Generating Long-Term Trading System Rules Using a Genetic Algorithm Based on Analyzing Historical Data

Dmitry Iskrich, Dmitry Grigoriev
Saint-Petersburg State University
Saint-Petersburg, Russia
st034749@student.spbu.ru, d.a.grigoriev@spbu.ru

***Abstract*—In current times, trading success depends on choosing a correct strategy. Algorithmic trading is often based on technical analysis - an approach where the values of one or several technical indicators are translated into buy or sell signals. Thus, every trader's main challenge is the choice and use of the most fitting trading rules. In our work, we suggest an evolutionary algorithm for generating and selecting the most fitting trading rules for interday trading, which are presented in the form of binary decision trees. A distinctive feature of this approach is the interpretation of the evaluation of the current state of technical indicators with the help of dynamic ranges that are recalculated on a daily basis. This allows to create long-term trading rules. We demonstrate the effectiveness of this system for the Top-5 stocks of the United States IT sector and discuss the ways to improve it.**

I. Introduction

Today, many investors' success is a result of the correct choice of strategy. There are numerous factors that influence the dynamics of the stock market. In stock trading, two approaches are used for predicting the price of a certain stock: technical analysis and fundamental analysis. In fundamental analysis, the indicators of most importance are dividend policy, enterprise value, and profit and sales values. The technical analysis, in its turn, is based on analyzing the behavior of the price of the asset and calculating additional indicators, which, when following certain patterns, help in identifying the direction of the current trend. Surely, a trader should be guided by a number of technical indicators. Combinations of the indicator and the corresponding value can create numerous buy/sell rules. Considering that, the trader can make assumptions concerning the asset's prospects [1].

Identifying these patterns is a nontrivial problem. AI methods are frequently used in development of an optimal trading strategy. They can either optimize pre-determined strategies or generate new ones and optimize them later. There is a large number of studies that present a method for determining the winning strategy that is based on neural networks [2], [3]. In paper [4], using Bayesian networks is suggested as a basis for trading decision making. The agent approach is also quite popular [5], [6]. It introduces a domain of decision-making agents. For example, in paper [7] the agents are considered to be a population, and the best agent is selected by a genetic algorithm. One of the most convenient representations of a trading strategy is binary decision trees [8], [9], [10]. After the best tree is determined, it can be used as an adviser for making decisions regarding the current market situation. Methods that employ this data structure are widely used [11], [12]. In particular, in reference [10] decision trees are used to classify the current market action (BUY, HOLD, SELL). This approach uses immutable decision trees, and it is bound by specific indicators. In turn, in the system presented in reference [13], an evolutionary algorithm constructs a binary decision tree based on mathematical functions, constants, and technical indicators. Despite this approach's good adaptivity to the current market environment, its focus on constant values limits the time of result validity to short-term.

In our study we use a similar approach to making trading decisions. A genetic algorithm constructs an optimal decision tree based on historical data. This tree can be later translated into trade system rules. In this case, the tree is represented by a set of rules that depict the current market environment. In contrast to references [10] and [13], technical indicator states are classified on the basis of their dynamic ranges. This allows to extend the time of rule validity to long-term. However, we had to modify the structure of the decision tree.

A more detailed description of the structure of the tree can be found in Section II. Section III contains the description of the steps of the genetic algorithm for constructing a set of optimal decision trees. The principles of choosing the best individuals are listed in Section IV, and the analysis of their efficiency is presented in Section V.

II. Decision tree

A decision tree is a directed acyclic graph whose leaf nodes contain decisions. In this case, they can be buy or sell signals. The process of determining the decision starts from the root of the tree. Depending on the node's value, there is a transition either to its right or left child. As a result, a leaf at the end is taken. This strategy representation can be convenient in the translation of certain trading systems into finite rules, because this way buy and sell signals can be reduced to a conjunctive normal form. The problem of (finding) the best binary decision tree is an NP-complete problem [14], which is why heuristics are used for finding a solution.

The node of the tree is a predicate that determines transition to one of its children. In every non-terminal node, the value of a technical indicator is compared to a certain level to see whether

they match. Indicators most commonly have real values, but their current state can be associated with a certain level. This approach was introduced in paper [15]. In other words, at a given time, a technical indicator can have one value from the enumeration: {*VERY LOW, LOW, MEDIUM, HIGH, VERY HIGH*}. In Fig. 1 you can see an example of decision tree.

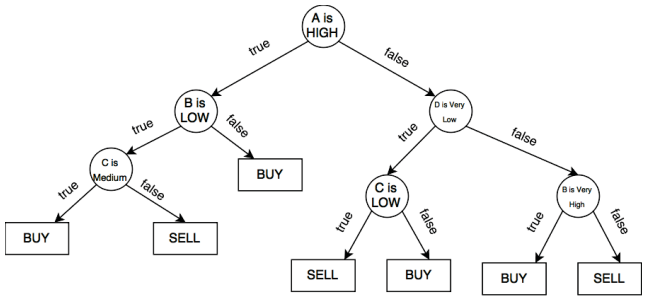

Fig. 1. Simple decision tree

The tree can also be presented as a trivial algorithm:

Algorithm 1 Simple trading algorithm

```
if A is HIGH then
  if B is LOW then
    if C is MEDIUM then
      BUY
    else
      SELL
    end if
  else
    BUY
  end if
else
  if D is VERY LOW then
    if C is LOW then
      SELL
    else
      BUY
    end if
  else
    if B is VERY HIGH then
      BUY
    else
      SELL
    end if
  end if
end if
```

After every iteration of this algorithm either a BUY or SELL signal will be received. However, the actual trading action (Buy, Sell, Hold) depends on the direction of the current position. Transitions between the corresponding position states are defined in Table I.

Note that we do not use "HOLD" as a signal. Generally, its presence reduces the number of trades and, thereby, lowers the trading fee. However, this way the time of position hold increases, and, consequently, so do the financial risks. In our work, we use a simplified algorithm, and we focus on exiting the position as fast as possible.

TABLE I. TRADING ACTION ON SIGNAL

Current position	Signal	Next position	Trading action
Out of position	BUY	Long	Buy stock
Out of position	SELL	Short	Sell stock
Long	BUY	Long	Hold
Long	SELL	Out of position	Sell stock
Short	BUY	Out of position	Buy stock
Short	SELL	Short	Hold

III. GENETIC ALGORITHM

We use a genetic algorithm for determining the best tree. All evolutionary steps are involved.

A. Initialization

The initial population is generated randomly with a preset offspring size. Every node is constructed by selecting a random indicator and associating it with a random level. After that, its offspring is created and so on. The height of the tree is determined by the number of indicators available. After this height has been reached, generation of non-terminal nodes is halted by creation of random leaf nodes.

B. Selection

The individuals that fit the most are selected for the so-called next generation, which is created with the purpose of further tree modification by genetic operators. This selection is performed via roulette wheel selection. For the each tree the probability of being chosen is:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}, \tag{1}$$

where f_i is the fitness function of the T_i tree.

C. Fitness function

The objective function is calculated via trading simulation using historical data. It presents the profit margin as a percentage.

$$f_i = (\frac{FinalBalance}{StartBalance} - 1) * 100, \tag{2}$$

where $StartBalance$ is the amount of money in the beginning of the simulation, and $FinalBalance$ is the amount of money at the end. It should be noted, that total stock return, except any dividends paid, has the same value. Therefore, we make assumptions that f_i and return are equal.

D. Genetic Operators

Here, the genetic operators are constructed by analogy with the paper [13]. They include the crossover operator and the mutation operator. Because of these two operations, the principles of inheritance and self-adaptivity can be achieved. Considering the specifics of tree data structures, these operations are based upon both terminal and non-terminal node manipulations.

The crossover operator takes an input of two parent trees that are chosen via roulette wheel selection with the probability determined by the formula (1), and outputs two offspring trees that are members of the next generation. These offspring trees, in their turn, are created by choosing a random node in both

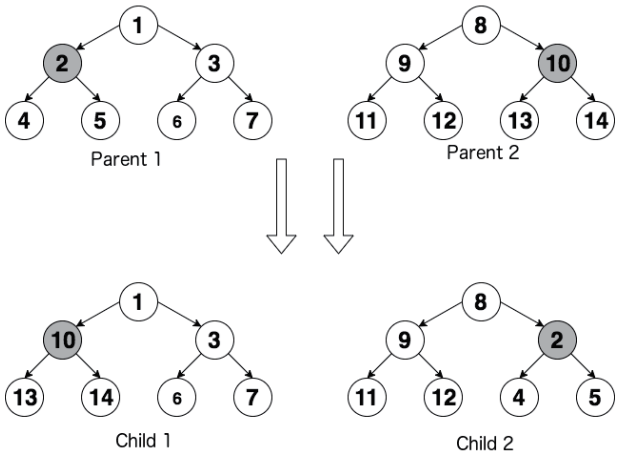

Fig. 2. Crossover operator

parent trees and swapping subtrees starting from these nodes. The mutation operator is a unary operator. It is intended for protecting the generation (presented as a set of trees) from premature convergence. It is based on the same approach as the crossover operator, which is selecting and swapping two random nodes in the tree. It is worth noting that the nodes do not have to be of the same type.

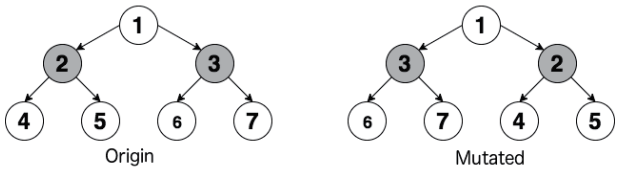

Fig. 3. Mutation operator

Removing excess nodes is an important step after using these operators. In terms of mathematical logic it can be interpreted as removal of redundant disjuncts. A typical example of such redundancy is presented in Fig. 4. When visiting node *IND1 is LOW*, we already know that the predicate is false, because we have transitioned to this node from *IND1 is HIGH*. Excess node removal is performed via replacing nodes. In this case, the true-way of the root node will link to the false-way of the *IND1 is LOW* node. After creating a new generation, the excess node removal algorithm 2 is run for each tree that had been modified by crossover or mutation. As the result, a tree either retains its structure, or, if an unnecessary predicate is found in a node, nodes are replaced according to the algorithm below. The output is an optimized tree that works the same way as the original one.

Algorithm 2 Tree optimization algorithm

Input: T - decision tree
Output: $Optimized\ tree$
for $node$ in T **do**
for rch in $node.RightChild$ **do**
if ($rch.Index$ = $node.Index$) **then**
if ($rch.Value$ = $node.value$) **then**
rch = $rch.Right$
else
rch = $rch.Left$
end if
end if
end for
for lch in $node.LeftChild$ **do**
if ($lch.Index$ = $node.Index$
and $lch.Value$ = $node.Value$) **then**
lch = $rch.Left$
end if
end for
end for
return T

This algorithm is based on traversing the tree and searching for nodes that do not influence the result. In this case $Index$ is the name of the indicator, $Value$ is its value, $LeftChild$ and $RightChild$ are its offsprings.

This kind of an algorithm may seem quite time-consuming at first, but in the process of calculating the fitness function (the trading simulation), calculating redundant predicates can drastically hinder the performance.

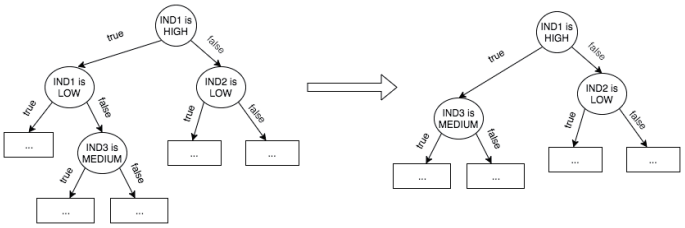

Fig. 4. Optimization process

E. General algorithm

Considering the aforementioned factors, we can now construct a general algorithm:

- Step 1. Determining the parameters of the genetic algorithm: n is the population size, m - population part intended for the mutation operator, l - population part intended for the crossover operator, p - the number of generations.
- Step 2. Calculating the ranges for dividing indicator values into levels (Subsection IV-F).
- Step 3. Generating the initial population. Calculation of the objective function f_i by the formula (2) for each individual.
  $CurrentGen = 0$

- Step 4. Creating a new generation. Selecting the survivors via roulette wheel selection. $CurrentGen = CurrentGen + 1$
- Step 5. Creating the offspring. Selecting the parents via roulette wheel selection and generating the offspring by the crossover operator. The number of parents cannot exceed $l\%$ from the population.
- Step 6. Application of the mutation operator to randomly selected $m\%$ of the population.
- Step 7. Calculating the fitness function f_i for every individual by formula (2).
- Step 8. Optimizing the trees obtained with crossover and mutation. (Algorithm 2).
- Repeat steps 4-8 while $CurrentGen < p$.

The purpose of the developed genetic algorithm is finding the best trading strategies for detecting the most efficient one at the training stage. For the sake of simplicity, our system analyses market close results and generates signals for the next trading day. In other words, every day after the trading session has closed the system decides whether to buy, sell or do nothing with the stock.

IV. Training

During training the system receives an input of trading decision trees from the last generation of the general genetic algorithm, evaluates their performance by certain metrics and identifies the most efficient trading strategy for further forward-testing. The best trading system will be selected for each stock.

A. Input Data

As the object of our research we have chosen stocks of the five largest IT companies (Table II). The training dataset includes the price data for the last 9 years (2007-2015). Therefore, the selection of the best-trained trees was performed on a set of 2268 signals.

TABLE II. IT companies

Company name	Revenue (billions)	Symbol
Apple Inc.	$233.7	AAPL
Alphabet Inc.	$74.99	GOOG
Amazon.com	$107.0	AMZN
Microsoft	$93.58	MSFT
IBM	$82.46	IBM

B. Parameter settings

The main parameters of a genetic algorithm are the number of generations and the number of individuals in a single generation. As these parameters increase, the quality of the decision improves, but the total working time grows significantly. Thus, it is necessary to select the optimal parameters in terms of the return to working time ratio.

Average returns for each of the 5 stocks and working time for 100, 200, 250 and 300 individuals are listed in Table III. As you can see, the number of individuals being higher than 200 brings about minor fluctuations of returns. However, the working time is increased significantly. In turn, Table IV lists the experiments that used 25 generations for determining the best indicator in terms of profit. Thus, the best decision is reached in less than 20 generations in the majority of cases. System parameters are listed in Table V.

TABLE III. Population metrics

Size	Average returns	Average time
100	23.8%	177s
200	35.6 %	358s
250	36.2	452s
300	35.0	546s

TABLE IV. Generations with best returns

Instrument	Generation number
AMZN	19
AAPL	16
IBM	14
MSFT	21
GOOG	19

It should be noted that the most labor-intensive operations are the excess nodes removal algorithm (Algorithm 2) and the indicator level calculation (Subsection IV-F). The time spent downloading the price data and computing the statistics is not included in the measurements.

C. Hardware and Software Setups

The setup used in our experiments:

- Python 2.7
- Intel® Core™ i5 2700 Mhz
- OS X Yosemite 10.10
- 8GB (DDR3) RAM

We used the pandas module for receiving, storing and processing price data [16]. We also used the NumPy extension for calculating indicator values since it is optimized for vector computations [17].

D. Metrics

In addition to the main metric that is the return value, the following metrics have been introduced for analyzing the results:

1) Sharpe ratio is the indicator of strategy efficiency, introduced by W.F. Sharpe [18]. It assesses the ratio between profit and possible risk. It is calculated as follows:

$$SharpeRatio = \frac{r_p - R_f}{\delta(x)}, \quad (3)$$

where r_p stands for average earnings, R_f is a risk-free rate (e.g. a bank deposit) and $\delta(x)$ is the standard deviation of the earnings.

TABLE V. Parameter settings

Parameter	Value
Population size	200
Number of generations	20
Crossover rate	0.35
Mutation rate	0.05

2) Maximum drawdown is the maximum balance decrease from its local maximum, expressed as a percentage. It allows to estimate the maximum capital loss when working with a given strategy.
3) Profit/loss trades ratio.
4) Returns as a percentage is calculated by formula (2).

E. Indicators

Below we describe the indicators that the system uses for its trading recommendations:

1) Relative strength index (RSI). This oscillator compares the number of recent increases of the price of the asset with the number of its decreases and presents this information as a number from 0 to 100. It is calculated as follows:

$$RSI = 100 - \frac{100}{1 + U/D}, \quad (4)$$

where U is the number of positive price changes, and D is the number of negative price changes during the considered period .
2) Stochastic oscillator (STO) shows a state of the current price relative to the price range of a certain period. It is a percentage which is calculated according to the formula:

$$STO_t = \frac{C_t - L_n}{H_n - L_n} \cdot 100, \quad (5)$$

where C_t is the current price, L_n is the lowest price for the past n periods, and H_n is the highest price for the past n periods.
3) Chaikin oscillator (CHO). It helps predict the behavior of the Accumulation/Distribution indicator. This indicator takes into account changes of price and trading volume. At first, the Accumulation/Distribution (A/D) indicator is calculated:

$$A/D_i = \frac{(Close - Low) - (High - Close)}{High - Low} \times \\ \times Volume - A/D_{i-1}, \quad (6)$$

where A/D_i - current day A/D value,
$Close$ - closing price,
Low - minimum day price,
$High$ - maximum day price,
$Volume$ - trading volume for this day,
A/D_{i-1} - A/D indicator value for the previous day.

After that, the value of the oscillator is:

$$CHO = EMA(A/D, 3) - EMA(A/D, 10), \quad (7)$$

where EMA(A/D, n) is a n-periodic exponential moving average of the A/D indicator.

F. Indicator levels

As mentioned earlier, the numerical values of the indicator are divided into 5 levels from $VERYLOW$ to $VERYHIGH$. Ranges of data levels are calculated using sliding time intervals. Each trading day d previous trading days are reviewed and maximum and minimum values of a given indicator are identified. These values determine the boundaries of the general range. Further, this range is divided into 5 non-intersecting parts of the same size that are assigned the names: *VERY LOW, LOW, MEDIUM, HIGH, VERY HIGH*. The depth of the sliding time interval d identifies that period in the past during which the dynamics of indicator behavior are still essential for the current day. This depth is chosen empirically, based on the specifics of the behavior of price of a particular asset. For the sake of simplicity, in our study d is the duration of training (in this case, 2268 trading days). It is obvious that range calculation is separate for each indicator. For example, the AMZN indicator ranges of the 251st trading day of 2016 are presented in Tables VI, VII, VIII.

TABLE VI. RSI RANGES

Level	Range
VERY LOW	< 0.200
LOW	$[0.200; 32.562)$
MEDIUM	$[32.562; 64.924)$
HIGH	$[64.924; 97.286)$
VERY HIGH	≥ 97.286

TABLE VII. STOCHASTIC RANGES

Level	Range
VERY LOW	< 4.149
LOW	$[4.149; 36.078)$
MEDIUM	$[36.078; 68.007)$
HIGH	$[68.007; 99.937)$
VERY HIGH	≥ 99.937

TABLE VIII. CHAIKIN RANGES

Level	Range
VERY LOW	< -49.076M
LOW	$[-49.076$M$; 252.943$M$)$
MEDIUM	$[252.943$M$; 554.962$M$)$
HIGH	$[554.962$M$; 856.981$M$)$
VERY HIGH	≥ 856.981M

G. Output

After the end of the training process, we have obtained the most efficient strategies for each stock. In every case the Sharpe ratio is positive, except for AAPL, i.e. the profitability of these strategies exceeds the risk-free variant. In two particular cases the Sharpe ratio is greater than one, which shows the high efficiency of the investment portfolio (Fig. 5).

Max drawdown value stays within 10-30% with the exception of GOOG, for which it amounts to 55% (Fig. 6).

The profit/loss trades ratio is presented in Table IX.

TABLE IX. PROFIT/LOSS TRADES RATIO AFTER TRAINING

Symbol	Profit trades	Loss trades
AAPL	51.46%	48.54%
GOOG	57.83%	42.17%
AMZN	60.10%	39.90%
MSFT	60.05%	39.95%
IBM	54.79%	45.21%

In particular, the highest average number of profitable trades was reached using the AMZN trading strategy presented in Fig. 7. Total profit has also shown positive results. Return values of the stocks are shown in Fig. 8.

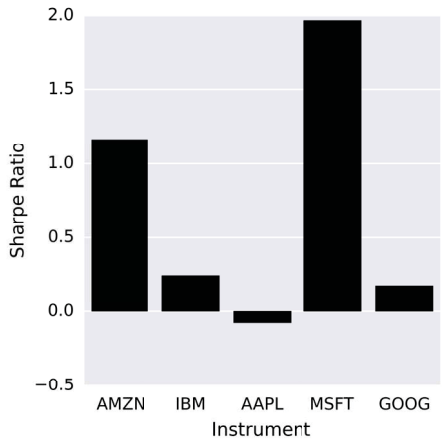

Fig. 5. Sharpe ratio after training

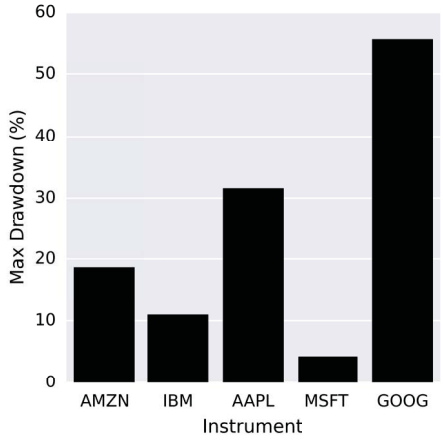

Fig. 6. Max drawdown after training

V. Forward testing

Forward testing is designed for a rough estimate of the behavior of a formed strategy in real life. After each stock has had the fittest trading strategy that was trained with the series of prices from 2007-2015 selected for it, the strategy is then tested on price data of the whole year of 2016. The results and the profit differences are shown in Table X.

TABLE X. Profitability for the untrained period

Symbol	Returns	Max drawdown	Difference with training
AAPL	11.1403%	6.6287	0.1058%
GOOG	3.7996%	5.2853	-53.3520%
AMZN	29.4216%	3.2461	10.7826%
MSFT	3.8532%	5.1864	-29.0845%
IBM	11.1735%	6.8542	-61.7575%

In the majority of cases "virtual" profitability degrades dramatically. However, the strategies remained efficient. This factor is represented by the value of the Sharpe ratio (Fig. 9). Despite its worsening, in all cases it exceeds the rate of risk-free return. Maximum drawdown values stayed within 7%.

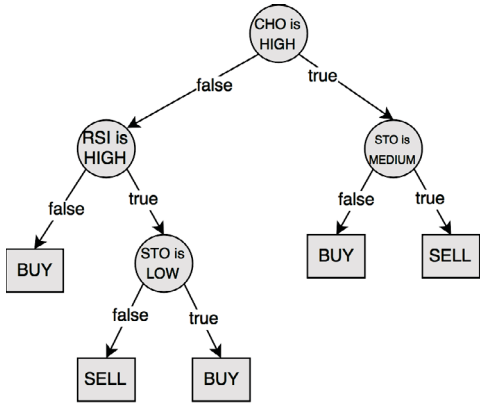

Fig. 7. Sample profitable AMZN trading strategy

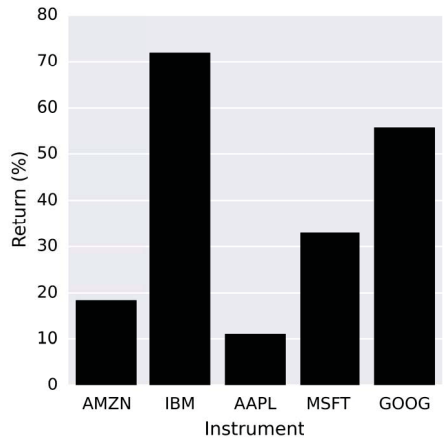

Fig. 8. Return values after training

VI. Conclusion and Future Work

The advantages of our approach are:

1) The simplicity of the generated rules. Representing a strategy as an understandable decision making tree provides the simplicity of the realization of mechanical trading systems. The resulting trees can be interpreted as either logical formulae or rules for a trading terminal.
2) Extensibility. The predicates in the rules do not depend on absolute values, interpreting the real value of an indicator as a level relative to the current state (from $VERY\,LOW$ to $VERY\,HIGH$). Similar rules can be applied to any indicators and datasets. Most approaches so far do not possess this quality.

For a variety of instruments the presented algorithm shows a low annual rate. It is obvious that the efficiency of the constructed trading strategies depends on the set of technical indicators and their parameters being used. It is necessary to automate the selection of optimal indicator parameters, and then research the applicability of other technical indicators.

There is also the problem of selecting the parameters of the genetic algorithm itself, since right now they are chosen

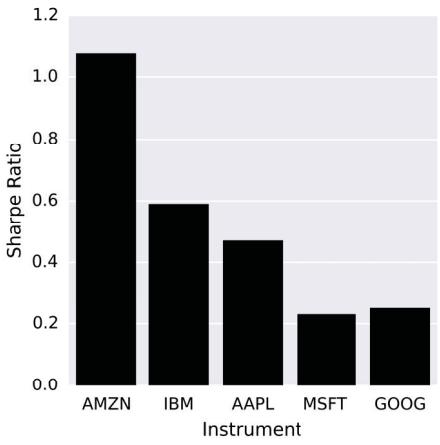

Fig. 9. Sharpe ratio for the untrained period

empirically. Moreover, it is necessary to investigate the influence of the fitness function selection on profit increase and risk decrease of the resulting trading strategy.

REFERENCES

[1] M.J John, "Technical analysis of the financial markets: a comprehensive guide to trading methods and applications (2nd ed.)", New York Institute of Finance, 1999.

[2] K. Bergerson D. C. Wunsch II. "A Commodity Trading Model based on a Neural Network-Expert System Hybrid" , Neural Trading Company, 1991.

[3] B.K. Wong and Y. Selvi, "Neural network applications in finance: A review and analysis of literatures (1990-1996)", *Information & Management*, Vol. 34, 1998.

[4] M. Bendtsen, and J. M. Peña, "Learning gated Bayesian networks for algorithmic trading", *European Workshop on Probabilistic Graphical Models*, Springer International Publishing, 2016, pp. 49-64.

[5] J. Korczak, M. Hernes and M. Bac, "Fundamental analysis in the multi-agent trading system", *Computer Science and Information Systems (FedCSIS)*, Federated Conference on pp. 1169-1174, 2016.

[6] S.J. Leal, M. Napoletano, A. Roventini and G. Fagiolo, "Rock around the clock: An agent-based model of low-and high-frequency trading", *Journal of Evolutionary Economics*, pp. 49-76, 2016.

[7] C. Schoreels, B. Logan, J.M. Garibaldi, "Agent based genetic algorithm employing financial technical analysis for making trading decisions using historical equity market data", *International Conference on Intelligent Agent Technology*, 2004.

[8] B.A. Sheldon, "Binary Decision Diagrams", *IEEE Transactions on Computers*, pp. 509–516, 1978.

[9] L. Rokach and O. Maimon, "Data mining with decision trees: theory and applications", 2014.

[10] F. X. Satriyo D. Nugroho, T. Bharata Adji, S. Fauziati, "Decision support system for stock trading using multiple indicators decision tree", *The 1st International Conference on Information Technology, Computer, and Electrical Engineering*, 2014.

[11] C. N. Ochotorena, C. A. Yap , E. Dadios and E. Sybingco, "Robust stock trading using fuzzy decision trees", *Computational Intelligence for Financial Engineering and Economics (CIFEr)*, pp. 1-8, 2012.

[12] R. Dash and P.K. Dash, "A hybrid stock trading framework integrating technical analysis with machine learning techniques", *The Journal of Finance and Data Science*, pp. 42-57, 2016.

[13] J. Potvin, P. Soriano, M. Vallée "Generating trading rules on the stock markets with genetic programming", *Computers & Operations Research*, 2004.

[14] R. L. Rivest, "Constructing optimal binary decision trees is NP-complete", IRIA - Laboria, 1976.

[15] A. Ghandar, Z. Michalewicz, M. Schmidt, T. To, and R. Zurbrugg, "Computational Intelligence for Evolving Trading Rules", *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION 1*, pp. 71-86, 2009.

[16] Python Data Analysis Library, Web: http://pandas.pydata.org/

[17] NumPy framework, Web: http://www.numpy.org/

[18] W.F. Sharpe, "The sharpe ratio", *The journal of portfolio management*, 1994, pp. 49-58.