Comparison of Genetic Algorithms for Trading Strategies

Petr Kroha[1] and Matthias Friedrich[2]

[1] Czech Technical University in Prague,
Faculty of Information Technology, Department of Software Engineering,
Thakurova 9, 160 00 Praha 6, Czech Republic
kroha@informatik.tu-chemnitz.de
[2] Chemnitz University of Technology, Strasse der Nationen 62, 09111 Chemnitz, Germany
matthias_friedrich@ymail.com

Abstract. In this contribution, we describe and compare two genetic systems which create trading strategies. The first system is based on the idea that the connection weight matrix of a neural network represents the genotype of an individual and can be changed by genetic algorithm. The second system uses genetic programming to derive trading strategies. As input data in our experiments, we used technical indicators of NASDAQ stocks. As output, the algorithms generate trading strategies, i.e. buy, hold, and sell signals. Our hypothesis that strategies obtained by genetic programming bring better results than buy-and-hold strategy has been proven as statistically significant. We discuss our results and compare them to our previous experiments with fuzzy technology, fractal approach, and with simple technical indicator strategy.

Keywords: Genetic algorithms, neurogenetic approach, neuroevolutionary system, genetic programming, neural network, investment, forecast, trading, financial modeling, technical analysis.

1 Introduction

Analyzing financial markets is a very interesting and popular field. Especially, forecasting is a hot topic. However, the question is how successfully and reliable a market behavior can be predicted. There is no consensus in the expert community because two main, contradictory, competing hypotheses on market processes have been formulated.

Efficient market hypothesis [6], [11] states that markets are efficient in the sense that current stock prices reflect completely all currently known information that could anticipate future market, i.e. there is no information hidden that could be used to predict future market development.

Later, inefficient market hypothesis [15] was formulated because some anomalies in market development have been found that cannot be explained as being caused by efficient markets. More or less, market trading need buyers and sellers at the same time. So, a consensus would stop trading.

Compared to systems in physics, reflexivity of markets and investors is a very important factor. It states that investors influence the market by changing their biases, mind,

V. Geffert et al. (Eds.): SOFSEM 2014, LNCS 8327, pp. 383–394, 2014.

interest, and trading rules. Similarly, market changes influence behavior of investors. It will be investigated by crowd psychology.

Market processes are driven by events and by trends. Events happen and are represented by news. Predictable events have usually no influence on stock prices because investors presume them and prices adapt to them before. Unpredictable events cause big changes in stock prices but they are unpredictable like earthquake. Trends are given by investor's behavior. The chance to predict trends seems to be slightly better than to predict earthquake. So, the effort to forecast markets is more or less the effort to predict investors' behavior. It can bring good results in time periods when trends are dominating and only expected events happen. We did not investigate possibilities of short term prediction, e.g. for day trading, because the influence of noise is stronger than in the case of long term prediction. Because of that we compare the performance of our prototypes with buy-and-hold method that will be used as a standard in such investigations.

Genetic algorithms can be used in many ways to optimize systems. The important parts of the algorithms are: how to specify what will be coded as genotype, and how the fitness will be calculated.

The first possibility of using genetic algorithms is that neural network (Fig. 1) connection weights can be optimized by a genetic algorithm instead of the commonly used back propagation method. Elements of the weight matrix describing the neural network topology are coded as real values and mapped into genotype. Such a system will be called neurogenetic or neuroevolutionary system. We implemented such system as our prototype A - Section 3.1.

The second possibility how to use genetic algorithms in financial application is that system parameters (e.g. open and close stock value, technical indicators) are coded as tree leaves (operands), and operators working with them (e.g. or, and, if, less-or-equal) are coded as tree nodes (Fig. 2). This method will be called genetic programming (GP). We start using a random placement of operators and operands into a tree that will represent the genotype (in our prototype B). As published in [1], we used the swapping of subtrees from both parents for the crossover. The following mutation selects a subtree of a parent and replaces it by a randomly created tree - Section 3.2.

All genetic algorithms follow the same procedure. After a initial population of genotypes is constructed, the fitness of each individual is calculated. For the recombination operation, two parents are selected according to their fitness (in our prototypes with the linear ranking method), and their genotypes are crossed. The new created individual is subsequently modified by the mutation operation. This process will continue until the new population is created, which is consistently followed by applying the fitness calculation and the genetic operations.

In this paper, we describe the trading system A based on neurogenetic approach and the genetic programming trading system B. Our original contribution is that we compounded methods of [2], [9], and [14] for the neurogenetic approach in our prototype A, and also improved the genetic programming method (tree swapping) presented in [1] in our prototype B. We added technical indicators as tree nodes and we modified the probability of the creation of tree nodes. We specified our own fitness functions for both prototypes. Both methods are described in detail in Section 3.

Additionally, we successfully tested statistical significance of the hypothesis that the genetic programming prototype B brings better results than buy-and-hold strategy used in comparisons as a standard. Further, we compared all achieved results, and results obtained using fuzzy and fractal technology that we used in our previous works [8].

The rest of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we introduce the developed prototypes. Fitness function is described in Section 4. Then, we present data used in Section 5. The following section 6 describes the implementation, experiments, and results. Statistical significance proving is given in Section 7. The comparison of genetic with fuzzy and fractal methods is mentioned in Section 8. In the last section, we conclude our work.

2 Related Work

There are many interesting works investigating similar problems as our research.

In [9], a neural network having one hidden-layer has been used. It was trained using back propagation to find a local optimum. Compared to this work, we did not use back propagation in the prototype A.

In [2], a multi-layer network was used. Genetic algorithm mutation changes weights but also the topology of a cycle-free neural network. It was used for day traders, and after a planned day profit was reached, the system generated a sell signal. Compared to this work, we did not optimize the topology, but we used recombination of individuals and applied the moving window technique in the prototype A. Our output was not day trader oriented.

The authors of [14] suppose that there is no optimum of buy and sell signals, and because of that supervised learning (e.g. back propagation) cannot be used. To optimize weights, only genetic algorithm was applied using the moving window technique.

In [5], back propagation is replaced by simulated annealing, the input vector represents 5 technical indicators. The output is a stock value predicted for the next day, i.e. buy-, hold-, and sell signals are not generated.

Fuzzy technology instead of genetic algorithm to optimize network topology is used in [10].

In [7], authors focus on optimization of technical indicator parameters but differently to our approach they do not use tree swapping.

A method to evaluate individuals which were proposed to be applied in automated trading is described in [12]. Instead, we used our own fitness function described below.

It is very difficult to compare the methods mentioned above because of the large variety of approaches, parameters, and used data. So, we compared tests of our prototypes running on the same data.

3 Our Prototypes

3.1 Our Neurogenetic Prototype A

Our goal was to investigate how a trading strategy in terms of buy, sell, and hold signals can be represented and generated from output values of a neurogenetic system. Furthermore we wanted to prospect whether using a neurogenetic system can bring better

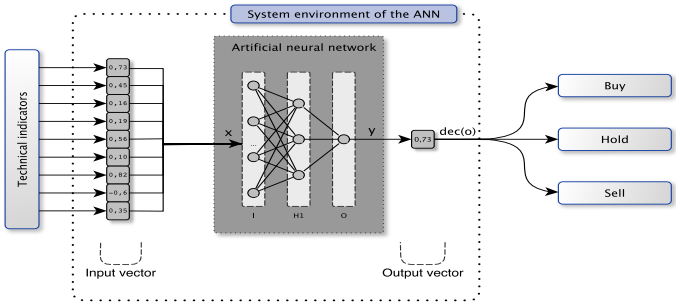

Fig. 1. Instance of a neural network as an individuum of genetic algorithm

results than genetic programming, fuzzy and fractal technology that we investigated in our previous work [8].

For the basis of our prototype A we used ideas from [2], [9], and [14] in a specific composition with some additional improvements.

The important aspect of genetic optimization is how to map the problem into the genotype. In this case, the input vector is represented by system parameters. They are combined using an activation function and weight parameters which are stored in a matrix correspondingly to the network topology. The output vector is finally decoded into buy, hold and sell signals. To recombine genotype of two parents (i.e. their matrices) the 2D-recombination method will be used. For the mutation, we developed the noise-layer-mutation, which added small randomly created values to each element of the matrix of a specific layer using the gaussian distribution $\mathcal{N}(0, 1)$.

The weight optimization was implemented using a genetic algorithm in periods with moving window technique. We abstained from the back propagation algorithm because we agree with the assumption in [14] saying that buy and sell decisions for middle and long term trading strategy cannot be predicted using supervised learning algorithms.

3.2 Our Prototype B Based on Genetic Programming

Our genetic programming prototype B is an improved algorithm based on the method published in [1]. As described in Section 1, the trading rule is represented by a tree using system parameters as leaves and operators as tree nodes Fig. 2.

Our main improvement is that we used different selection probabilities for different kinds of nodes. This means, when creating a random tree (e.g. for initial population or mutation), the selection probability is not uniform distributed over nodes in one category, as given in Table 1. The effect is that system values like technical indicators have a higher probability of occurrence within the tree, and therefore they influence the trading strategy more than fixed parameters. Furthermore, it reduced the number of combinations that have semantically only very limited or improbable occurrence but cause a tree explosion. The probabilities we used are our estimations based on our experience with description of many strategies.

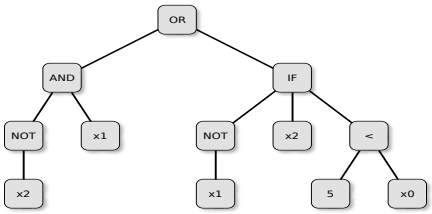

Fig. 2. An instance of a tree in genetic programming

Table 1. Probabilities of rule application

	Parameter	Value	P
Boolean	Basic functions	if-then-else, and, or, not	12.5 % each
	Enhanced functions	$<, >$	25.0 % each
Numeric	Basic functions	+, -, *, : , norm	2.0 % each
	Close value	price	30 %
	Enhanced function	number, average, maximum, minimum, lag	4.0 % each
	Indicators	ROC, MACD, SO, TCI	10 % each

4 Fitness Function

Fitness function evaluates behavior of phenotype, i.e. behavior of the individual that was generated from genotype. Next generation individuals will be constructed from two individuals that have usually a high fitness function value, since all common selection methods (i.e. linear ranking) are fitness-oriented.

In our applications, individuals represent trading strategies. An important factor of the fitness function is the money earned by each strategy m_{strat}. An individual obtains a start money amount m_{init} and uses it following its strategy s_t during n days. The money m_{strat} of an individual, i.e. of each strategy, achieved in a period is calculated using close stock value x_t and the market position pos_t of day t corresponding to the trading strategy:

$$m_{strat} = m_{init} \cdot \prod_{t=i-n+1}^{i} \left(\frac{x_t}{x_{t-1}} \right)^{pos_t}, \quad pos_t = \begin{cases} 1 & \text{, if } s_{t-1} \Leftrightarrow \text{ buy} \\ 0 & \text{, if } s_{t-1} \Leftrightarrow \text{ sell} \\ pos_{t-1} & \text{, if } s_{t-1} \Leftrightarrow \text{ hold} \end{cases} \quad (1)$$

The calculation of the fitness function in our neurogenetic system contained several components as described below:

– profit of each strategy f_{strat} is calculated in relation to the profit of the buy-and-hold strategy (m_{bah} denotes money obtained by the buy-and-hold strategy)

$$f_{strat} = m_{strat} - m_{bah} = m_{strat} - \left(m_{init} \cdot \frac{x_i}{x_{i-n+1}} \right) \quad (2)$$

– absolute value of profit - $f_{strat-abs}$ - indicates the profit or loss of money caused by the strategy, i.e. it represents the rule that we want not only to be better as the buy-and-hold strategy, but we do not want to loose money

$$f_{strat-abs} = m_{strat} - m_{init} \tag{3}$$

– penalty function - $f_{strat-pen}$ - penalizes individuals that do not change position and simply follow buy-and-hold strategy during $\tilde{n}$ coherent days, using an adjustment factor α

$$f_{strat-pen} = \alpha \cdot m_{init} \cdot \left(\frac{\tilde{n}}{n}\right)^3 \tag{4}$$

– absolute relation between profit and loss - $f_{strat-pl}$ - respects the risk β resulted from the using of the trading strategy

$$f_{strat-pl} = \beta \cdot \overline{p} \;\; + \;\; (1-\beta) \cdot \overline{l} \tag{5}$$

The average profit $\overline{p}$ is given by positive return during a period of days running consecutively

$$\overline{p} = \frac{1}{\sum_{t=i-n+1}^{i} p_t} \cdot \sum_{t=i-n+1}^{i} \left(\frac{x_t}{x_{t-1}} \cdot m_{t-1}\right) \cdot p_t, \quad p_t = \begin{cases} 1 & \text{, if } x_t > x_{t-1} \\ 0 & \text{, otherwise} \end{cases} \tag{6}$$

The average loss $\overline{l}$ is calculated similarly:

$$\overline{l} = \frac{1}{\sum_{t=i-n+1}^{i} l_t} \cdot \sum_{t=i-n+1}^{i} \left(\frac{x_t}{x_{t-1}} \cdot m_{t-1}\right) \cdot l_t, \quad l_t = \begin{cases} 1 & \text{, if } x_t < x_{t-1} \\ 0 & \text{, otherwise} \end{cases} \tag{7}$$

The components described above are combined in the fitness value $\mathcal{F}$:

$$\mathcal{F}_{strat} = (1-\gamma) \cdot f_{strat-bah} + \gamma \cdot f_{strat-abs} + f_{strat-pen} + f_{strat-pl} \tag{8}$$

The parameter γ represents the weight of f_{bah} and f_{abs}. In our preliminary study, we found that $\gamma = 0.6$ was the most suitable value. Other authors construct different fitness functions.

5 Data Used

In the first part of our experiments called preliminary study, we tried to specify suitable parameters and their values that could be used as fixed during the optimization problem. Since we used connection weights to be optimized in our neurogenetic prototype A,

there are many methods which could be applied for the genetic algorithm. Because of the complexity, it is practically impossible to optimize all the parameters.

During the preliminary study we evaluated different parameter combinations for the neurogenetic system, i.e. we altered the topology, recombination and mutation methods. Starting with the configuration given in Table 3, we picked the parameters in succession and evaluated different values and methods. The description of all experiments and evaluations of our preliminary study is out of scope of this paper. Of course, we know that the parameter combination we fixed does not guarantee the global optimum but computational complexity of other approach were immense. The final configuration used for the test is given in Table 4.

Table 2. Input vector variables of our neural network

Position	Variable	Position	Variable	Position	Variable
1	$close_t$	14	$MACD_t(20, 40)$	27	$SO_t(40)$
2	$SMA_t(5)$	15	$ROC_t(20)$	28	$TCI_t(40, 80)$
3	$EMA_t(5)$	16	$SO_t(20)$	29	$BB_t(40, 2.0)$
4	$MACD_t(10, 20)$	17	$TCI_t(20, 40)$	30	$RAVI_t(10, 100)$
5	$ROC_t(10)$	18	$BB_t(20, 2.0)$	31	$RSI_t(14)$
6	$SO_t(10)$	19	$RAVI_t(6, 60)$	32	$SMA_t(50)$
7	$TCI_t(10, 20)$	20	$RSI_t(9)$	33	$EMA_t(50)$
8	$BB_t(10, 2.0)$	21	$high_t$	34	$SMA_t(100)$
9	$RAVI_t(3, 30)$	22	low_t	35	$EMA_t(100)$
10	$RSI_t(3)$	23	$SMA_t(20)$	36	$SMA_t(200)$
11	$open_t$	24	$EMA_t(20)$	37	$EMA_t(200)$
12	$SMA_t(10)$	25	$MACD_t(40, 80)$		
13	$EMA_t(10)$	26	$ROC_t(40)$		

As 37 input values, we used technical indicators (SMA means Simple Moving Average, EMA means Exponential Moving Average etc. - all acronyms are given in [13]) and stock values (open, high, low, close) as given in Table 2.

Both prototypes have been tested at stocks of companies listed in NASDAQ-100 in June 2009. As time period for the evaluation we considered the 01.01.2003 start and the 01.10.2009 as end point. A test case is represented by the values of a time series of a stock in one year, which we used as test period. The training and selection period contained the values of 12 respectively 6 months directly before start of the test period. Since some companies do not have stock values in the whole periods, there were 621 test cases.

6 Experiments and Results

To run our experiments with prototype A, we used an Apple Mac Pro 4,1 with two Intel Xeon E5520 processors, 4 cores / 8 threads each, clock rate 2,26 GHz. The experiments of our prototype B have been executed on a computer with processor Intel Core 2 Duo E6300, clock rate 1,86 GHz.

Table 3. The configuration of the neurogenetic system in the preliminary study

Category	Parameter	Value
Topology	Input-layer	37
	Hidden-layer 1	20
	Hidden-layer 2	8
	Output-layer	1
Activation function	Function type	Logistic Function
	Parameter α	2
	Parameter β	3
	Parameter γ	3
Threshold parameters	Activation potential θ	1.0
	Buy signal t_{buy}	0.66
	Sell signal t_{sell}	-0.66
Population	Initialization	Gaussian distribution
	Number of generations	100
Selection	Selection method	Linear ranking
	Elitism	0
	Minimum	0.5
	Maximum	1.5
Crossover	Recombination method	Layer-crossover
	Probability p_c	0.8
Mutation	Mutation method	Noise-layer
	Probability p_m	0.2
	Distance	0.1
Capital	Seed money	10000 $
	Transaction costs	Relative: 0.25 %
Fitness measurement	Computation strategy of fitness	See section 4
Moving windows	Number of windows	6
	Number of overlapping days	30
Population size	Training period	1000
	Selection period	50
	Test period	1

Experiments were very time consuming. Each algorithm execution took about 27 minutes and 18 seconds, i.e. the tests would need altogether approximately 282 hours and 38 minutes. Since we could parallelize our prototype A, we only needed 156 hours 49 minutes for the evaluation.

Both methods achieved better results than the strategy buy-and-hold. In the case of genetic programming, the prototype B makes 2.72 % a higher profit towards the buy-and-hold strategy.

Compared to buy-and-hold strategy, the strategies of the neurogenetic prototype A earned in average 91.82 % more during the training period, but only 0.65 % more during the test period. It achieved in 99.19 % better results as buy-and-hold strategy during the training period, but only 44.28 % during the test period. It seems to be overfitted.

The strategies generated by the genetic programming system (prototype B) earned in average only 55.20 % more than buy-and-hold strategy in training period, but 2.72 %

Table 4. The configuration of the final tests

Parameter	Neurogenetic system	Genetic programming
Attributes of Individuals	Input-layer: 37 Hidden-layer: 19 Output-layer: 1 Tangens hyperbolicus $t_{buy} = 0.5; t_{sell} = -0.33; \theta = 0.0$	Non-uniform distribution, see Table 1 Maximum tree depth: 7
Population	Initialization: Gaussian distribution Generations: 150	Generations: 75
Selection	Linear ranking Minimum: 0.5 Maximum: 1.5 Individuals for elitism: 0	
Crossover	2D-recombination $p_c = 0.9$	Tree swapping $p_c = 0.7$
Mutation	Noise-layer $p_m = 0.7$ Distance: 0.1 (Gaussian distribution)	Tree switching $p_m = 0.7$
Capital	Seed money: 10.000 $ Transaction costs: 10 $ for each transaction	
Fitness measurement	See section 4	
Moving windows	Number of windows: 2 Number of overlapping days: 75	–
Population size	Training period: 100 Selection period: 50 Test period: 1	Training period: 50 Selection period: 20 Test period: 1

more during the test period. During the training period, it was better in 90.82 % of cases; during the test period, it was better in 44.61 % of cases.

If we consider only stocks that provide a better performance using the generated strategies compared to buy-and-hold strategy then strategies generated by prototype A (neurogenetic) give 26.98 % and strategies generated by prototype B (genetic programming) give 28.73 % more earning during the test period. The results are summarized in Table 5.

7 Statistical Hypothesis Testing

Because of the results described above, we used statistical hypothesis testing (Z-test because the sample size is large and the population variance known) and proved both prototypes using the following hypotheses with the parameters $\alpha = 5$ % and $\mu_0 = 1.0$ which leads to the parameter $\Phi(z_{1-\alpha}) = 1.6449$.

$H_0 =$ the expected earning of the generated strategy is equal or less compared to the earning of the buy-and-hold strategy

$H_1 =$ the expected earning of the generated strategy is greater compared to the earning of the buy-and-hold strategy

Table 5. Results of neurogenetic and genetic programming system compared to buy-and-hold strategy

	Neurogenetic system	Genetic programming
Average (Training)	1.9182	1.5520
Average (Test)	1.0065	1.0272
Number of better cases (Training)	616	564
Number of worse cases (Training)	5	57
Number of better cases (Test)	275	277
Number of worse cases (Test)	346	344
Average of better cases(Training)	1.9392	1.6160
Average of better cases (Test)	1.2698	1.2873

For the neurogenetic system we obtained $z = 0.4728 < \Phi(z_{1-\alpha})$. It means that the hypothesis H_0 can neither be rejected nor accepted.

In contrast, we obtained $z = 1.9910 > \Phi(z_{1-\alpha})$ for the genetic programming. This means that the hypothesis H_0 can be rejected. Therefore we can state that the strategies generated by genetic programming method earns more than the strategy buy-and-hold.

8 Comparison to Fuzzy and Fractal Technology

In our previous work [8], we investigated how fuzzy technology, fractal technology, and using of technical indicators can be used to generate trading strategies. Now, we used the same time series to prove the generated strategies as in [8], which allows the comparison of all five methods. The results obtained are shown sorted in Table 6. We can see that the system based on genetic programming implemented in prototype B brings the best results.

Table 6. Comparison of neurogenetic and genetic programming systems to results of fuzzy technology, fractal technology, and simple application of technical indicators

	Average	Standard deviation
Genetic programming	1.1149	0.8872
Neurogenetic system	0.9851	0.6954
Fractal analysis	0.9664	0.7247
Fuzzy control	0.8392	0.3566
Technical indicators	0.6473	0.3118

The small difference between Table 5 and Table 6 occurred because we investigated only 82 stocks of NASDAQ-100 in our previous work but all 100 stocks in this work.

There is a question concerning the size of data necessary for the training and selection process. To use our results in practice with several stocks, the time consumed by the execution must be reduced. A modest trader in Middle Europe respects the stock exchange in New York closing at 22:30 (CET) and Frankfurt stock exchange opening at

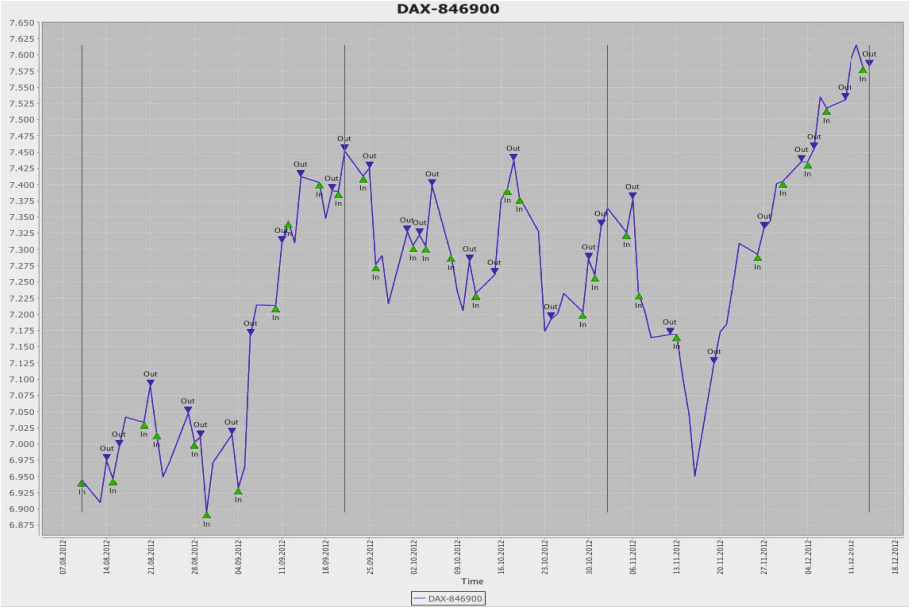

Fig. 3. Buy and Sell Signals for the DAX generated by genetic programming, separated in training, selection and test periods

9:00 (CET) the next day. There are 10.5 hours available to compute the most promising trading strategy for the next day influenced by the last data.

In subsequent experiments, we investigated a time series of one stock only and reduced the training, selection, and test to 1 month each and used 150 generations. The process running in 2 threads took 2 minutes only and gave signals shown in Fig. 3.

9 Conclusion

We implemented, improved, and tested two prototypes of methods based on genetic algorithms that generate trading strategies. In difference to other works, we took transaction costs into account (Table 4).

We expected that the neurogenetic prototype A based on genetic optimization of neural network weights brings the best results. However, we found that the method of genetic programming generates a better trading strategy, and that it brings more profit than the buy-and-hold strategy which is usually used for comparison. We proved statistical significance of this result. As we mentioned above, the optimization process is very time consuming when large data is used. We recognized that prior behavior of markets has a limited information content relative to current trading behavior. The difference between performance of trading decisions during a training period and during a test period is very big (Table 5).

In common, complex system like markets are influenced by very many parameters which can be further investigated.

Practically, trading practices used by investment banks and funds are secret, of course. However, they are not always successful. Some of them go bankrupt, e.g. the largest bankruptcy in U.S. history - Lehman Brothers loss 600 bilions in assets in 2008.

There is never a real consensus in finance - everybody tries to outsmart everybody else. This is why there are still buyers and sellers - a real consensus would stop trading. Obviously, markets differ from physical systems. It is known that using insider information or pretending investments are commonly used strategies, and we cannot model them. Because of such practices and because of the chaotic component given by events our market modeling possibilities remain limited.

References

1. Allen, F., Karjalainen, R.: Using genetic algorithms to find technical trading rules. Journal of Financial Economics 51, 245–271 (1999)
2. Azzini, A., Tettamanzi, A.: Evolving Neural Networks for Static Single-Position Automated Trading. Journal of Artificial Evolution and Applications, 1–17 (2008)
3. Brabazon, A., O'Neill, M.: Biological Inspired Algorithms for Financial Modelling. Springer (2006)
4. Brabazon, A., O'Neill, M., Dempsey, I.: An Introduction to Evolutionary Computation in Finance. IEEE Computational Intelligence Magazine, 42–55 (2008)
5. El-Henawy, I.M., Kamal, A.H., Abdelbary, H.A., Abas, A.R.: Predicting Stock Index Using Neural Network Combined with Evolutionary Computation Methods. In: The 7th International Conference on Informatics and Systems (INFOS), pp. 1–6 (2010)
6. Fama, E.: Efficient capital markets: A review of theory and empirical work. Journal of Finance 25, 383–417 (1970)
7. Kapoor, V., Dey, S., Khurana, A.P.: Genetic Algorithm: An Application to Technical Trading System Design. International Journal of Computer Applications 36(5) (2011)
8. Kroha, P., Lauschke, M.: Using Fuzzy and Fractal Methods for Analyzing Market Time Series. In: Proceedings of the International Conference on Fuzzy Computation and International Conference on Neural Computation ICFC 2010 and ICNC 2010, pp. 85–92 (2010)
9. Kwon, Y.-K., Moon, B.-R.: A Hybrid Neurogenetic Approach for Stock Forecasting. IEEE Transactions on Neural Networks 18, 851–864 (2007)
10. Li, R., Xiong, Z.: A Modified Genetic Fuzzy Neural Network with Application to Financial Distress Analysis. In: International Conference on Computational Intelligence for Modeling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (2006)
11. Malkiel, B.: A Random Walk Down Wall Street. W.W. Norton, New York (1996)
12. Matsui, K., Sato, H.: Neighborhood Evaluation in Acquiring Stock Trading Strategy Using Genetic Algorithms. International Journal of Computer Information Systems and Industrial Management Applications 4, 366–373 (2012)
13. Murphy, J.J.: Technical Analysis of the Financial Markets. Prentice Hall (1999)
14. Skabar, A., Cloete, I.: Neural networks, Financial Trading and the Efficient Markets Hypothesis. In: Proceedings of the Twenty-Fifth Australasian Conference on Computer Science ACSC 2002, vol. 4, pp. 241–249 (2002)
15. Shleifer, A.: Inefficient Markets – An Introduction to Behavioral Finance. Oxford University Press (2000)