remarks and possible future work directions.

II. Related works

Genetic Algorithms (GA) [2] have been extensively applied to financial optimization problems since their introduction by Holland [3], [4]. In the financial domain, researchers have explored various applications of GA, ranging from technical trading rule learning to portfolio optimization.

The main source of inspiration for our work comes from Fu et al. [5]. They explore the application of GA in portfolio management and technical analysis. In particular, they propose using genetic algorithms to optimize the parameters of various technical indicators and to determine the optimal weighting of assets in a portfolio. Both the traditional genetic algorithm and the hierarchical genetic algorithm are compared, evaluating their performance in different economic scenarios. The results show that genetic algorithms can provide significant improvements in portfolio optimization and in the adaptability of technical analysis.

In contrast, our work focuses solely on the optimization of technical indicators, without delving into portfolio allocation, testing several evolutionary strategies: the eaSimple, (μ, λ) and $(\mu + \lambda)$ evolutionary algorithms together with the NSGA-II algorithm as selection logic.

1) eaSimple Algorithm: The eaSimple evolutionary algorithm [6] takes 5 arguments, a population, a toolbox, a probability of mating each individual at each generation $cxpb$, a probability of mutating each individual at each generation $mutpb$ and a number of generations to accomplish $ngen$. The toolbox is containing the methods for selection, evaluation mutation and mating strategies.

2) (μ, λ) Algorithm: In the (μ, λ) selection scheme, μ parents generate λ offspring at each generation. Only the best

Algorithm 1 Generic Evolutionary Algorithm

Input: initial population, number of generations $ngen$, crossover probability $cxpb$, mutation probability $mutpb$
Output: final evolved population
evaluate(population)
for $g \leftarrow 1$ to $ngen$ **do**
    population $\leftarrow$ select(population, len(population))
    offspring $\leftarrow$ varAnd(population, toolbox, $cxpb$, $mutpb$)
    evaluate(offspring)
    population $\leftarrow$ offspring

μ individuals among the offspring are selected to become parents of the next generation. This approach ensures that each new generation consists solely of offspring, without carrying over any of the previous generation's individuals, thus promoting exploration of new areas in the search space.

3) $(\mu + \lambda)$ Algorithm: In the $(\mu + \lambda)$ selection scheme, μ parents generate λ offspring as well. However, the next generation is selected from the combined pool of parents and offspring. This allows the best individuals from the previous generation to survive into the next, promoting exploitation of good solutions while still allowing the search to explore new possibilities. The implementation of both previous variant follows the `DEAP` library [1].

4) $(NSGA-II)$ Selection: The algorithm is implemented based on [7]. The algorithm follows the general outline of a genetic algorithm with a modified mating and survival selection. In NSGA-II, first, individuals are selected front-wise. By doing so, there will be the situation where a front needs to be split because not all individuals are allowed to survive. In this splitting front, solutions are selected based on crowding distance. The crowding distance is the Manhattan Distance in the objective space. However, the extreme points are desired to be kept every generation and, therefore, get assigned a crowding distance of infinity. Furthermore, to increase some selection pressure, $NSGA - II$ uses a binary tournament mating selection. Each individual is first compared by rank and then crowding distance.

III. Methodology

This section describes the methodological framework adopted in this study. The goal is to design a multi-objective genetic algorithm (GA) that can optimize trading strategies. The methodology consists of four main components: the choice of financial indicators, the design of the genetic algorithm, the chromosome encoding scheme, and the fitness evaluation process.

A. Indicators

Technical indicators are widely used in quantitative finance to capture market trends and price momentum. These indicators can help the trader decide when to buy or to sell. They generate buy and sell signals based on their parametrization and stock information. In this work, a set of commonly used indicators is employed, that are also easy to calculate, including William's 100 R (W100r), Momentum (MO) and Moving Average Convergence Divergence crossover (MACD-crossover).

1) William's % R: William's %R is a technical indicator created by Larry Williams, that aims to tell if a stock is overbought of oversold. The indicator shows the current price level as it is related to the highest price in the previous period.

$$\%R = -100 \times \frac{high_{Ndays} - close_{today}}{high_{Ndays} - low_{Ndays}}$$

When the Williams ratio is close to -100 it means that the current price is close to the lowest price of the last period indicating the stock is oversold, when it is close to 0, the current price is closer to highest price of the last period indicating the stock is overbought.

Buy and Sell signals can be generated based on the value of the indicator and the use of thresholds. If the stock is overbought, $\%R > -20$, the indicator is telling the trader to sell, if oversold, $\%R < -80$, to buy.

Changing the time frame impacts the sensitivity of the indicator, but also increases the error.

By finding the best values for the time frame and buy and sell thresholds, the indicator can be finetuned.

2) Momentum: The momentum oscillator indicates market strength by measuring the rate of change in prices relative to past levels. For an x-day momentum, it is defined as:

$$M = V - V_x,$$

where V is the most recent closing price and V_x is the closing price x days earlier.

A positive momentum value suggests that the most recent price is higher than that of x days ago, whereas a negative value indicates the opposite. In practice, common choices for x are 10 or 14 days. Shorter periods make the oscillator more responsive but also noisier, while longer periods smooth the signal at the cost of slower reactions.

One of the key features of the momentum oscillator is that it can move ahead of price trends, often providing early signals of trend reversals. In practice, a crossing above the zero line is frequently interpreted as a buy signal, while a crossing below zero is interpreted as a sell signal.

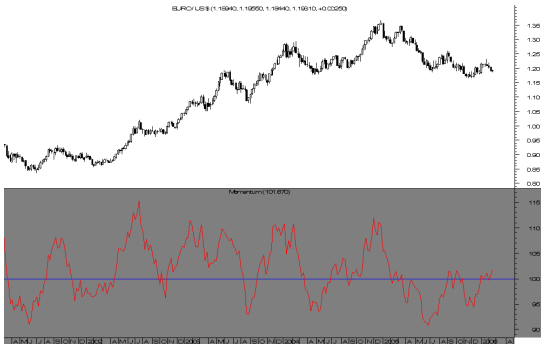

Fig. 1: Weekly chart of the Euro/Dollar exchange rate with the 14-period Momentum oscillator. Source: FeRR@ - Opera propria, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=125374551

3) MACD crossover: The moving average convergence/divergence indicator (MACD) measures the strength and the direction of stock prices. Created by Gerald Appel in the 1970s, the naming comes from the fact that the indicator shows the distancing and closing of moving averages. At the core it is using a short and long period moving average, their difference gives the MACD line:

$$MACD\ line = EMA_{short\_day} - EMA_{long\_day}$$

When the short period is above the long period average it indicates an upward momentum, otherwise a downward momentum. The greater the difference the bigger the strength.

The signal line is the exponential moving average of the MACD, by averaging the MACD line itself, it can reduce noise and false signals.

$$Signal\ line = EMA_{signal\_days}(MACD\ line)$$

The difference of the MACD line and the Signal line is referred as the histogram. When a crossover occurs and the MACD line goes above the signal line a buy signal is generated, and when it crosses it from above a sell signal is generated.

By carefully choosing the short, long and signal periods, the MACD crossover can be finetuned.

Fig. 2: Example of historical stock price data (top half) with the typical presentation of a MACD(12,26,9) indicator (bottom half). The blue line is the MACD series proper, the difference between the 12-day and 26-day EMAs of the price. The red line is the average or signal series, a 9-day EMA of the MACD series. The bar graph shows the divergence series, the difference of those two lines. Source: By Tradermatt at the English-language Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17936449

B. Genetic representation

Each individual is characterized by it's genetic code. The genes encode the parametrization of the indicators, and the weights for combining them. It is implemented as a python list with the DEAP framework.

```
The genetic code: [macd_short_days,
macd_long_days, macd_signal_days,
w100r_days, w100r_buy_threshold,
w100r_sell_threshold, momentum_days,
macd_weight, w100r_weight]
```

Every gene that includes the word *days* is an integer between 1 and 100. The thresholds for the $W100R$ are decimal numbers between -100 and 0 and the weights are between 0 and 1. The parameters and the stock prices are passed to each indicator, which return a pandas Series containing a signal for each day. The signal is either "BUY", "SELL" or "STAY". These signals are than combined to produce one final Series to make trading decisions with. The weight of the momentum indicator is calculated: $1 - macd\_weight - w100r\_weight$. The combined signal is created based on the following rules:

- if the weighted sum of the buy signals is greater than 0.5 for the day, a "BUY" signal is chosen
- if the weighted sum of the sell signal is greater than 0.5 for the day, a "SELL signal is chosen
- If neither signal has a majority the default "STAY" signal is chosen

In our genetic algorithm, not all individuals represent valid or meaningful solutions. To ensure that each candidate configuration is feasible, we implement a feasibility check. Specifically, as previously discussed, an individual is represented as a list of parameters for different indicators and their corresponding weights. The feasibility constraints are:

- $macd\_short\_days$ must not exceed $macd\_long\_days$.
- $w100r\_buy\_threshold$ must be lower than the sell threshold.
- $macd\_weight$ + $w100r\_weeight$ must be lower than 1, to maintain a proper contribution balance between indicators.
- if the individual made no trades, it is considered invalid.

An individual that violates any of these constraints is considered infeasible and is penalized in the selection process. This feasibility check guarantees that the GA explores only meaningful and implementable solutions.

C. Fitness Evaluation

The fitness of each individual is computed based on the historical stock data retrieved from the Yahoo Finance API. Given a chromosome, trading signals are generated according to the selected indicators and their confidence weights. The resulting trades are simulated over the historical time period, and two fitness values are calculated:

1) Total profit achieved by the strategy. (goal: maximize)
2) Number of trades executed. (goal: minimize)

These two values are treated as separate objectives in a Pareto optimization setting. The GA then evolves strategies that represent different trade-offs between profitability and trading frequency, allowing the identification of efficient strategies.

For each individual, the same stocks over the same period are simulated. The individual progresses through the days, and is able to buy and sell the currently selected stock. Each buy has to be followed by a sell, before another trade can be started. At the end of the given period the profits are added up. To take the stock price out of the equation we use ROI(Return Of Investment) $\frac{sell\_price - buy\_price}{buy\_price}$. This is repeated for all stocks and at the end the final profit score is given by the sum of profits for each company. The number of transactions is calculated by adding up all trades made by the strategy for all stocks in the given time period.

If an individual is infeasible it is penalized by setting the profit to $-infinite$ and the number of trades to $+infinite$.

D. Gene mutation

The mutation strategy we employ is different for the genes that encode day parameters from those that encode threshold or weight parameters. For each gene there is an individual mutation probability which affects if that specific gene will be modified. For day parameters a uniformly generated random integer between -20 and 20 will be added to the current value. The mutation is clamped, so the day always stays between 1 and 100. A similar strategy is used for the threshold, but a gaussian floating point number with 0 mean and a standard deviation of 20 is generated instead. The value is also clamped, but between $-100, 0$. The weight parameters share logic with the threshold, but with a standard deviation of 0.5 and clamped between 0 and 1.

IV. Experimental Setup

The GA follows the standard evolutionary cycle of initialization, evaluation, selection, crossover, mutation, and replacement. The DEAP Python library is used to implement the evolutionary operators and manage the multi-objective optimization process. The stock price of 20 companies was downloaded from 2005.01.01 until 2015.01.0.1 (The list can be found on github). Two experiments are ran, one using the (μ, λ) evolution strategy and one using (μ + λ). The configuration for both is identical:

- starting population is set to 100
- λ is set to 100
- μ is set to 50
- uniform crossover with probability of 0.5
- mutation probability of 0.3
- independent gene mutation probability of 0.8
- the algorithms are run for 50 generations

V. Results

In this section, we present the outcomes of our genetic algorithm experiments. The analysis focuses on the evolution of the fitness values across generations as well as the resulting Pareto front representing the trade-off between the two objectives: maximizing profit and minimizing the number of transactions.

A. (μ, λ) Strategy

1) Pareto Front: Each point represents an individual in the final pareto-hallofame, with the y-axis representing the number of transactions and the x-axis representing the profit. The Pareto front illustrates the trade-off between the two objectives, highlighting solutions that offer different balances between profitability and trading frequency. In our experiments the front obtained at the end of the evolutionary process, see Figure 3 exhibits three distinct patterns. In the initial region (bottom-left), it appears relatively flat, indicating that profits can increase while maintaining a low and nearly constant trading activity, at least up to a certain point. A noticeable gap then emerges in the intermediate region, where no solutions were found. This absence may reflect either a limitation in the algorithm's exploration capabilities or the actual lack of efficient configurations in that portion of the search space. Beyond this gap, the Pareto front resumes with a much steeper trend, resembling exponential growth: in this regime, profit increases are attainable, but only at the cost of a significantly higher number of transactions. This behavior highlights a clear trade-off between profitability and operational complexity.

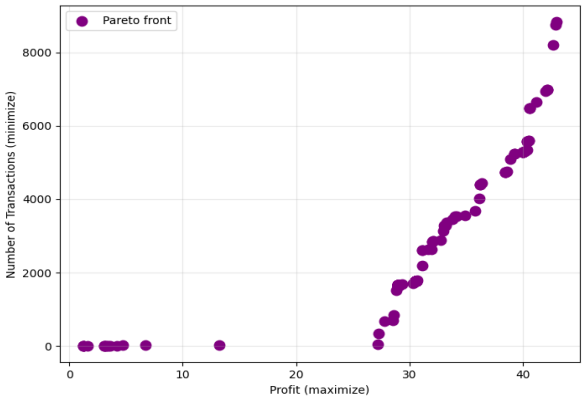

Fig. 3: Pareto front showing the trade-off between profit and number of transactions. (μ, λ)

2) Fitness Evolution: This visualization provides insight into the algorithm's progression toward optimal solutions and the population's improvement with respect to both objectives over time. Figure 4 illustrates the evolution of the two fitness values over generations. The maximum profit rises rapidly during the initial generations, but its growth slows around generation 10. In contrast, the population's average profit continues to increase steadily until approximately generation 40. The transaction plot, however, exhibits a less clear trend, with values fluctuating throughout the evolutionary process.

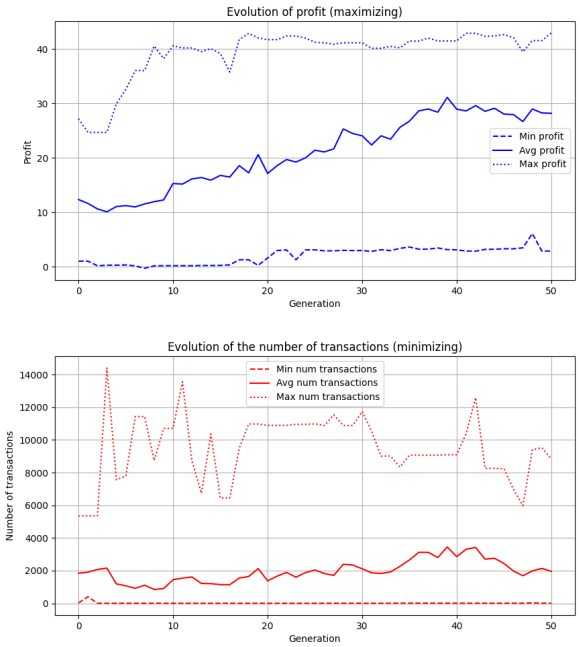

Fig. 4: Evolution of profit and number of transactions over generations. (μ, λ)

B. (μ + λ) Strategy

1) Pareto Front: The Pareto front show a similar trend to the previous algorithm, where initially the profits rapidly increase up to a point, after which the number of transactions start to increase, and the rate of the growth for profits slows down. Interestingly there are some solutions that are able to achieve very high profits with a low number of transactions, dominating every other solution in the area, creating a gap in the front.

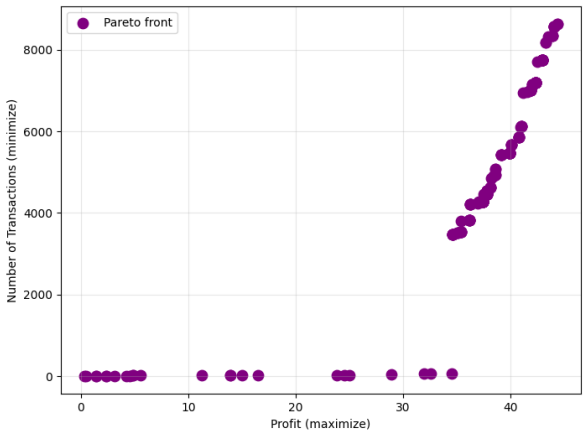

Fig. 5: Pareto front showing the trade-off between profit and number of transactions. (μ + λ)

2) Fitness Evolution: Similarly to the previous strategy the maximum profit increased quickly at first, but after generation 10 the progress slowed down. The average transactions showed a slight increasing trend.

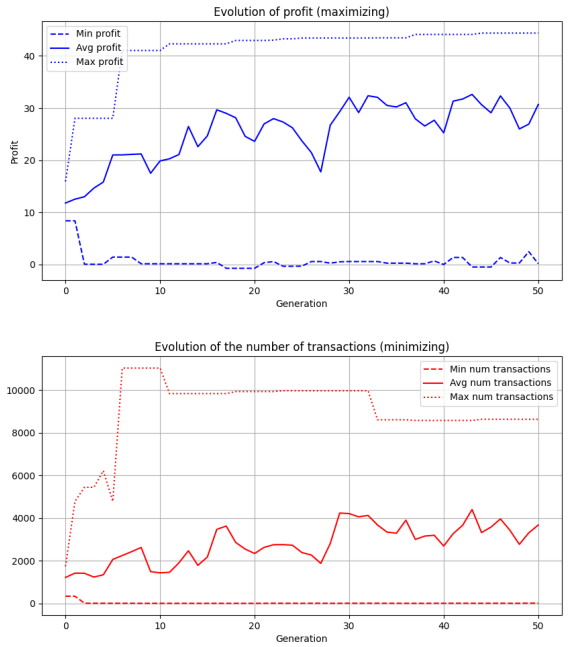

Fig. 6: Evolution of profit and number of transactions over generations. (μ + λ)

C. Analysis of the best solutions

Most solutions that achieved the highest profits had the MACD indicator's weight set to 0, which means that it didn't perform well. The other two indicators weights were usually evenly divided.

VI. Conclusions

In this study, a genetic algorithm was employed to optimize technical indicators for trading strategies, framed as a multi-objective problem aiming to maximize profit and minimize transaction frequency. The results indicate that the algorithm effectively enhanced profitability; however, controlling the number of transactions proved challenging. These findings suggest that incorporating explicit constraints on transaction volume or budget could further improve the practical applicability and balance of the optimized strategies. Both algorithms seemed to perform the same in our scenario. We also tested the eaSimple algorithm for a limited number of generations, but the results were not competitive with the other two strategies.

Future work may explore:

- Incorporating additional technical indicators or market features.
- Testing alternative selection mechanisms and genetic operators.
- Extending the analysis to multiple assets or longer historical periods.

Overall, the study confirms the effectiveness of evolutionary algorithms in handling multi-objective optimization problems in portfolio management and technical analysis.

REFERENCES

[1] F. Fortin, F.-A. D. Rainville, M. Gardner, M. Parizeau, and C. Gagné, *DEAP: Distributed Evolutionary Algorithms in Python*, 2012, [Online; accessed 28-Aug-2025]. [Online]. Available: https://github.com/DEAP/deap

[2] T. Back and H. P. Schwefel, "Evolutionary computation: An overview," in *Proc. IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 1996, pp. 20–29.

[3] J. H. Holland, "Outline for a logical theory of adaptive systems," *Journal of the ACM (JACM)*, vol. 9, no. 3, pp. 297–314, 1962.

[4] ——, *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press, 1975.

[5] T.-C. Fu, C.-P. Chung, and F.-L. Chung, "Adopting genetic algorithms for technical analysis and portfolio management," *Computers Mathematics with Applications*, vol. 66, no. 10, pp. 1743–1757, 2013.

[6] T. Back, D. B. Fogel, and Z. Michalewicz, *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing, 2000.

[7] K. Deb, A. Pratab, S. Agarwal, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.