A Genetic Programming Approach to Profitable Bitcoin Trading

Luca Claus
luca.claus@unitn.studenti.it - 257825
Course of Bio-Inspired Artificial Intelligence
University of Trento
2024-2025

***Abstract*—This project explores the use of Genetic Programming (GP), a subset of Evolutionary Algorithms (EA), to develop a novel, profitable trading strategy for Bitcoin. As a highly volatile asset, Bitcoin provides a unique opportunity to evaluate the robustness of algorithmic trading strategies. The GP algorithm evolves mathematical expressions that model optimal trading behaviors by combining a range of technical indicators, including the Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Bollinger Bands, and others. The study involves training the GP model on historical Bitcoin price data to maximize profitability. Backtesting on distinct datasets from different time periods ensures the strategy's generalizability and adaptability to varying market conditions. Results are analyzed in terms of return of profit and consistency, providing insights into the viability of GP-based trading systems in dynamic financial markets.**

I. Introduction

THE rapid growth of cryptocurrency markets, spearheaded by Bitcoin, has created a fertile ground for developing advanced algorithmic trading strategies. Unlike traditional assets, Bitcoin's 24/7 trading, high volatility, and lack of centralized regulation make it an attractive yet challenging target for quantitative analysis and strategy development. Among the various approaches in financial modeling, Genetic Programming (GP) stands out as a powerful tool for identifying nonlinear patterns and crafting adaptive trading strategies.

Genetic Programming, a class of Evolutionary Algorithms (EA), mimics the principles of natural selection to evolve solutions to complex problems. In the context of trading, GP iteratively generates and evaluates mathematical expressions that represent trading rules or strategies. These expressions are composed of terminal variables, such as price data and technical indicators, and functional operators, such as arithmetic or logical functions. The goal is to optimize (maximize) a fitness function.

This project focuses on leveraging GP to create a robust trading strategy for Bitcoin. The strategy development pipeline includes data preprocessing, feature extraction using technical indicators, and fitness evaluation through historical backtesting. The GP algorithm evolves a population of candidate strategies across generations, utilizing operations like crossover, mutation, and selection to refine their performance. Special attention is given to preventing overfitting by employing separate datasets for training.

The outcomes of this study aim to exploit the efficacy of GP in financial markets and its potential to outperform traditional trading methodologies. Furthermore, it highlights the challenges of applying EA algorithms to highly volatile and unpredictable assets like Bitcoin. The strategy is finally tested on different unknown environments to demostrate the consistency and adaptability on different market trends.

II. Methodologies

In this project, all the code is implemented in Python. The GP framework is based on DEAP (Distributed Evolutionary Algorithms in Python), a powerful and flexible library for evolutionary computation. For mathematical operations and data manipulation, we rely on Pandas and NumPy, which provide efficient and easy-to-use tools for handling financial data and computing technical indicators.

A. Data collection and preprocessing

To train the genetic algorithm, we collected historical price data from Binance [1], one of the largest cryptocurrency exchanges. The data is gathered in four different time periods, each with a 1-hour interval, representing various market conditions. Using a 1-hour interval allows us to capture short-term price movements and trends, making the data more suitable for intraday trading strategies, but not only. This granularity is ideal for capturing more frequent market fluctuations while still smoothing out the noise that may occur at lower timeframes (such as minutes). By analyzing data with a 1-hour interval, the genetic algorithm can learn to make decisions based on a balance between short-term market signals and overall market trends. This approach ensures that the resulting strategies can be highly responsive while still maintaining a broader market perspective. The four training periods each last approximately 6 months, covering a total span of 2 years. They are composed of:

- **Upward Trend:** A market that shows an increase in price, with a percentage difference between the start and end of the period around 77% (01-12-2023/01-06-2024).
- **Downward Trend**: A market where the price decreases, with a percentage difference of around -55% (01-11-2021/01-06-2022).

[1] https://www.binance.com

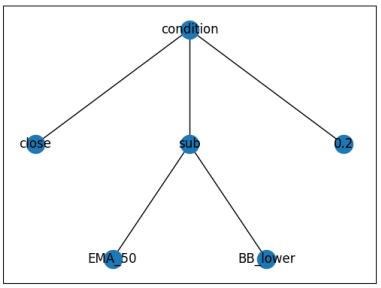

Fig. 1. Example of Individual

TABLE I
TERMINAL AND FUNCTION SETS FOR GENETIC PROGRAMMING.

Set	Description
Terminal Set	**Price Data:** open, close, high, low and volume. **Technical Indicators:** RSI, MACD, EMA_7, EMA_20, EMA_50, BB_upper, BB_lower and ATR. **Constants:** Fixed numerical values (0.2, 0.5, 5, 20...) and random [-1, 1].
Function Set	**Arithmetic Operators**: +, -, *, /. **Logical Functions:** $<$, $>$, AND, OR, NOT. **Conditional Functions:** if-else statements. **Statistical Functions:** min, max, mean and log.

- **Sideways Trend**: A market that stays within a narrow range, with price fluctuations of ±15% (01-07-2022/01-02-2023).
- **All Market Conditions**: A diverse collection that includes various trend types, helping the model learn to identify patterns across different market behaviors (01-01-2021/01-06-2021).

Once the data is collected, we proceed with preprocessing to ensure it is ready for modeling. This includes:

- Cleaning the data: Handling missing values and any potential outliers.
- Feature Engineering: Calculating a wide array of technical indicators to provide meaningful input features for the genetic programming model. These indicators include Moving Averages (EMA), Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Bollinger Bands, and others, which help in capturing important market signals and trends.

B. Design of the GP Model

The heart of the project is the Genetic Programming (GP) model, which aims to evolve a trading strategy based on the historical data and indicators. The GP model is designed to perform the following key tasks:

1) Design of Individuals: Each individual represents a potential trading strategy in the form of a program tree, as the example shown in Figure 1. The tree consists of:

- Functions: These are mathematical or logical operations used to combine and process the indicators and price data.
- Terminals: These are the inputs to the functions, which typically include the price data and the calculated indicators. Terminals may also include constants or random values used in the strategy.

The function and the terminal sets are shown in Table I.

2) Population Initialization and Technique: This step consists of:

- Population Size: 100 individuals are generated to start the evolution process.
- Initialization Method: We use a half-and-half method to initialize the population. This technique generates individuals using both full trees and grow trees, providing diversity in the initial population.
- Number of generations: The number of generations chosen for the training process is 20.

3) Fitness evaluation: Each individual in the population is evaluated based on its fitness, which reflects the performance of the strategy. The performance is based on the calculation of the profit generated by *buy*, *sell*, or *hold* actions. These signals are determined by executing the individual and interpreting the result using the following function:

$$S(x) = \begin{cases} \text{Buy} & \text{if } x > 0.5 \\ \text{Sell} & \text{if } x < -0.5 \\ \text{Hold} & \text{otherwise} \end{cases} \quad (1)$$

These signals are used to perform the trades and to realize profit/loss based on price data.

The fitness function incorporates several performance metrics, including total profit, the percentage of positive trades, and the number of trades executed. These metrics together provide a comprehensive evaluation of how effectively the strategy can operate and adapt to market conditions.

Designing an effective fitness function is a challenging task, as focusing solely on total profit can lead to overfitting. In such cases, the model might exploit few large positive trades and lots of small negative trades, while failing to adapt effectively to new or unseen market conditions. To mitigate this issue, we have integrated the percentage of positive trades.

Furthermore, to encourage robustness in the strategies, we introduce in the function the number of trades. This incentivizes diversification in trading actions and prevents the model from relying on a single trade for success. This dual approach percentages of positive trades and rewarding multiple trades ensures that the strategies developed are both profitable and adaptable, with a focus on consistent performance across a range of market scenarios.

In particular, we define the follow symbols:

- $F(x)$: The fitness function, which takes an individual as input and returns its fitness score.
- tot_pro: the total profit of the strategy
- num_tra: the total number of trades
- pos_pct: number of positives trades on total number $(number\_of\_positive\_trades)/num\_tra$

The fitness function to be maximazed is the following:

$$F(tot\_pro, pos\_pct, num\_tra) = \exp^{(tot\_pro/\alpha)} \cdot pos\_pct \cdot num\_tra$$

where α is a parameter used to scale down the total profit in order to prevent excessively large values. In this function, the exponential term, raised to the power of the total profit, gives greater weight to higher profits. The positive trade fraction is included to reward strategies with a higher proportion of profitable trades, while the total number of trades is also factored in to encourage the strategy to maximize trading activity. The combination of these three parameters ensures that the strategy strives to maximize profits, the percentage of successful trades, and the total number of trades.

The fitness function is calculated for each of the four different training periods. To obtain the final fitness value, the individual fitness values from all the periods are summed together. This cumulative approach ensures that the strategy is evaluated across multiple timeframes, allowing it to demonstrate its overall effectiveness and robustness.

4) Evolutionary process: The evolutionary algorithm iteratively improves the population by applying genetic operators such as selection, crossover, and mutation:

- **Selection:** Individuals are selected based on their fitness, where fitter individuals have a higher probability of being chosen for reproduction.
- **Hall of Fame:** The Hall of Fame is used to maintain a record of the best solutions found during the run of the genetic algorithm.
- **Crossover (Recombination):** Two parent individuals are selected, and their genetic material (trees) is combined to create offspring, producing new and potentially more effective strategies.
- **Mutation:** Random changes are made to an individual's strategy (e.g., swapping functions or terminals), which allows the algorithm to explore new possibilities and avoid local optima.

The parameters for the genetic algorithm were carefully selected to balance exploration and exploitation of the solution space, see Table II. Tournament selection with a size of 3 was chosen to ensure competitive pressure while maintaining diversity in the population. One-point crossover was selected with a probability of 0.7, enabling efficient exploration of the search space by combining traits of parent individuals. A uniform mutation with a probability of 0.05 was applied to introduce variability and prevent premature convergence. Finally, a Hall of Fame of size 3 was incorporated to preserve the best individuals throughout the evolutionary process, ensuring that the top solutions are not lost during generations.

III. Results

In this section, we will explore the strategy identified bu the GP algorithm and analyze the performance of the strategy on new environments.

A. Best Strategy identified

After the training phase of the model, the best individual/strategy identified by the algorithm is shown in Figure 2.

TABLE II
GENETIC ALGORITHM PARAMETERS

Parameter	Value
Selection Method	Tournament Selection
Tournament Size	3
Crossover Method	One-Point Crossover
Crossover Probability	0.7
Mutation Method	Uniform Mutation
Mutation Probability	0.05
Hall of Fame Size	3

TABLE III
GENETIC ALGORITHM PARAMETERS

Parameter	Value
Test Period	01-01-17 / 31-12-18
Market Price (%)	-9 %
Strategy Profit (%)	3328 %
Number of intervals	11880
Number of trades	1238
Trades pro day	1.7 (circa)

TABLE IV
GENETIC ALGORITHM PARAMETERS

Parameter	Value
Test Period	01-01-25 / 15-01-25
Market Price (%)	0 %
Strategy Profit (%)	2.8 %
Number of intervals	288
Number of trades	30
Trades pro day	2 (circa)

The figure illustrates the combination of terminal and function nodes that make up the optimal strategy. This representation highlights how the strategy was constructed by the algorithm through the selection and combination of these components.

B. Testing Environment

To evaluate the performance of the strategy implemented by the algorithm, two distinct and previously unknown time periods were analyzed: 2017-2018 and half of January 2025. The interval duration was consistently maintained at 1-hour, as in the training phase. The results, shown in the tables III and IV and figures 3 and 4, demonstrate that the strategy outperformed the market in both periods, yielding positive returns. In both cases, the strategy delivered superior performance compared to the market, highlighting its robustness and effectiveness under varying market conditions.

IV. Conclusion

A. Problems, mitigations and lesson learned

Developing and testing a trading strategy in such a chaotic and dynamic environment presents numerous challenges. The unpredictability of the market, the varying efficiency of different strategies across different periods, and the inherent randomness of financial movements make it difficult to achieve consistently high performance.

One of the key challenges encountered was to find the right parameters facing the extensive runtime required for simulations. Due to the nature of genetic algorithms, balancing

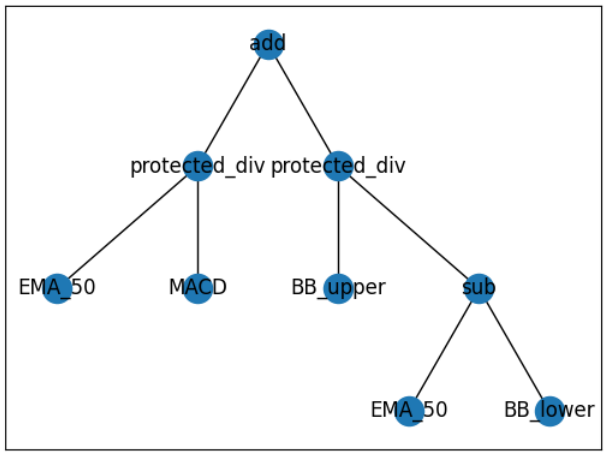

Fig. 2. Best Strategy identified by the algorithm

exploration (searching for new, potentially better strategies) and exploitation (refining existing successful strategies) proved to be a crucial but difficult task. Finding the right parameters to maintain this balance was essential to prevent overfitting while ensuring meaningful strategy improvements.

Another major issue was designing an effective fitness function. Initially, multiple variations were tested, but counterintuitively, simpler functions provided better results. More complex fitness functions often introduced unnecessary bias or over-optimization, which led to suboptimal generalization in new market conditions.

Also different training datasets and strategies were tested, but at the end, the individuals more profitable and prepared for unknown environments were the one trained with all the different types of market trends.

The key lesson learned from this project is that more complex solutions are not always superior. Then, it's crucial to evaluate all parameters of the model carefully, as each parameter can significantly impact the overall performance. Even small adjustments can make a considerable difference in the effectiveness of the strategy.

B. Conclusion

Finding a profitable and stable strategy for the cryptocurrency market is a challenging task. However, through the use of a fine-tuned genetic programming (GP) implementation, it is possible to develop novel strategies and explore new possibilities. In this paper, we demonstrate the ability to discover a new strategy that not only performs profitably in previously unseen environments but also outperforms the market. It is important to acknowledge that these markets are highly unstable, and a single, non-continue adaptable strategy may become inefficient over extended periods. Given the dynamic nature of financial markets, strategies need to remain flexible to adjust to changing conditions. Additionally, in real-world trading, factors such as transaction fees and slippage must be taken into account, as they can significantly impact the overall profitability of the strategy. These considerations highlight the need for a robust approach that can adapt to market fluctuations and account for operational costs.

C. Future works

While the results demonstrated the feasibility of the approach, there is room for further improvement. Future work can focus on refining the adaptation mechanisms within the genetic algorithm to make it more responsive to real-time market changes. Different strategies for different market trends could be used to more adaptability and better performance in every moment. Exploring hybrid approaches, such as combining genetic algorithms with other types of Evolutionary Algorithms, could also be a promising avenue for improving strategy robustness.

Fig. 3. Best Strategy tested in 2017-2018

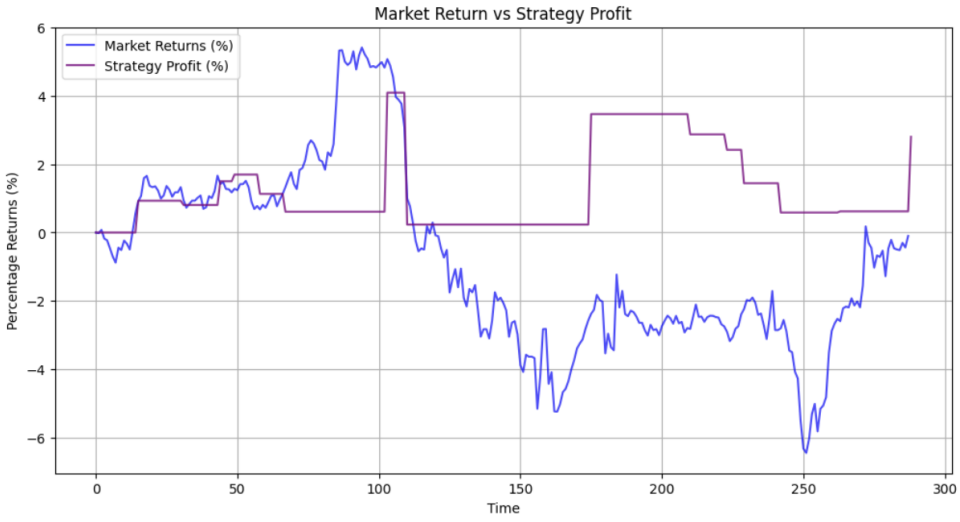

Fig. 4. Best Strategy tested in January 2025