Pattern searcher for decision making of trading agents using Genetic Algorithm

Felipe V. Cacique
DCC - UFMG
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
vitalcac@ufmg.br

Adriano C. M. Pereira
DCC - UFMG
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
adrianoc@dcc.ufmg.br

***Abstract*—In the last few years, there was a growth regarding the use of computational methods in the field of finance, especially to negotiations in the stock market. In this paper, we aim to bring new ideas and approaches to the development of automated trading or bots based on historical data of financial series. Our model, named Pattern Searcher, was inspired in unsupervised learning methods and evolutionary optimization. Given a trading agent with its predefined parameters, the method uses the power of Genetic Algorithm (GA) to search, within a set of financial indicators, for the region that provides a higher positive financial return. This implementation exhibited desirable properties compared to some Machine Learning methods, such as the simplification of the system flow and the generation of rules that humans can clearly understand. Besides, we have generated strategy portfolios, composed by the strategies derived from the Pattern Searcher method, which were also optimized via GA. The system was able to generate very profitable trading agents and portfolios on the Brazilian stock market, surpassing important benchmarks.**

***Index Terms*—Genetic Algorithm, trading agent, stock market, portfolio, leverage, Pattern Searcher, finance.**

I. Introduction

The stock market can be a very profitable place for those who have good trading strategies. Based on rules, people can decide when is the right time to buy, sell or hold a stock, find out the involved risks, the financial income, and the amount of necessary investment. If one finds out a pattern that leads him to buy a stock when the price is low and sell it when the price is high, for example, he will receive profitable financial returns. A good strategy may depend on exhaustive tests and validation on historical data, in order to know how the proposed strategy would have worked in the past, to estimate how it will work in the future, and to best adjusts its parameters.

It is common for investors to make trading decisions based on technical indicators, which is an analysis focused on the pattern of price movements [1]. In a simplified manner, the buy and sell signal can be represented by Boolean expressions using a combination of those indicators. For example, in a typical moving average crossover strategy [2], a buy order is sent when the price crosses above the moving average of x periods and there is an increase in the traded volume, that may confirm an uptrend movement. Similarly, when the price cross below the moving average, a sell order is sent.

We can improve and create new strategies by combining different financial indicators and adjusting their parameters on historical price data. Usually, this is a time expensive process due to the huge number of indicators and parameters that can be used, even for computers. Thus, looking for computational optimization techniques, such as Genetic Algorithm, is a must.

Genetic Algorithm (GA) is an optimization technique that can be used for a variety of problems, such as designing aircraft, evolving electronic circuits, finding hardware bugs and optimizing asset portfolios. GA is an analogy to Darwin's theory of evolution of species and the genetic field, in order to develop an algorithm that searches for solutions in a variety of computational problems [3]. The idea is to evolve a population of possible solutions in the searching space of the problem. At each generation, the strongest individuals within the population are more likely to survive and combine their genetic material to generate better individuals.

In this context, the goal of this project is to develop an algorithm that searches for profitable patterns for decision making of automatized trading agents (also called trading robots), using Genetic Algorithm. In other words, creating a generator of trading strategies. The main idea is that the GA will evolve trading rules to hopefully create more profitable strategies, which tells the right time to buy or sell a stock. In addition, the GA will also be applied to generate and optimize portfolios using the found strategies, aiming to potentially improve the financial returns. The implementation and optimization were done in the trading software Metatrader 5 (MT5), using its MQL5 programming language and its strategy tester. This choice was done intending for a simple and quick transition between simulated and live trading operations.

The remainder of this paper is organized as follows: Section II presents a literature review addressing Machine Learning and Evolutionary Algorithm techniques applied to finance. The following is the methodology adopted to achieve the proposed goal, including the Pattern Searcher modeling in Section III, the controlled leverage methodology in Section IV and the generation of strategy portfolios in Section V. Section VI presents the experimental validation of the trading agents and the agent portfolios trained via Genetic Algorithm. Finally, Chapter 5 concludes the work, summarising the most relevant results and discussing future works.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

II. Related Works

The majority of computational tools used by investors to automate investment strategies and create stock trading systems uses Machine Learning methods, especially Long Short Term Memory (LSTM). LSTM is a Recurrent Neural Network that has memory mechanisms that are well-suited for classifying and making predictions based on time series data [4].

[5] combined stock prices with stock news data from Google Trends using the LSTM network for decision support. The use of LTSM improved the predictability from 51.9% to 58.2%, compared to the simple model of Neural Network MLP (Multi-Layer Perceptron). Similarly to this work, [6] used the LSTM to predict the volatility of the Shanghai Shenzhen CSI300 Index, achieving an accuracy of 78% in the predictions.

The LSTM neural network has shown to be very appropriate for temporal series like financial data. However, as market behavior changes over time, this model needs to be periodically retrained. This creates a necessity for models that are able to dynamically adapt to new market conditions. An alternative is the use of Reinforcement Learning.

Reinforcement Learning (RL) is a learning system where an agent perceives the environment and takes actions in order to maximize a notion of cumulative reward [7]. In the context of automated trading systems, [8] used the Q-Learning and proposed algorithm, named as Recurrent Reinforcement Learning (RRL), to trade a portfolio. They used the Sharpe index as the reward function, which measures a relation between the portfolio profitability and the investment risk. Both algorithms were profitable and surpassed the baseline Buy-and-Hold.

In literature, there is a parallel approach to Machine Learning that uses Evolutionary Algorithms (EAs), such as Genetic Algorithm (GA), for creating trading decision support systems. Evolutionary Algorithms has been used for feature selection (select the best inputs for a Machine Learning model), rule parameter (find the best parameters for a specific rule or system), rule combination (combine the best rules) and rule construction (evolve more complex rules from simple ones).

[9] compared two Genetic Systems for creating trading strategies. In the first, GA was used to train the weights of a Neural Network, which received technical indicators of NASDAQ stocks as inputs. The second system used Genetic Programming (GP) to derive trading strategies in a tree representation. Both systems returned, as output, trading signals such as buy, sell and hold. They found that both were more profitable than the Buy-and-Hold strategy, which is usually used for comparison in many works.

In [10]'s work, GA was used for rule construction in intraday trading, and its performance was compared to a Reinforcement Learning system, a Markov Decision Process-based system and a heuristic. The rules found by GA exhibited several desirable properties. Unlike many Machine Learning implementations, the rules are not "black boxes" and can be understood by humans. The GA outperformed the other methods in new unseen data, although none of the methods produced significant profits at realistic transaction costs.

[11] has developed a decision making and trading algorithm named Goldminer. It used the power of Genetic Programming, combined with indicators of technical analysis to identify the time of purchasing and selling a stock. The individual was composed of the logical operators AND, OR, XOR and the indicators of technical analysis, while the fitness was calculated simply by counting the profitability (all profits subtracted from all losses). The inclusion of the two windows verification to validate the models was a key distinguishing point for reaching the target time and profit in more than 90% of the runs tested on the Brazilian stock market B3 (*Bolsa, Brasil, Balcão*)

[12] did similar work. They have implemented an Evolutionary Algorithm for generating trading rules for intraday trading, but with different individual representation. The individual was a binary decision tree, directed acyclic, whose leaf nodes contained buy and sell decisions. As the fitness function, they used the total stock financial return. When the strategies generated by the algorithm were tested in the forward test, in the major cases, the profitability degraded drastically. However, the strategies remained efficient, and in all cases, it exceeded the rate of risk-free return, and the maximum drawdown values (worse consecutive sequence of losses) stayed within 7%.

There is a financial forecasting tool based on Evolutionary Algorithms designed to help investors to create trading rules in a form of Genetic Decision Trees that can be readable, called EDDIE (Evolutionary Dynamic Data Investment Evaluator) [13]. According to [14], its most recent version (EDDIE 8) is not constrained in using pre-defined indicators, but it automatically chooses the optimal ones. The tests were performed in an artificial data set, which the authors knew patters existed. Eddie 8 was able to find those patterns, but it seems that it was having difficulties with searching effectively in the space state.

According to [15], studies applying GA for rule discovery in algorithmic trading have been increasing in the last few years, but few papers have been published so far. There is still a lack of comparison between Evolutionary Algorithm implementations and other techniques. It indicates that there is a need for more study on this topic and there is still plenty to explore. In their paper, they suggested some future research directions, including:

1) Predicting future market trends: predict future market trends and search for more predictable and influential classification standards can bring much help (e.g., strategies can be selected accordingly to the prediction);
2) Considering liquidity and transaction costs in more precise and positive ways, and combining portfolio selection techniques and period for training and learning;
3) Combining portfolio selection techniques: researchers need to consider the correlation of stocks in different environments, relationships between trading rules, transaction cost, and selection of fitness function.

In the field of finance, backtest overfitting is one of the most important open problems [16]. Backtest overfitting is when the developed strategy performs well in the backtest because it has "memorized" random patterns of the data set,

rather than learning important features about it. As those patterns are unlikely to happen in the future, the strategy fails in real account trading. Several steps can help to reduce the overfitting presence, including developing models for entire asset classes, rather than for specific securities [16], and reducing the model complexity [17].

In our work, we aimed to address some of the suggestions of [15] and also to develop a system with reduced complexity, attempting to avoid overfitting.

III. Pattern Searcher model trained via Genetic Algorithm

This section describes the Pattern Searcher modeling, the training process of the model using GA, and the normalization of the indicators that were used as model input.

A. Pattern Searcher model

In this work, we propose a model for decision making of trading agents, which we named Pattern Searcher. This model, inspired by Machine Learning unsupervised methods and Evolutionary Algorithm, automatically searches for buy and sell patterns using as input a set of indicators. In order to present details about the modeling, we have divided it into: *Idea*, *Hypothesis*, *Problem*, *Solution*, and *How to use the system*:

Idea: Given a trading agent with its pre-defined proprieties (e.g., stop loss, take profit, stop trail, close position time, indicator parameters) and a set of financial indicators (e.g., MACD, RSI, MA, PIVOT), the method will search, within this set of indicators, for the region that provides the higher positive financial return.

Hypothesis: There is a region within the multidimensional space of indicators that gives highly positive financial returns for a given operation (e.g., buy or sell), which can be covered by a geometric primitive (e.g., ellipses, boxes). Fig. 1 illustrates a profitable region, composed by 2 indicators, being covered by a box.

Problem: Find the properties of the geometric primitive that cover the region (e.g., center and radius for ellipses, or center and edge for boxes).

Solution: Use of Genetic Algorithm to find the edges es and the centers cs of each indicator, in case of a box, in order to maximize the financial returns for a given historical dataset.

How to use the system for decision making: The agent will only open a position if the current indicator signals are inside the geometric primitive.

In the situation where the geometric primitive is a box, we can express the rules derived from the Pattern Searcher in the form of Boolean expression. This is an example of how the rule would looks like: **if** $MACD(x_1, x_2, x_3) > y_1$ **and** $RSI(x_4) < y_2$ **then** BUY. This Boolean expression means that the buy position opens if $MACD$ signal is above y_1 and RSI signal is bellow y_2. Notice that, beside the specification of the indicators, this expression have two kinds of parameters: the indicator's parameters (x_1, x_2, x_3, x_4) and the signal thresholds (y_1, y_2). Our model focuses primarily on finding the signal threshold ys, while fixating the values of the indicator's parameters xs. Even though the indicator's parameters could be optimized simultaneously with the signal threshold, it would increase the complexity of the model and also the chance of overfitting [17], since too many parameters would be optimized at once. Thus, keeping our model as simple as possible may help to prevent overfitting.

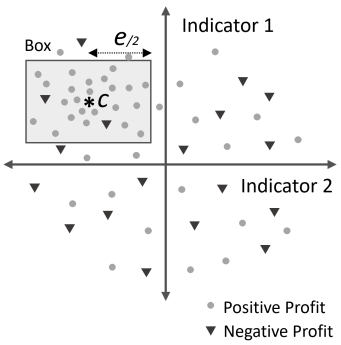

Fig. 1. The dots and triangles represent, respectively, positive and negative financial returns from trades on historical market data, performed by an agent randomly opening positions. For a given set of indicators, it might end up generating a concentration of positive financial returns, which is a region of high expected return. If this region exists, with the Pattern Searcher method we could cluster it using geometric figures, which in this case is a box of center c and edge e.

We have defined the domain of the center and the edge of the Pattern Searcher model as $-1 \leq c \leq 1$ and $0 \leq e \leq 2.4$. All the indicators were normalized within the interval from -1 to 1, and by this, the geometric figure can cover the whole space, in case of $c = 0$ and $e = 2.4$, for example. This allows our model to not only optimize the parameters but also to do *feature selection*. It means that poor features added to the model can be automatically removed during training if the figure covers completely the feature dimension.

B. Geometric primitives and indicators

The regions with highly expected financial returns within the multidimensional space of indicators may have different shapes, so different geometric figures could be used. In this work, we have implemented ellipses and boxes:

Ellipse: It can be represented by a center c and a radius r for each indicator: $Ellipse = (c_1, r_1, c_2, r_2 \ldots c_n, r_n)$. The region inside an ellipse is given by (1):

$$\frac{(c_1 - r_1)^2}{r_1^2} + \frac{(c_2 - r_2)^2}{r_2^2} + \ldots + \frac{(c_n - r_n)^2}{r_n^2} \leq 1 \quad (1)$$

Box: It can be represented by a center c and an edge e for each indicator: $Box = (c_1, e_1, c_2, e_2 \ldots c_n, e_n)$. The region inside a box and the trade rule can be given by a Boolean expression:

If $(c_1 - e_1/2 < indicator_1 < c_1 + e_1/2)$ **and**
$(c_2 - e_2/2 < indicator_2 < c_2 + e_2/2)$ **and**
$\ldots$
$(c_n - e_n/2 < indicator_n < c_n + e_n/2)$ **then**
Open Position$(TYPE)$

In our experiments, we have used boxes only. The reason for this is that the possibility of representing the trade rules by Boolean expressions makes easier their interpretation.

We have used 10 different indicators, which were: Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Moving Average (MA), Trend Strength Indicator (ADX), Money Flow Index (MFI), Stochastic Oscillator (STO), Bollinger Bands (BB), Pivot Points (PVT), Volume, and Time (day time in minutes). Those indicators were slightly modified to return a single float signal without losing the original information that they were designed for.

C. Model optimization using Genetic Algorithm

We have stated the problem of finding the parameters of the geometric primitive, which are the boxes centers and edges for each indicator. To accomplish that, we have used Genetic Algorithm, as it has shown to be extremely efficient in optimizing parameters in a variety of problems. In the GA perspective, the problem resumes in "finding the optimal parameters in order to maximize the profits". The process was divided into: *Setting the agent parameters*, and *starting the optimization*. Fig. 2 shows these steps and their details.

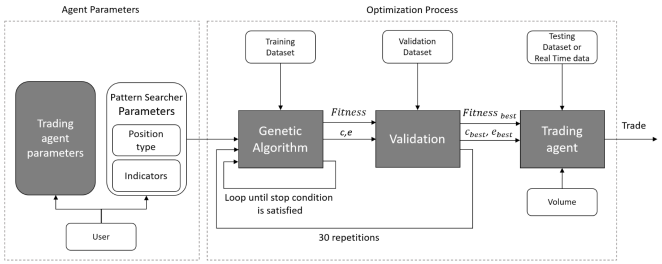

Fig. 2. Pattern Searcher optimization diagram. After all the parameters are configured, the agent's Pattern Searcher is trained on a historical dataset using GA. The best individual from validation is taken for testing on unseen data.

Before the optimization, we have *set the agent parameters*. Firstly we have configured the agent strategy variables regarding risk management, position management and order management, which includes variables such as take profit, stop loss and volume of contracts. The values of those parameters were chosen accordingly to some prior experience with trading strategies design and the desired outcome. About the later, we desire to capture large movements of the market, since it would improve the financial returns while decreasing the brokerage costs. Because of this, we have set large take profits and stop losses, around 1% of the asset price. Next, we have configured the *pattern searcher parameters*. Here, we have chosen the indicators to be used, the trade type (buy or sell) for each pattern and the number of patterns the agent will trade (up to 4 patterns). Each pattern was trained separately, attempting to reduce the number of the GA input variables.

After all the parameters were chosen, we finally started the *optimization process*. It was done using the Metatrader 5 Genetic Optimizer [18]. Since it is a built-in MT5 tool, we could not have access to the crossover and mutation types and rates. The optimization consisted in *training*, *validating*, and *testing*.

In the *training*, the Genetic Algorithm evolved near to 40 *generations* of individuals until the stop criteria were satisfied. The *individuals'* chromosome were represented by the centers (cs) and edges (es) of each indicator that compounded an agent pattern. The cs' domain varied from -1 to 1, and the es' domain from 0 to 2.4. The *initial population* was generated with a range from 64 to 256 individuals [19], each of them representing a solution. All individuals were evaluated through a trading simulation onto the stock historical data and received a value of fitness that express the quality of the strategy. Our *Fitness function* was a combination between total profit and drawdown, given by: $Fitness = TotalProfit \times Drawdown$. It measures a relationship between profitability and risk, leading the GA not only to evolve individuals that have great profits but also individuals with low drawdown.

In the *validation*, we have checked the agent's performance on unseen data, quantified by the same fitness function used for training. It is a very important step for removing agents that have probably suffered from overfitting. These are agents that performed well in the training because they have "memorized" the data set, rather than learning interesting features about it. Agents that passed the validation have higher chances to be more robust and perform well in real trading. Since GA is a stochastic process, we have repeated the whole training and validation processes for 30 times, and the best individual from validation was taken for *testing*.

In the *testing*, we have evaluated the performance of the agents in a new unseen data, which reflects what we would expect in a real account trading.

D. Other considerations

There are some other relevant considerations that worth to be discussed, which are: the number of simultaneously positions allowed per agent, the inverse condition being satisfied, and the simplicity of the system flow.

The agents are only allowed to have one opened position at each time. This results in uncorrelated patterns within an agent. The reason for this is that correlated pattern signals would probably overlap and their trade signal would be ignored. Consequently, there would not be any improvement in the agents' performance. The only way to improve the total profit while not violating the restriction of contracts per agent would be finding patterns in which signals do not overlap. It would significantly improve the agent's efficiency.

In addition, the agents were programmed to close an open position if has passed 2 hours after the position was opened, or if the inverse condition was satisfied. About the later, if an agent has a buy position, and one of its patterns sends a signal to sell, for example, then the agent closes the buy position and opens a sell position.

Indeed, the implementation exhibited a desirable property compared to some Machine Learning methods, which is the simplification of the system flow. The model and the strategy are interconnected, in a way that the training directly optimizes the agent's performance in terms of profitability.

IV. Controlled Leverage and Monthly Financial financial return Calculation

Leverage is when an investor uses borrowed money in an attempt to increase the ratio of financial return of an investment [20]. A leverage ratio of 1:10, for example, means that we can negotiate 10 times more cash of what we have in the broker's account. This can potentially increase the financial returns, but also the losses.

One can ask: "How do we calculate the initial investment needed to trade with leverage, in order to have great financial returns while minimizing the risks?" To answer this question, we should not think about maximizing the financial returns but think about controlling the risks. What we want is a trade-off between financial returns and risk. Aiming to calculate the initial deposit, we have proposed a method that we will refer to as "Controlled Leverage". We have not found much information about this in literature, but we believe that many professional traders use a similar approach.

A. Controlled Leverage

The drawdown (DD), which is the worse consecutive loss, needs total attention because it is capable of breaking an account balance if there is not enough margin to trade in that period. Brokers may ask little margin per contract of Future assets, for example, which allows people to start trading with a very small amount of money. However, to reduce the risks and "survive" from the DD, we should invest more money per contract than the margin asked by the broker.

In this work we estimated the amount of initial investment ($BalanceNeeded$) based on the DD of the backtest ($DrawnDown$), defined by (2):

$$BalanceNeeded = DrownDown \times RiskCoef \quad (2)$$

Where the $RiskCoef$ is a risk coefficient. If the $RiskCoef$ is set as 5, it means that we will have 5 times more money in the account than the DD from backtesting. The higher the value we chose, the less will be the risk of breaking an account. We could also rewrite this formula by (3):

$$BalanceNeeded = DrawnDown/TradePct \quad (3)$$

Where $TradePct$ represents the percentage of the initial deposit that we are effectively trading. Lower $TradePct$ results in higher financial returns but also higher risks.

B. Monthly Financial Return

Once the initial deposit was estimated and we have the monthly profit ($MonthlyProfit$), we can calculate the monthly financial return by (4):

$$MonthlyReturn = MonthlyProfit/BalanceNeeded \quad (4)$$

Notice that the financial return is inversely proportional to the DD. Lower DD reduces the $BalanceNeeded$, and consequently increases the Monthly Financial Return. Aiming to decrease the DD, we have generated strategy portfolios, which are described in section V.

V. Strategy Portfolio

In finance, a portfolio is a group of financial assets that can include stocks, bonds, commodities, currencies and cash equivalents [21]. In this context, a strategy portfolio is a portfolio composed of different strategies with the twin objectives of maximizing financial return while minimizing risk. Instead of negotiating all contracts in one single strategy, we would distribute those contracts among a set of strategies. The weight w represents the percentage volume that each agent within a portfolio will trade. The weights were optimized through Genetic Algorithm. Fig. 3 synthesize the portfolio's training elements.

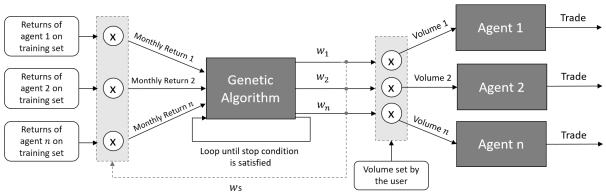

Fig. 3. Portfolio optimization diagram. The GA individuals are represented by weights. At each generation, the weights multiply the agents' historical financial returns, to generate portfolios with w's distribution. We calculate the monthly returns of those portfolios and use them as the GA fitness function.

The *individuals* were represented by an array with the portfolio weights (w's). w's could assume R values from 0 to 1. The *initial population* was generated with 100 individuals, that were decoded in solutions for the problem. It was done by, firstly, normalizing the weights with (5):

$$w_i = \frac{w_i'}{\sum_{i=1}^{n} w_i'} \times 100\% \quad (5)$$

Where the w_i value is the percentage of contract volume for each agent, and $\sum_{i=1}^{n} w_i = 100\%$. Then, the w_i was multiplied to the return distribution of the agent i, resulting in portfolios with w weighted distribution. Each portfolio was evaluated by a *fitness function*, which in this case is the monthly financial return. We have considered it an interesting fitness measure because its calculation incorporates both the drawdown and the total profit. Consequently, the GA would look for a trade-off between risk and financial return. In addition to the usual GA setup, we have incorporated a restriction that limits the standard deviation of the weights. This would lead to more balanced weights, without too large or too little w values (that would resulting in removing strategies from the portfolio and increasing the chance of overfitting). The *crossover* and *mutation* rates were 92% and 8%, respectively. The algorithm have ran for a total of 30 *generations*, and the best individual was taken for testing.

VI. Experiments

In this section, we present the experimental validation of the trading agents and the agent portfolios trained via Genetic Algorithm. We have organized the results and analysis in 5 parts. In the first part, we explain *general considerations*, including the platform and data used to train the agents, the trading costs and the Brazilian benchmarks. In the second part,

we describe the *portfolios' generation*, giving details about the generation of 3 different portfolios. In the third part, we show *the agent patterns* obtained through GA. In the fourth part of the analysis, we discuss the *agents' and portfolios' performance without leverage*, in terms of profitability of the testing set. In the fifth and last part, we present the *agents' and portfolios' performance with leverage*. We compare the results of the 3 different portfolios with the single agent's performance and compare them to important Brazilian benchmarks.

A. General considerations

1) Platform: Most of the implementations and tests were done using the trading platform Metatrader 5 and its programming language called MQL5. They allow the development of trading robots, performing backtests and trading with a real account.

2) Dataset: The dataset where we have run the model, consisted of the Brazilian BM&F Mini Ibovespa Futures (WIN) and Mini Dollar Futures (WDO). Each WIN and WDO point corresponds to R$ 0.20 and R$ 10.00, respectively. They were chosen because of their high liquidity, financial return potential, low brokerage fees and high leverage allowed by the broker. The data contained candlesticks of different timeframes with volume, open, high, low and close prices. The data period chosen for the experiments went from 2013-10-18 to 2019-07-12. The data was split as following:

1) 2013-10-18 to 2017-01-01: Data set for training;
2) 2017-01-01 to 2018-01-01: Data set for validation;
3) 2018-01-01 to 2019-07-12: Data set for testing.

Those same periods were used both for training the Pattern Searcher model and training the portfolio weights.

3) Initial deposit and trade margin: For our analysis, we have considered an initial deposit of R$100,000.00 and a trading margin ($TradePct$) of 10%, which means that we are using R$10,000 for trade.

4) Trade costs: We have considered all the trading costs, totaling R$0.48 per contract. These costs were already applied in the experiments.

5) Brazilian Benchmarks: There are 2 important Brazilian benchmarks: SELIC (rate of risk-free return) and IBOV (floating income rate).

SELIC: is the Brazilian interest rate. It had an average annualized rate of 6.46% during the period between 2017-12-07 and 2019-07-31 [22].

IBOV: is the Brazilian stock market index. It had a growth of 48% from 2018-01-01 to 2019-07-12, that is approximately 31% per year, and had a maximum cumulative fall, or " drawdown", of 23%.

B. Portfolios' generation

We have used the agents generated by the Pattern Searcher model to create 3 strategy portfolios: Portfolio A, Portfolio B, and Portfolio C. In each of them, the weights were determined differently.

In the Portfolio A, the weights are equal, with each agent trading the same volume of contracts. In the Portfolio B, we have used the Genetic Algorithm without restricting the weight's standard deviation. In the Portfolio C we have optimized the weights such as the Portfolio B optimization, however with the restriction that limits the standard deviation of the weights. The Table I shows the ws distribution.

C. Agent patterns

We have generated 8 agents, 5 on WIN and 3 on WDO, using the Pattern Searcher model. Each of them received different parameters, indicators, timeframes, and symbols, intending to improve the diversity of the found patterns. The experiments were all focused on day trading, which means that all positions were opened and closed in the same day.

In order to show the outcome rules generated by the model, while not extending this section, Table II shows the parameters of Agent 1 only, and its generated patterns, expressed in form of Boolean rules.

D. Agents' and portfolios' performance without leverage

The Table III shows the agents' and portfolios' performance without considering any leverage.

We can see that all agents, except Agent 7, were profitable during the testing and had an average monthly income of around 1%. The best agent was Agent 5, with a monthly income of 2.72% and an annualized financial return of 38.56%. In this situation where we did not consider any leverage, the portfolios did not give any improvement in terms of monthly financial return, however, their drawdown was considerably smaller compared to the single agent's, providing lower risks.

6 out of 8 agents, and the 3 portfolios, surpassed the SELIC in terms of annualized financial return. However, only the Agent 5 overcame the IBOV, with 38.56% against 31% of annualized financial return. In terms of drawdown, the agent's and portfolio's drawdowns were smaller.

E. Agents' and portfolios' performance with leverage

To facilitate the comparison among the agent results, we attempted to normalize the results by equaling the risks of the agents, in a way that all agents have the same drawdown of the training set. We have used the $TradePct = 10\%$, and used the "controlled leverage" approach to calculate the number of contracts. Table IV shows the number of contracts of each agent and each portfolio, and Table V shows the test results using the distribution of contracts calculated previously.

1) Agent's performance: A total of 7 out of 8 agents had positive total net profit during the test period. The total net profit varied from R$ 14,328.01 to R$ 53,459.32, corresponding to a growth of 14.33% to 53.46%. The drawdown of the test stayed between 3,38% and 14.21%, deviating up to 40% from the training drawdown.

The Agent 7 was the only one that presented poor results. It's possible to see strategies that work well for a long time and suddenly stop to work. We could address the reason for it to the change of market behavior over time, possibly for external factors, such as economy, interest rate, and inflation. All the other agents are likely to have loss periods like Agent 7 at some moment. This reinforces the benefits of using a strategy portfolio.

TABLE I
PORTFOLIO'S WEIGHTS (w_s)

Weight	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	Total	Std	Description
Strategy Portfolio A	12.50%	12.50%	12.50%	12.50%	12.50%	12.50%	12.50%	12.50%	100.00%	0.00%	Equal w_s
Strategy Portfolio B	29.87%	12.81%	4.78%	18.82%	18.98%	3.33%	1.95%	9.46%	100.00%	9.62%	w_s trained via GA
Strategy Portfolio C	13.41%	9.56%	12.57%	12.71%	18.08%	10.28%	10.17%	13.22%	100.00%	2.71%	w_s trained via GA with std restriction

TABLE II
AGENT 1 PARAMETERS AND PATTERNS

Agent 1

Symbol:	WIN
Timeframe:	5 minutes
TP/SL type:	percentual of the entry price
TakeProfit:	0.08%
StopLoss:	0.008%
Order placement:	placed 3 ticks after the candle close price
Close position:	at the end of the day 120 minutes after the position was opened if inverse condition is satisfied

Pattern 1:

if -1.00<RSI(6)<0.52 **and** 0.26<ADX(14)<0.30 **and**
0.33<PvtR3()<1.00 **and** 0.22<STO(14)<1.00 **then** Sell

Pattern 2:

if -1.00<RSI(6)<-0.57 **and** -1.00<MA(28)<-0.80 **then** Buy

2) Portfolio's performance: The portfolios A, B, and C presented a total net profit of R$ 115,044.55, R$ 138,983.47 and R$ 137,468.44, and an annualized financial return of 108.09% 139.96% and 138.46%, respectively. All portfolios managed to more than double the initial deposit in a period of 1 year. The drawdown went from 6.58% to 10.36%, which is coherent with the 10% drawdown of the training set.

The way the portfolio's weights were determined significantly implicated in the results. The Portfolio A had great profitably even with no training. The training using GA improved further the financial returns of the Portfolios B and C. The Portfolio B has a slightly higher monthly income than the Portfolio C, 7.46% against 7.40%, however, the Portfolio C is better since its drawdown was only 6.60%, 36% smaller than the Portfolio B drawdown. The reason for that is the possibility of having occurred some overfitting of the agent B. Its weights assumed an unbalanced distribution, with a very high volume of contracts in the Agent 1 and only a very few in the Agents 6 and 7 (see Table I). The effect of this is similar to removing agents from the portfolio and relying only on a few.

On the other hand, restricting the standard deviation of the weights of the Portfolio C resulted in a soft optimization with weights ending up having close values from each other. This soft optimization was enough to increase the performance of the portfolio while maintaining its robustness on unseen data.

3) Agent's performance vs Portfolio's performance: The portfolios had a considerably higher performance compared to the single agents' performance. The portfolio's highest annualized return was 3 times greater than the highest single agent's annualized financial return, 139.96% against 42.25%. This wide difference is mostly due to the drastic risk decrease that a portfolio can yield. The decrease of the drawdown opens room for increasing the leverage, and consequently, achieving higher profitability for the same amount of risk.

4) Comparison with Benchmarks: 6 out of 8 agents, and the 3 portfolios, overcame the SELIC rate in terms of annualized financial return. Only 2 agents (Agent 5 and 6) were able to beat the IBOV, with 41.25% and 40.28%, respectively, against 31% of annualized financial return. On the other hand, all of the 3 portfolios surpassed the IBOV with considerable advantage. The Portfolio C, which we have considered as being the best among all portfolios, had an annualized financial return of 138.46%, which is over 4 times greater than the IBOV's. Also, the Portfolio C's drawdown was 6.60%, which is around 3.5 times smaller than the IBOV's cumulative fall of 23%.

VII. CONCLUSION

This paper proposed a model that automatically searches for profitable trading patterns, named Pattern Searcher, and generated strategy portfolios, both trained via GA. As a result, 7 out of 8 agents generated through the Pattern Searcher model were very profitable.

The creation of portfolios considerably improved the profitability compared to single agents', outperforming the Brazilian benchmarks. The best portfolio achieved an annualized financial return of 138.46% and a drawdown of 6,56%, With this return, it would more than double the initial deposit in one year.

The system showed some desirable properties. One of them is the generation of rules than can be easily interpreted by humans, allowing the investor to accept or reject the strategy according to his knowledge. Another desirable characteristic is the simplification of the system flow compared to some Machine Learning methods. The model and the strategy are interconnected, in a way that the training directly optimizes the agent performance in terms of profitability.

Although the results seem promising, it is important to have in mind that past results are not a guarantee of future results. The stock market may change its behavior over time, and the agent's strategy might lose effectiveness.

Future work would consist of testing the robustness and consistency of the GA agents, and applying the method in other datasets, including international financial markets (e.g., NYSE, NASDAQ, Tokyo), cryptocurrencies and forex. In addition, we would extend the approach to agents that trade multiple assets simultaneously, rather than specific assets. According to [16], this can help to prevent overfitting, resulting in more robust systems.

ACKNOWLEDGMENTS

This work was partially funded by the Brazilian National Institute of Science and Technology for the Web (grant no.

TABLE III
AGENT'S AND PORTFOLIO'S RESULTS WITHOUT LEVERAGE

Agent	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8	Portfolio A	Portfolio B	Portfolio C
Initial Deposit	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00
Contracts	5.00	5.00	5.00	5.00	5.00	2.86	2.86	2.86	3.90	4.50	3.99
Gross Profit	R$ 70,910.01	R$ 49,750.35	R$ 63,374.32	R$ 47,442.65	R$ 111,861.63	R$ 21,730.72	R$ 28,774.65	R$ 27,776.29	R$ 29,998.35	R$ 42,565.98	R$ 32,869.77
Gross Loss	-R$ 50,992.32	-R$ 26,997.03	-R$ 37,683.19	-R$ 37,104.41	-R$ 61,417.03	-R$ 11,386.55	-R$ 28,806.10	-R$ 19,940.64	-R$ 14,303.44	-R$ 21,396.58	-R$ 15,165.00
Total Net Profit	R$ 19,917.68	R$ 22,753.33	R$ 25,691.13	R$ 10,338.24	R$ 50,444.59	R$ 10,344.17	-R$ 31.46	R$ 7,835.66	R$ 15,694.92	R$ 21,169.40	R$ 17,704.77
Total Financial Return	19.91%	22.75%	25.69%	10.34%	50,44%	10.34%	0.00%	7.84%	15.69%	21.16%	17.70%
Drawdown	R$ -7,131.67	R$ -5,184.22	R$ -3,216.35	R$ -4,728.66	R$ -4,451.29	R$ -1,037.95	R$ -7,283.18	R$ -2,011.67	R$ -895.19	R$ -1,583.43	R$ -848.93
Drawdown (%)	-7.13%	-5,18%	-3,21%	-4,72%	-4,45%	-1,03%	-7,28%	-2,01%	-0,90%	-1,58%	-0,85%
Profit per Month	R$ 1,072.77	R$ 1,225.49	R$ 1,383.72	R$ 556.82	R$ 2,716.94	R$ 557.14	-R$ 1.69	R$ 422.03	R$ 845.33	R$ 1,140.18	R$ 953.58
Monthly Return	1.07%	1.23%	1.38%	0.56%	2.72%	0.56%	0.00%	0.42%	0.85%	1.14%	0.95%
Annualized Return	13.86%	15.97%	18.20%	6.99%	38.56%	6.99%	-0.02%	5.26%	10.78%	14.79%	12.24%

TABLE IV
ESTIMATION OF THE NUMBER OF CONTRACTS PER AGENT

Agent	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8	Portfolio A	Portfolio B	Portfolio C
Balance Drawdown (from training set)	R$ 1,004.32	R$ 1,213.39	R$ 1,901.23	R$ 1,443.08	R$ 1,887.21	R$ 690.52	R$ 700.88	R$ 1,079.81	R$ 348.92	R$ 339.51	R$ 322.62
Balance Needed (from training set)	R$ 10,043.18	R$ 12,133.86	R$ 19,012.30	R$ 14,430.81	R$ 18,872.14	R$ 6,905.23	R$ 7,008.81	R$ 10,798.10	R$ 3,489.24	R$ 3,395.10	R$ 3,226.20
Contracts to make DD = 10%*Deposit	9.96	8.24	5.26	6.93	5.30	14.48	14.27	9.26	28.66	29.45	31.00

TABLE V
AGENT'S AND PORTFOLIO'S RESULTS WITH LEVERAGE

Trading Agents	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8	Portfolio A	Portfolio B	Portfolio C
Initial Deposit	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00
Contracts per Entry	9.96	8.24	5.26	6.93	5.30	14.48	14.27	9.26	28.66	29.45	31.00
Gross Profit	R$ 141,210.23	R$ 82,002.53	R$ 66,666.65	R$ 65,751.88	R$ 118,546.83	R$ 110,034.78	R$ 143,548.75	R$ 89,941.64	R$ 220,308.34	R$ 278,480.57	R$ 255,216.85
Gross Loss	-R$ 101,546.14	-R$ 44,498.67	-R$ 39,640.85	-R$ 51,423.87	-R$ 65,087.51	-R$ 57,656.46	-R$ 143,705.68	-R$ 64,569.22	-R$ 105,044.65	-R$ 139,983.45	-R$ 117,748.44
Total Net Profit	R$ 39,664.08	R$ 37,503.86	R$ 27,025.80	R$ 14,328.01	R$ 53,459.32	R$ 52,378.32	-R$ 156.93	R$ 25,372.42	R$ 115,263.70	R$ 138,497.12	R$ 137,468.41
Total Financial Return	39.66%	37.50%	27.03%	14.33%	53.46%	52.38%	- 0.16%	25.37%	115.26%	138.50%	137.47%
Profit Factor	1.39	1.84	1.68	1.28	1.82	1.91	1.00	1.39	2.10	1.99	2.17
Drawdown	-R$ 14,206.30	-R$ 8,543.60	-R$ 3,383.60	-R$ 6,553.92	-R$ 4,718.37	-R$ 5,255.08	-R$ 36,339.52	-R$ 6,513.31	-R$ 6,578.52	-R$ 10,362.66	-R$ 6,595.70
Drawdown (%)	- 14,21%	- 8,54%	- 3,38%	- 6,55%	- 4,72%	- 5,26%	- 36,34%	- 6,51%	- 6,58%	- 10,36%	- 6,60%
Entries	314	163	272	252	531	170	144	130	1976	1976	1976
Entry Costs	R$ 1,500.72	R$ 644.81	R$ 686.71	R$ 838.21	R$ 1,350.56	R$ 1,181.71	R$ 986.19	R$ 577.88	R$ 3,397.88	R$ 4,194.16	R$ 4,005.61
Leverage	1.99	1.65	1.05	1.39	1.06	5.06	4.99	3.24	7.34	6.54	7.76
Days with Entry	215	120	188	176	282	149	139	124	368	368	368
Profit per Month	R$ 2,136.31	R$ 2,019.96	R$ 1,455.61	R$ 771.71	R$ 2,879.32	R$ 2,821.09	-R$ 8.45	R$ 1,366.56	R$ 6,208.10	R$ 7,459.45	R$ 7,404.04
Success Rate	54.88%	60.00%	57.45%	54.55%	53.90%	69.13%	51.80%	53.23%	57.61%	56.79%	58.97%
Std. Dev. of the returns	1100.14	759.29	522.44	566.01	708.49	755.42	1374.65	805.30	1098.57	1462.34	1256.46
Monthly Return (%)	2.14%	2.02%	1.46%	0.77%	2.88%	2.82%	-0.01%	1.37%	6.21%	7.46%	7.40%
Annualized Return	29.33%	27.55%	19.22%	9.80%	41.25%	40.28%	-0.10%	17.96%	108.09%	139.96%	138.46%

573871/2008-6), MASWeb (grant FAPEMIG/PRONEX APQ-01400-14), CAPES, CNPq (grant 459301/2014-4), Finep, and Fapemig.

REFERENCES

[1] Investopedia, "Technical analysis." https://www.investopedia.com/terms/t/technicalanalysis.asp, 2018. [Online; accessed 25-November-2018].

[2] C. M. Investopedia, "Moving averages: Strategies." https://www.investopedia.com/university/movingaverage/movingaverages4.asp, 2018. [Online; accessed 25-November-2018].

[3] J. R. Koza, "Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems," 06 1990.

[4] A. Padua Braga, *Redes neurais artificiais: teoria e aplicaes*. LTC Editora, 2007.

[5] D. Faustryjak, L. Jackowska-Strumillo, and M. Majchrowicz, "Forward forecast of stock prices using lstm neural networks with statistical analysis of published messages," pp. 288–292, May 2018.

[6] S. Liu, G. Liao, and Y. Ding, "Stock transaction prediction modeling and analysis based on lstm," pp. 2787–2790, May 2018.

[7] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 2 nd ed., 2018.

[8] J. Moody and M. Saffell, "Learning to trade via direct reinforcement," *IEEE Transactions on Neural Networks*, vol. 12, pp. 875–889, jul 2001.

[9] P. Kroha and M. Friedrich, "Comparison of genetic algorithms for trading strategies," 01 2014.

[10] M. A. Dempster, T. W. Payne, Y. Romahi, and G. W. Thompson, "Computational learning techniques for intraday fx trading using popular technical indicators," *Trans. Neur. Netw.*, vol. 12, pp. 744–754, July 2001.

[11] A. Pimenta, F. G. Guimarães, E. G. Carrano, C. A. L. Nametala, and R. H. C. Takahashi, "Goldminer: A genetic programming based algorithm applied to brazilian stock market," in *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pp. 397–402, Dec 2014.

[12] D. Iskrich and D. Grigoriev, "Generating long-term trading system rules using a genetic algorithm based on analyzing historical data," in *2017 20th Conference of Open Innovations Association (FRUCT)*, pp. 91–97, April 2017.

[13] E. Tsang, P. Yung, and J. Li, "Eddie-automation, a decision support tool for financial forecasting," *Decis. Support Syst.*, vol. 37, pp. 559–565, Sept. 2004.

[14] M. Kampouridis and E. Tsang, "Eddie for investment opportunities forecasting: Extending the search space of the gp," in *IEEE Congress on Evolutionary Computation*, pp. 1–8, July 2010.

[15] Y. Hu, K. Liu, X. Zhang, L. Su, E. Ngai, and M. Liu, "Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review," *Applied Soft Computing*, vol. 36, 07 2015.

[16] M. L. de Prado, *Advances in Financial Machine Learning*. Wiley Publishing, 1st ed., 2018.

[17] D. Bailey, J. J. Borwein, M. Lopez de Prado, and Q. Zhu, "Pseudo-mathematics and financial charlatanism: The effects of backtest over-fitting on out-of-sample performance," *Notices of the American Mathematical Society*, vol. 61, p. 458, 05 2014.

[18] M. S. Corp, "Trading strategy tester: Test and optimize your trading robot before you use it for real trading." https://www.metatrader5.com/en/automated-trading/strategy-tester, 2020.

[19] M. S. Corp, "Optimization types." https://www.metatrader5.com/en/terminal/help/algotrading/optimization_types.

[20] A. Hayes, "Leverage." https://www.investopedia.com/terms/l/leverage.asp, 2019. [Online; accessed 07-January-2020].

[21] J. CHEN, "What is a portfolio?." https://www.investopedia.com/terms/p/portfolio.asp, 2019.

[22] BancoCentral, "Taxas de juros básicas." https://www.metatrader5.com/en/automated-trading/strategy-tester, 2020.