Optimisation of Trading Strategies using Genetic Algorithms

Word Count: 2993

Github: github.com/jasperpato/Evolutionary-Trading-Bot

Tin Chi Pang
University of Western Australia
23301921@student.uwa.edu.au

Jasper Paterson
University of Western Australia
22736341@student.uwa.edu.au

Dhruv Jobanputra
University of Western Australia
22704304@student.uwa.edu.au

Asad Maza
University of Western Australia
21211711@student.uwa.edu.au

Gnaneshwar Reddy
University of Western Australia
23832048@student.uwa.edu.au

***Abstract*—Genetic algorithms (GAs) are a family of evolutionary algorithms that maintain a population of candidate solutions in the hypothesis space and iteratively alter each solution via genetic operators. Various studies have applied GAs to the field of cryptocurrency trading, yet few focus on evolving trading strategies represented by disjunctive normal form (DNF) expressions. In our paper, we aim to evolve trading strategies represented by DNF expressions that maximise profit using candle values and technical indicators. We represent each DNF expression as a chromosome that is generated randomly with randomly chosen indicators. We experimented with the end-portfolio value and the Sortino ratio as the fitness function and conducted multiple trials with various population sizes. When evaluated on the testing set and against benchmark agents, we found that the best portfolio-trained agent consistently outperformed the Sortino-trained agent. However, our result may be suboptimal due to numerous methodological limitations, such as the lack of population diversity, non-representative dataset, and suboptimal GA hyperparameters. Future investigations should address these limitations to produce more robust and adaptive trading strategies.**

***Index Terms*—genetic algorithm, technical analysis, Bitcoin trading, optimisation**

I. Introduction

Genetic algorithms (GAs) are a family of evolutionary algorithms that maintain a population of candidate solutions in the hypothesis space [1]. Searching the space involves selecting a subset of the solutions to perform crossover and mutation which furthers their location on the search space to hopefully areas with better fitness value.

In our paper, we have used GAs to evolve a population of trading strategies to optimise the return profit from Bitcoin trading. Specifically, each strategy is represented by a buy and sell trigger that themselves are represented by a disjunctive normal form (DNF) expression.

II. Methodology

We chose GAs for several reasons. For one, much of the research in the field of cryptocurrency trading utilises GAs in their methodology to optimise trading strategies, establishing a precedent for our work [2]–[4]. Additionally, chromosomes provide an intuitive representation for combinatorial strategies such as DNF expressions. GAs are much simpler to implement in contrast with other techniques such as differential evolution which may also be less effective for combinatorial strategies [5]. Furthermore, GAs with a diverse population can explore a wider area of the search space simultaneously, offering parallelism that reduces the amount of time required to obtain an optimal solution, making GAs an apt solution to our optimisation problem.

A. Chromosome Representation

Each DNF expression is represented by a chromosome encoded as a 3D list, where disjunctions, conjunctions, and literals are each represented as a list with the relevant operators implicitly joining each element. For example, the following expression

$$(\neg(A > C * B) \wedge (C > C * C)) \vee (B > C * A) \quad (1)$$

is represented by the 3D list below.

```
[
    [["not", "A", "C", "B"], ["C", "C", "C"]],
    [[ "B", "C", "A"]]
]
```

Each DNF expression is randomly generated and soft-bounded using probability, where conjunctions and literals each have a 10% chance of being extended. Though the space of all DNF expressions is technically infinite, the chance of generating an n-lengthed expression diminishes as n grows, making most of this space practically inaccessible.

We observe that when both buy and sell chromosomes are generated independently, the transaction frequency of the agents reduces, likely because the probability that both

chromosomes will frequently trigger transactions is low. This is addressed by deriving the sell chromosome from the buy chromosome by swapping the first and last value in its literals, making transaction frequency depend solely on the buy chromosome.

During DNF generation, the symbol A may take up any indicators from the `ta` module, excluding the ADX indicators as they result in errors during random parameter generation. In contrast, only the closing price is used for the symbol B.

The initial range of numeric values is also restricted. Upon indicator selection, its arguments are randomly generated within a range $[k - x, k + x]$ where k is the default value of a given parameter and $x = 10$ for integers and $x = 5.0$ for float values. Though, all values are clipped to a minimum of 1 or 0 for integers and floats respectively, meaning this range is smaller in practice. Initial constants in expressions have the range $[0.0, 5.0]$ and all float values are discretised by rounding to three decimal places, creating a finite initial numeric space.

During the GA, there are no hard bounds for numeric values. Instead, we use soft-bounding, allowing values to exceed initial bounds but discarding them if they don't contribute to fitness.

B. Genetic Algorithm

Our GA is taken and modified from [6] which uses roulette wheel selection [7]. The pseudocode is provided below.

Algorithm 1: Genetic Algorithm

```
Input : Population size n
Input : Number of offsprings n_o
Input : Initial mutation probability p = 0.3
Input : Number of iteration m
S ← Generate a random population with size n;
evaluate(S);
best ← get best individuals from S;
for iteration = 0 until m do
    p' ← p((m − iteration)/m);
    mating ← rouletteWheelSelection(S, n_o);
    offspring ← crossover(mating);
    for s ∈ offspring do
        s ← mutate s with chance p';
    end
    offspring ∪ best;
    offspring ∪ migration(n − n_p);
    S ← offspring;
    evaluate(S);
    best ← get best individuals from S;
end
return best solutions;
```

We implemented adaptive mutation where the chance of mutation in each iteration p' decreases as the number of iterations continues, intending to pivot the search away from exploration to exploitation during later stages of the algorithm. Additionally, migration is used to repopulate the offspring set with random agents to introduce diversity into the population.

C. Genetic Operators

The crossover operator is a form of one-point crossover where, given two randomly sampled parents, a random crosspoint is chosen for each chromosome [6]. The chromosome exchanges segment at that point to form two new DNF expressions for the resulting children. Due to limitations in our list representation, crosspoints are only chosen at disjunctions.

When a mutation occurs, one of four operations can randomly occur:

1) The list of constants is randomly shuffled.
2) Gaussian noise is applied to the integer parameters for a single indicator with $\sigma = 3$
3) Gaussian noise is applied to the float parameters for a single indicator with $\sigma = 0.5$
4) Gaussian noise is applied to the constants with $\sigma = 0.5$

When Gaussian mutation is performed on a list of numeric values, a bit mask is additionally generated that determines which value in the list will be changed by the Gaussian noise.

D. Fitness Functions

We compared two fitness functions in evaluating trading strategies: portfolio value and the Sortino ratio. To discourage passivity, a fitness value of -1 is assigned to agents who did not trade.

1) Portfolio Value: The portfolio value is calculated as the total return on investment (ROI) for the evaluation period, i.e., the end-portfolio value of a given strategy, without accounting for transaction behaviours during training.

2) Sortino Ratio: The Sortino ratio is a risk-adjusted performance metric that takes into account the downward risk of a trading strategy, penalising downward deviations for daily portfolio values throughout the trading periods [8]. It's given by the equation

$$\textit{Sortino Ratio} = \frac{R_p - R_f}{\sigma_d} \tag{2}$$

where R_p is the average daily return achieved by the portfolio, R_f is the rate of return on a risk-free investment chosen based on government bonds, and σ_d measures the volatility of negative returns in the portfolio. R_f is set to 1.2% as it approximates the 10-year Australian bond yield rate for the training period, divided by 365 to convert from annual to daily rate [9]. The Sortino ratio is a variation of the Sharpe ratio which is used to calculate the overall market volatility, while the Sortino ratio only considers the downward volatility. This enables us to calculate the level of risk for the return and measure if the strategy is able to make good transactions for the risks it takes.

III. Experiments and Results

A. Experimental Setup

We conducted 30 trials using the Sortino ratio and 30 trials using the portfolio value as the fitness function. Both groups are divided into three batches of ten trials that each used a population size of 50, 100, and 200. All trials ran for 100

Fitness function	Portfolio			Sortino		
Population size	50	100	200	50	100	200
Portfolio value	102.90	114.28	124.55	94.45	98.37	89.74
Sortino ratio	-	-	-	0.0101	0.0140	0.0035

TABLE I
AVERAGE FITNESS OF THE BEST STRATEGIES IN EACH BATCH FOR THE TRAINING DATA

Fitness function	Portfolio			Sortino		
Population size	50	100	200	50	100	200
Portfolio value	122.45	173.77	167.50	140.73	133.02	114.01
Sortino ratio	-	-	-	0.0536	0.0853	0.0287

TABLE II
FITNESS VALUE OF THE BEST STRATEGIES IN EACH BATCH FOR THE TRAINING DATA

iterations using a mutation probability of 0.3, and the number of offspring was set to be 70% of the population size. The candle data is split into the training and testing set with an 80:20 ratio.

B. Best Fitness

The strategy with the best fitness during testing is taken from the final population of each trial.

The average fitness of top-performing strategies per population size and fitness function is in Table I, and the fitness of the best strategy per population size and fitness function is shown in Table II. As expected, almost all portfolio-trained strategies achieved higher portfolio value than those in the Sortino group, with the exception that the best individual in the portfolio group with a population size of 50 which had a lower portfolio than the best individual trained with population sizes of 50 and 100 in the second group. For Table I, the average portfolio increases with population size in the portfolio group, yet this trend is not observed in the Sortino group in Table II.

For both groups, the strategy that generated the most profit in the testing set is selected as the best-performing agent.

The best strategies are then evaluated against the testing set with the results outlined in Table III, where two of the best portfolio-trained strategies outperformed all the Sortino-trained strategies, with the highest portfolio achieved being $170.34. The last portfolio-trained strategy ended with a worse portfolio than it started with. Notably, two of the Sortino-trained strategies did not trade as indicated by the -1 fitness penalty.

Fitness function	Portfolio			Sortino		
Population size	50	100	200	50	100	200
Portfolio value	114.03	170.34	89.98	100.00	104.05	100.00
Sortino ratio	-	-	-	-1.0	0.0337	-1.0

TABLE III
FITNESS VALUE OF THE BEST STRATEGIES IN EACH BATCH FOR THE TESTING DATA

Strategies	Portfolio
Best Portfolio	173.77
Best Sortino	133.02
Simple	44.00
Random	35.72

TABLE IV
PORTFOLIO OF THE BEST PORTFOLIO-TRAINED AND SORTINO-TRAINED AGENTS AGAINST RANDOM AND SIMPLE AGENTS ON THE TRAINING SET

Strategies	Portfolio
Best Portfolio	170.34
Best Sortino	104.45
Simple	129.74
Random	119.40

TABLE V
PORTFOLIO OF THE BEST PORTFOLIO-TRAINED AND SORTINO-TRAINED AGENTS AGAINST RANDOM AND SIMPLE AGENTS ON THE TESTING SET

C. DNF Expression of Best-Performing Strategies

The best-performing portfolio-trained strategy has the buy chromosome

$$(\text{close} > 0.46 \cdot \text{FI}) \wedge \neg(\text{close} > 0.46 \cdot \text{TSI}) \tag{3}$$

where FI is the force index with the parameter `window=28` and TSI is the true strength index with parameters `window_slow=23, window_fast=21`.

The best-performing Sortino-trained strategy has the buy chromosome

$$\neg(\text{KeltnerChannel}_{\text{high}} > 0.927 \cdot \text{close}) \vee (\text{close} > 4.285 \cdot \text{PSAR}_{\text{up}}) \tag{4}$$

where $\text{KeltnerChannel}_{\text{high}}$ is the high band for the Keltner channel with parameters `window=23, window_atr=1` and PSAR_{up} is the up trend of the parabolic stop and reverse.

The sell chromosomes for both strategies are omitted for brevity as they can be induced from the buy chromosomes.

D. Benchmark Strategies

We compared the strategy with the highest fitness during training from both Sortino and the portfolio group against two agents. The random agent trades with a 5% probability at any time, and the simple agent buys when the EMA indicator crosses over the SMA indicator and sells when the reverse occurs. The portfolio of all four agents is shown in Table IV and Table V, and the trading pattern is graphed in Figure 1 and Figure 2. Both trained strategies outperformed the simple and random agents on the training set. Unexpectedly, the Sortino-trained agent performed the worst on the testing set compared to other agents, with the portfolio-trained agent once again outperforming the other strategies. The trained agents generally transacted less frequently when compared with the benchmark strategies.

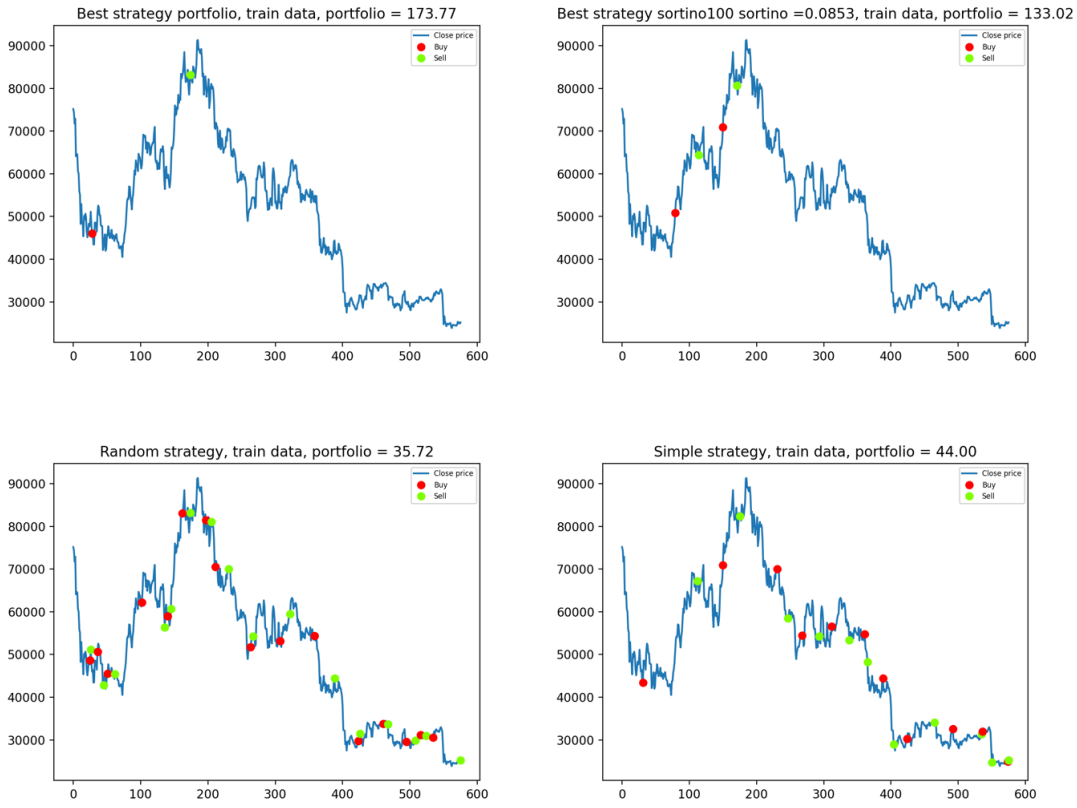

Fig. 1. Transactions of the best portfolio-trained, best Sortino-trained, random, and simple agent on the training set

IV. Discussion

A. Result Evaluation

We expected that portfolio-trained strategies would yield a higher portfolio in the training set and anticipated they will underperform when compared against the Sortino-trained strategies during testing. Theoretically, Sortino-trained agents should generalise better to the test data as the Sortino Ratio accounts for each transaction behaviour. However, this was not the case, as the portfolio-trained agents almost always outperformed the Sortino-trained agent. This may be due to the low Sortino ratio of the agents which was mostly below 0.1, yet a typically good ratio is two or above. This perhaps is due to the volatile nature of the Bitcoin market where each transaction incurs high risks that heavily outweigh the profit, and hence the Sortino ratio may not be appropriate for this market.

The best-performing strategies from both groups traded infrequently which may be due to the 2% transaction cost involved that penalises frequent transactions. For the best portfolio-trained agent, it was able to buy and sell near the local minimum and global maximum respectively to generate a large return on both training and testing sets. The best Sortino-trained agent instead performed four transactions in the training set on upward trends. Similarly, in the testing set, it only buy on upward trends and did not sell at the peak. Its avoidance of local minima may be due to its aversion to risks, only trading when it's certain the current price is on an upward trajectory.

Comparing the two fitness functions overall, it was found that using the portfolio was able to maximise the profits made by the strategies, but also increased the variance between the strategies, whereas, the Sortino ratio allowed for more consistent yet lower returns.

Although we have observed profitable strategies, several limitations in our study must be addressed to determine the validity of our results.

B. Diversity Deficit

We observed that many of the top ten strategies in each trial consist of multiple individuals with the same chromosome which indicates a deficit of diversity in our population. As we have implemented migration and used random initial populations, this observation implies our crossover and mutation operator failed to maintain diversity. Our soft-bounding of DNF expression lengths means most expressions will only have one to two disjuncts. The crossover operator - which only selects crosspoints between disjuncts - is thus likely to swap the whole DNF expression if it only has one disjunct as opposed to individual segments which results in children

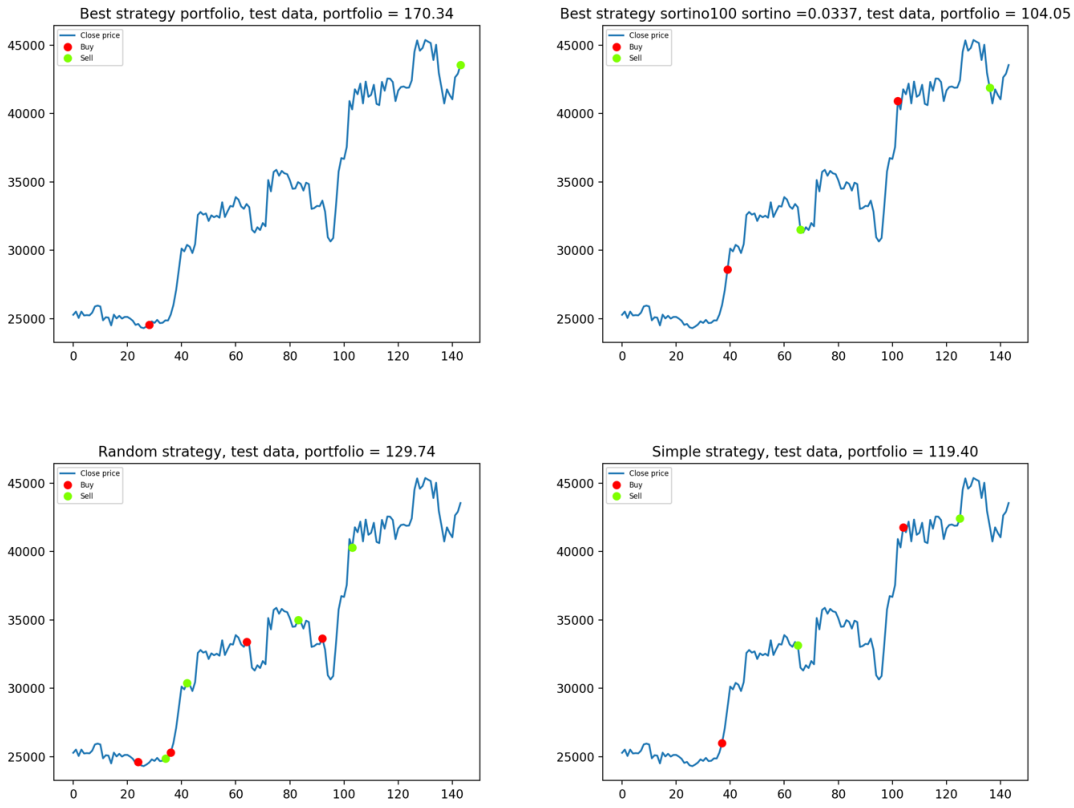

Fig. 2. Transactions of the best portfolio-trained, best Sortino-trained, random, and simple agent on the testing set

that are the exact copy of their parent. To remedy this, a tree representation may better allow crossing over via the exchange of subtrees on any nodes in the tree. The mutation operator also failed to mutate the numeric parameters adequately. When a mutation occurs, the target of mutation is chosen randomly without considering the chromosome content, meaning an empty list can be chosen for mutation. For indicator mutations, only the parameters of one indicator are mutated which again may not contain any elements at all. If a non-empty list is selected, each value in the list is still not guaranteed mutation due to the randomly generated bitmask. When combined, the likelihood that a value is mutated is greatly diminished, making it difficult to introduce diversity. This then implies our GA failed to search the hypothesis space adequately and may have converged prematurely to suboptimal solutions.

C. Non-Representative Dataset

It's possible that the overall trend of the testing data biased our results. The testing set displayed a steep upward trend that almost guarantee a profit, as seen by the result of the random strategy which generated profit despite acting randomly. Our testing set thus does not accurately reflect the market conditions which includes both upward and downward trends. To address this, we could instead segment the data into different epochs that reflect various market conditions to train the agents on, potentially resulting in more robust and adaptive strategies.

D. GA Hyperparameters

Due to time constraints, we did not hyper-tune the GA hyperparameters which may lead to suboptimal performance. For instance, we included all indicators in the `ta` module such that our hypothesis space may be excessively large that our chosen iteration and population sizes cannot adequately explore this space, though this can also be solved by further restricting the number of indicators used. Hyper-tuning techniques such as grid search which considers each combination of hyperparameters values may be employed in future investigations to optimise the performance of the GA [10].

V. Conclusion

In conclusion, we have used genetic algorithms (GAs) to evolve a population of trading strategies using technical indicators, representing the buy and sell expressions as chromosomes. We found that in the testing set, only the best portfolio-trained agent outperformed the random and simple agent while the best Sortino-trained agent generated the least profit. Yet, various limitations in our approach should be accounted for when considering our results, such as a low diversity in the populations, a non-representative dataset, and

suboptimal choices of GA hyperparameters. Regardless, we demonstrated the potential for optimising Bitcoin trading using GAs. Future investigation should address these design flaws using a larger, more representative dataset and incorporate systematic hyper-tuning such as grid search to improve the performance of the GA to create more robust and adaptive trading agents.

REFERENCES

[1] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091–8126, Feb. 2021. [Online]. Available: https://doi.org/10.1007/s11042-020-10139-6

[2] L. Cocco, R. Tonelli, and M. Marchesi, "An Agent-Based Artificial Market Model for Studying the Bitcoin Trading," *IEEE Access*, vol. 7, pp. 42 908–42 920, 2019, conference Name: IEEE Access.

[3] D. F. Gerritsen, E. Bouri, E. Ramezanifar, and D. Roubaud, "The profitability of technical trading rules in the Bitcoin market," *Finance Research Letters*, vol. 34, p. 101263, May 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1544612319303770

[4] K. Grobys, S. Ahmed, and N. Sapkota, "Technical trading rules in the cryptocurrency market," *Finance Research Letters*, vol. 32, p. 101396, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1544612319308852

[5] X. Yu and M. Gen, *Introduction to evolutionary algorithms*. Springer Science & Business Media, 2010.

[6] G. Acampora, R. Schiattarella, and A. Vitiello, "Quantum Mating Operator: A New Approach to Evolve Chromosomes in Genetic Algorithms," in *2022 IEEE Congress on Evolutionary Computation (CEC)*, Jul. 2022, pp. 1–8.

[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[8] T. N. Rollinger and S. T. Hoffman, "Sortino: a 'sharper'ratio," *Chicago, Illinois: Red Rock Capital*, 2013.

[9] [Online]. Available: http://www.worldgovernmentbonds.com/bond-historical-data/australia/10-years/

[10] D. Shanthi and N. Chethan, "Genetic algorithm based hyper-parameter tuning to improve the performance of machine learning models," *SN Computer Science*, vol. 4, no. 2, p. 119, 2022.