Genetically Evolved Agents for Stock Price Prediction

Milan Jakel

Research Institute of the IT4Innovations Centre of Excellence, Silesian University in Opava, Bezrucovo namesti 13, 746 01 Opava, Czech Republic, milan@jakel.cz

Petr Sosík

Research Institute of the IT4Innovations Centre of Excellence, Silesian University in Opava, Bezrucovo namesti 13, 746 01 Opava, Czech Republic, petr.sosik@fpf.slu.cz

Abstract: Our intention is to evolve agents genetically to maximize their stock price prediction ability. A newly designed stock price prediction algorithm benchmark is used as a fitness function. A portfolio of seven US blue-chip stocks has been set for experimental purposes. We use daily time series of stock prices from 2000 to 2011 divided into two segments, in-sample for genetic algorithm evolution and out-of-sample for evaluation. Agents act as prediction algorithms based on Japanese candlestick patterns expressed as logical formulas and encoded by a tree encoding. Evaluation by the benchmark shows this is a promising way to develop successful stock prices prediction algorithm.

Keywords: agent; genetic; multi-agent; genetic algorithm; time series; financial; stock; stock market; prediction; forecast

1 Introduction

The original methods for financial time series prediction are based on mathematical statistics [3, 16]. When predicting a stock market we create a prediction of a particular time series [3]. Each stock traded on a stock exchange is characterized by a time series. A record of such a time series includes four price values: open price, high price, low price and close price [8, 20]. For short, we call it an OHLC time series. See an example of an OHLC time series in Table 1.

Table 1
An example of several records of daily OHLC stock time series

Date	Open	High	Low	Close
2011/12/19	$ 17,34	$ 17,57	$ 17,22	$ 17,36
2011/12/20	$ 17,36	$ 17,38	$ 17,22	$ 17,22
2011/12/21	$ 17,22	$ 17,30	$ 17,17	$ 17,21

The multi-agent paradigm [10, 21] offers another possible approach to stock time series prediction. An agent acts as a prediction algorithm. The stock exchange is the environment in which such agents operate. The environment provides an input for the agent. The input is an OHLC time series of a particular stock. The agent affects the environment by producing stock exchange orders for buying and selling stocks. The scheme of an agent and a multi-agent environment is depicted in Figure 1.

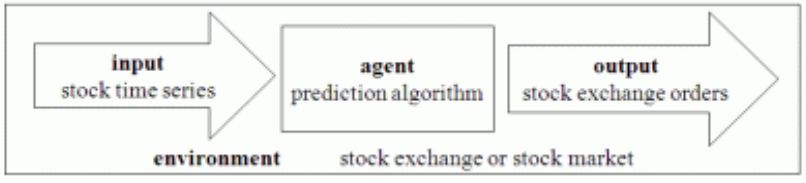

Figure 1
Scheme of an agent, its inputs, outputs, and the environment

There are four types of **stock exchange orders** [8, 20]:

- BUY to buy stocks,
- SELL to sell stocks previously bought using a BUY order,
- SHORT to borrow and sell stocks,
- COVER to buy and return stocks previously borrowed and sold using a SHORT order.

For clarification, we consider MARKET stock exchange orders only [20].

Agents can profit from an **uptrend** of stock prices by realizing a **long** position. A long position is opened by placing a BUY order. It is closed by placing a SELL order. On the other hand, agents can profit from a **downtrend** of stock prices by realizing a **short** position. A short position is opened by placing a SHORT order and it is closed by placing a COVER stock exchange order.

Agents in our experiment **autonomously decide** when to open a long position (long trade) or when to open a short position (short trade) or when to do nothing. For this purpose, there is a **prediction algorithm** as a crucial part of the internal structure of the agent. In our experiment, all positions (trades) are closed (terminated) at the end of the trading day, and trading costs (broker commissions) are not included.

Many recent papers deal with genetic evolution of prediction algorithms based on indicators of technical analysis [8, 20]. In this paper, agents do not use genetically evolved indicators of technical analysis but Japanese candlestick patterns [17, 19] as their prediction algorithms. We generalize this approach and draw inspiration from principles of data mining and knowledge discovery [11].

In the process of time series prediction, it is necessary to set an **error function**. And when doing a genetic evolution of agents, it is necessary to set a **fitness function**. In most of the recent papers, there are widely-known mean square error (MSE) and root mean square error (RMSE) error functions [3] used both as fitness functions and as evaluation functions. In this paper, we use the newly-designed stock price prediction algorithms benchmark which is based on the simulation of trades the prediction algorithm has generated. The benchmark is able to compare both numerical and categorical prediction algorithms.

For simplicity, let us consider the term 'agent' and the term 'prediction algorithm of agent' as synonyms and let us later in this paper solely use the term 'agent'.

2 Literature Review

2.1 Recent Papers

Nowadays, stock market prediction is frequently based on techniques and tools of artificial intelligence. Construction methods of stock price prediction algorithms by genetic algorithms are presented in [1], [4], [6], [13], and [14]. Soft computing methods, like neural networks and fuzzy systems, for time series prediction are described, e.g. in [15].

In [1], the authors use genetic algorithm (GA) with tree encoding. Entry rules they search for are based on mathematical statistics, while entry rules in this paper are based on candlestick patterns.

Chen [4] uses GA to generate entry rules based on indicators of technical analysis [8, 20], for instance, relative strength index (RSI). Chen's fitness function includes a standard deviation of returns (i.e. profits and losses), while our fitness function includes a variance coefficient of the set of profits and losses. Chen uses a generational genetic algorithm and a fixed generations termination condition, as we do in our experiment.

Genetic programming and genetic network programming are the basic applications and extensions of genetic algorithms. In [6], they use genetic network programming to develop a stock trading model based on indicators of technical analysis. Their multi-agent network system is trained by SARSA learning algorithm.

In [13], Japanese candlestick patterns are modelled using fuzzy linguistic variables. This model is further used in [14] and extended with a genetic algorithm for the selection of the fittest candlestick patterns.

In [9], we proposed a design of a stock price prediction algorithm based on case based reasoning (CBR). The problem of measuring the distance between the original case (stored in a knowledge base) and the new case, which is the main problem to solve during CBR system design, inspired the author to create the stock price prediction algorithms benchmark described in the next section.

2.2 Stock Price Prediction Algorithms Benchmark

The main idea of the benchmark is to evaluate a stock price prediction algorithm. It is done by quantifying two description statistics values of the set of profit and losses. Such a set is resulting from transactions executed (simulated) according to entry signals generated by the prediction algorithm. We use the benchmark in this paper in two ways:

- to construct a fitness function for the GA;
- to benchmark the fittest agents resulting from the GA.

The price at which the position is opened is called the **entry price** (denoted as *Entry*) [20]. The price at which the position is terminated is called the **exit price** (denoted as *Exit*) [20]. The benchmark calculations are specified below.

Let t be the number of entry signals generated by a given agent operating on a given time series. Let $P(1), P(2), ..., P(t)$ be a set of coefficients of profit or loss resulting from the corresponding transactions.

For a long trade $r \in \langle 1, t \rangle$ we calculate the coefficient of profit or loss as follows:

$$P(r)_{long} = Exit / Entry \tag{1}$$

For a short trade $r \in \langle 1, t \rangle$ we calculate the coefficient of profit or loss as follows:

$$P(r)_{short} = Entry / Exit \tag{2}$$

$P(r)$ coefficients are relative numbers. In the benchmark, we calculate the average and the variance of the $P(r)$ relative numbers set. The geometric mean is suitable to calculate an average of such set of relative numbers.

The geometric mean M of a set of t values $P(1), P(2), ..., P(t)$ is defined as follows:

$$M = \left(P(1) \,.\, P(2) \,.\, \ldots \,.P(t) \right)^{\frac{1}{t}} = \sqrt[t]{P(1) \,.\, P(2) \,.\, \ldots \,.P(t)} \tag{3}$$

Let *P(1), P(2), ..., P(t)* be a set of *t* values and let *M* be the geometric mean of the *P(1), P(2), ..., P(t)* set. The variation coefficient *C* of the *P(1), P(2), ..., P(t)* set is defined as follows:

$$C = \frac{\sqrt{\sigma^2}}{M} \tag{4}$$

where σ^2 is the variance defined as follows:

$$\sigma^2 = \frac{1}{t}\sum_{i=1}^{t}(P(i) - M)^2 \tag{5}$$

3 Multi-Agent System Specification

We assume a population of *n* agents *A(1), A(2), ..., A(n)*. We assume *m* generations of agents. Let us denote the initial generation as *A(1, 0), A(2, 0), ..., A(n, 0)* and the last generation as *A(1, m), A(2, m), ..., A(n, m)*. Each agent represents a prediction algorithm based on a candlestick pattern encoded by a tree encoding.

We assume a set of *k* OHLC time series *S(1), S(2), ..., S(k)*. The environment of the multi-agent system [10, 21] is represented by the *S(1), S(2), ..., S(k)* set.

3.1 Agents, Candlestick Pattern Logical Formulas

Japanese candlestick patterns [17, 19] can be expressed in a form of logical formulas. In Figure 2, there are the Bullish Engulfing pattern and the Bearish Engulfing pattern depicted in a candlestick chart.

Figure 2
Bullish Engulfing pattern and Bearish Engulfing pattern. Source: [19]

An occurrence of the Bullish (or Bearish) Engulfing pattern in a candlestick chart should be interpreted as a long (or short) position entry signal [19].

The Bullish Engulfing pattern is expressed in a form of logical formula as follows:

((Open[1] > Close[1]) AND (Open[0] < Close[0])) AND ((Open[1] < Close[0]) AND (Open[0] < Close[1])) (6)

The Bearish Engulfing pattern is expressed in a form of logical formula as follows:

((Open[1] < Close[1]) AND (Open[0] > Close[0])) AND ((Open[1] > Close[0]) AND (Open[0] > Close[1])) (7)

Candlestick pattern formulas (6) and (7) are composed of four variables *Open[], High[], Low[], Close[]* with the **shift parameter** in square brackets. Value *[0]* means no shift. Usually, when there is no shift, i.e. *[0]*, the shift parameter is omitted. Usually, *Open[]* is indicated as *O[]* for short, *H[]* stands for *High[], L[]* stands for *Low[]*, and *C[]* stands for *Close[]*. Value *[1]* means the first previous record of the time series, value *[2]* means the second one, etc.

Furthermore, candlestick pattern formulas (6) and (7) are composed of numerical operators (+, -), comparison operators (<, >, >=, =<), logical operators (*AND, OR*), and parentheses *(,)*.

Each agent *A(c, d)* is represented by a formula composed like (6) and (7). There are two different types of agents: **long agents** and **short agents**. Long agents generate long entry signals only and short agents generate short entry signals only. The candlestick pattern formulas are evaluated for each record of given time series. *TRUE* evaluation of the formula poses an entry signal. *FALSE* evaluation of the formula poses no signal.

3.2 Environment, Stock Price Time Series

For the purposes of our experiment, a portfolio of seven stocks listed in the US stock index Dow Jones Industrial Average (DJIA) has been compiled. All stocks in the portfolio are traded at NYSE and NASDAQ stock exchanges. The portfolio is listed in Table 2.

Table 2

Portfolio of seven global stocks used in our experiment. Source: [22]

Symbol	Company name	Sector
AA	Alcoa Inc.	basic materials
BA	Boeing Co.	aerospace
CAT	Caterpillar Inc.	industrial goods
DIS	Walt Disney Co.	entertainment
GE	General Electric Company	machinery

IBM	International Business Machines Corp.	computer systems
KO	The Coca-Cola Company	consumer goods

We use an OHLC time series with **daily timeframe** [3]. One record of a daily time series represents one stock exchange trading day [8, 20]. Before being used in our experiment, the given time series were split adjusted and dividend adjusted [8].

3.3 Entry Signals and Corresponding Transactions

Let *S(f, i)* be the *i*-th record of time series *S(f)*. Agent *A(c, d)*, where $c \in <1, n>$ and $d \in <0, m>$, working on time series *S(f)*, where $f \in <1, k>$, generates entry signals as follows:

- for every record *i* of time series *S(f, i)* is the logical formula *A(c, d)* evaluated,
- if the evaluation of *A(c, d)* on *S(f, i)* is *TRUE* and *A(c, d)* is a long agent, then an entry signal for a long trade is generated,
- if the evaluation of *A(c, d)* on *S(f, i)* is *TRUE* and *A(c, d)* is a short agent, then an entry signal for a short trade is generated,
- if the evaluation of *A(c, d)* on *S(f, i)* is *FALSE*, then no entry signal is generated.

Entry signals are generated on daily stock time series. This method of generating entry signals is mentioned in [20] and it is called **swing trading** or **intraday trading**. Once we have an entry signal, we enter the market (open a position) at the beginning of next trading day, i.e. 'at open' [8, 20]. We terminate the position at the end of the same trading day, i.e. 'at close'.

4 Genetic Algorithm Specification

A genetic algorithm (GA) is an evolutionary search heuristic resulting in a fittest solution of an encoded problem. The advantage of GA is in its parallelism. Information on GA proposed in this chapter is based on and related to references [2], [5], [7], [12], and [18].

First of all, an encoding mechanism must be designed to represent each agent as a genome. Typically used encodings in GA are binary encoding, value encoding, permutation encoding, and tree encoding. In our experiment, the **tree encoding** is used. See below the specification.

To evaluate the quality of particular agents, a fitness function is needed in GA. A fitness function is used by a selection operator to select quality agents for

reproduction. In our experiment, a **multi-objective** fitness function **based on** stock price prediction algorithms **benchmark** is used. See below the specification.

The initialization method sets the way how the initial generation of agents *A(1, 0), A(2, 0), ..., A(n, 0)* must be initialized at the beginning of the GA evolution. In our experiment, the initial agents are generated by a random function in a form similar to candlestick pattern formulas given in expressions (6) and (7). Such an initialization method implements **randomly generated** candlestick pattern logical formulas.

The selection operator determines how agents are selected from the population for the crossover operator. There are two basic methods to select agents to be parents for crossover: roulette wheel selection and tournament selection. In our experiment, we use the **tournament selection**; i.e. a set of two or more randomly selected agents is compared and the fittest agent is selected for crossover.

The crossover operator exchanges parts of genomes among two or more parent agents to create a new offspring. In our experiment, the **subtree crossover** is used. Subtree crossover produces an offspring by replacing a subtree of one parent agent by a subtree of other parent agent. There is a set of rules to keep the order to secure the validity of the newly created candlestick pattern logical formulas. The principle of the subtree crossover is depicted in Figure 4.

The mutation operator randomly changes agents resulting from the crossover operator. In our experiment, the **subtree mutation** is used. The subtree mutation replaces the parental subtree with a randomly generated subtree. There is a set of rules to keep the order to secure the validity of newly created candlestick pattern logical formulas. Mutation introduces more randomness into the population. The principle of the subtree mutation is depicted in figure 5.

The replacement method specifies how the population is updated by removing parents and adding offsprings. In GA, there are several commonly used replacement methods: uniform replacement, crowding replacement, tournament replacement, and parental replacement. In our experiment, the **parental replacement** is used, i.e. every parent agent is replaced by its offspring agent.

The termination condition determines when to stop the evolution and obtain the solution that is represented by the fittest agent from the last population. There are several termination conditions usually used in GA: fitness convergence, diversity convergence, fitness target, and fixed generations. In our experiment, we use the **fixed generations** termination condition. The evolution is stopped when the *m*-th generation is completed.

4.1 Tree Encoding, Tree Crossover, Tree Mutation

In our experiment, for the purpose of encoding candlestick pattern logical formulas into GA genomes, we use the tree encoding. The principle of the tree encoding is illustrated in Figure 3. There is Bullish Engulfing pattern logical formula (6) encoded into the tree genome.

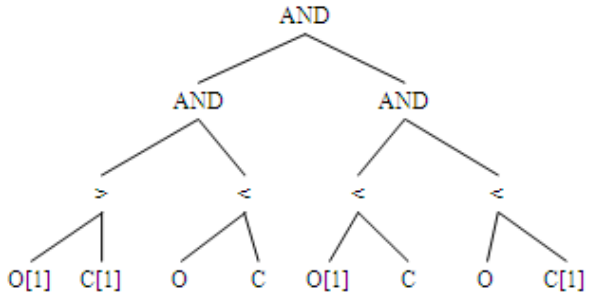

Figure 3
Bullish Engulfing pattern logical formula encoded into the tree genome

In Figure 4, there is an example of a subtree crossover operator. And in Figure 5, there is an example of a subtree mutation principle.

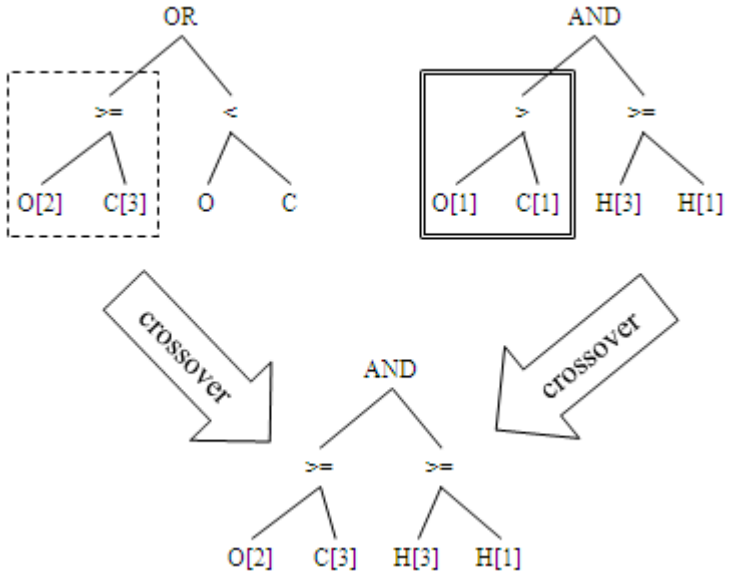

Figure 4
An example of the subtree crossover. The offspring is created by replacing a subtree (marked by double line region) of one parent agent by a subtree (marked by dashed line region) of other parent agent.

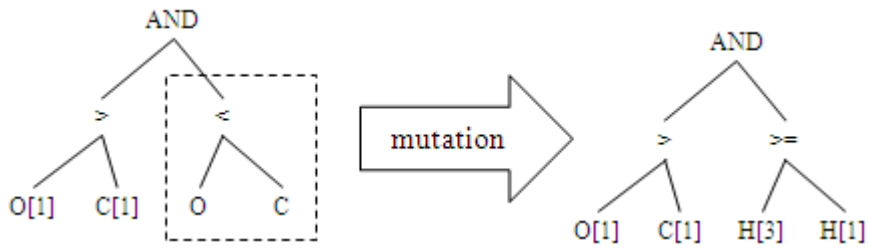

Figure 5

An example of a subtree mutation. The parental subtree intended for mutation is marked by dashed line region. The arrow points from the parental agent to the offspring agent.

4.2 Fitness Function

We use a multi-objective fitness function in our experiment. There are three objectives:

- to maximize the average of profits and losses,
- to minimize the variance of profits and losses,
- to maximize the number of trades (i.e. the number of generated entry signals).

Let $P(r)$, where $r \in \langle 1, t \rangle$, be a set of t coefficients of profit or loss of a given prediction algorithm (or agent) operating on a given time series. The first objective is based on the benchmark calculation of geometric mean (3) of the $P(r)$ set. The second objective is based on benchmark calculation of variance coefficient (4), (5) of the $P(r)$ set.

To calculate the value of the fitness function of agent $A(c, d)$ operating on time series $S(f)$ we firstly need to determine:

- the possible maximum average *MaxM* (geometric mean) of the set of coefficients of profit or loss $P(r)$ that can be reached on a given time series $S(f)$,
- the possible minimum variation coefficient *MinC* of the set of coefficients of profit or loss $P(r)$ that can be reached on a given time series $S(f)$,
- the possible maximum number of trades *MaxNum* that can be done on a given time series $S(f)$.

Let us assume the agent $A(c, d)$ operating on time series $S(f)$. The agent has generated a set of entry signals. The set of profit or loss coefficients $P(r)$ has been calculated according to expressions (1) and (2). Let M be the geometric mean of $P(r)$ set calculated according to expression (3). Let C be the variation coefficient of $P(r)$ set calculated according to expressions (4) and (5). And let *Num* be the

number of elements of *P(r)* set, i.e. the number of entry signals of *A(c, d)* on time series *S(f)*. The fitness function is defined as follows:

$$Fitness = (M / MaxM) + (MinC / C) + (Num / MaxNum) \quad (8)$$

4.3 Summary of GA Parameters

In this chapter, we provide a list of parameters of the genetic algorithm (GA) used in our experiment, see Table 3.

Table 3
Summary of genetic algorithm parameters used in our experiment

Parameter	**Value**
initialization method	random function
population size	50 agents
encoding	tree encoding
maximum allowed tree depth	4 levels
fitness function	multi-objective, see expression (8)
replacement method	parental replacement
selection operator	tournament selection
tournament size	2 agents
crossover operator	tree crossover
crossover production	45 agents (90% of population size)
mutation operator	subtree mutation
mutation production	5 agents (10% of population size)
termination condition	fixed generations
number of generations	25 generations

According to the results of our preliminary experiments we have set the parameters of the GA to achieve the best convergence. We have set the population size to 50 and total number of generations to 25 as for the terminal condition. Such a GA is called **generational genetic algorithm**.

When using a tree encoding, it should be convenient to set a maximum for the number of levels the tree encoding can use. When the value of the maximum is above the optimum, the GA is harder to converge. If allowing a large **maximum tree depth**, then also allowing candlestick pattern formulas created by the GA to become much more complicated, and this is probably resulting into an **overfitting** [3] of given stock price time series. We have set the maximum allowed tree depth to 4 levels.

The GA was best to converge when we set the percentage of offspings created by the crossover operator to 90 percent and the less 10 percent was created by the mutation operator.

Time series *S(f)* used in our experiment represent a continuous period of daily stock price data from 3rd Jan 2000 to 30th Dec 2011. The given time series are split into two periods. The **in-sample** period of given time series is used for the evolution of agents, i.e. for the **training** of the multi-agent system. The in-sample period runs from 3rd Jan 2000 to 31st Dec 2008. The **out-of-sample** period is used for the **evaluation** of the multi-agent system. The out-of-sample period runs from 1st Jan 2009 to 30th Dec 2011.

5 Experimental Results

In our experiment, the genetic algorithm (GA) was launched 14 times. For each of the seven given time series we created 25 populations of long agents and 25 populations of short agents. The fittest long agent and the fittest short agent for a particular time series are the solutions found by the GA search heuristics.

Table 4
Agents with the best fitness

Type of agent	Agent (logical formula of the prediction algorithm)
AA long	(H[1] >= C[4]) AND (O - L[1] <= H[4] - H[2])
AA short	(O - L[2] > H[2] - O[2]) OR (C[1] - H[3] >= H[2] - H[4])
BA long	(L[2] - H[3] > H[2] - C[5]) OR (L[2] - L[3] > H[1] - C[5])
BA short	(C[3] - C[4] > C[4] - L[1]) AND (C[2] - C[4] >= H[1] - H[2])
CAT long	(H[2] >= H[4]) AND ((L[3] - L[4] > O[4] - L[5]) OR (H[3] < H[5]))
CAT short	(H - C[4] < O[3] - O[1]) AND (H[4] > H[2])
DIS long	(C[5] - C[4] > O - L[1]) OR (H[1] - L[5] <= L[5] - L[2])
DIS short	(L >= L[4]) AND (C[4] - O[4] >= O[2] - C[3])
GE long	(H[4] <= O[2]) AND (L[4] - L[1] <= O[2] - H[4])
GE short	(H - C[1] <= L[3] - L[1]) OR (C[4] >= H[2])
IBM long	(L[3] - C[1] < C[1] - H[2]) OR (L - C[2] >= O[4] - L[3])
IBM short	(H[2] - L[1] >= H - O[4]) AND (L[4] - H[5] >= O - H[3])
KO long	(O[3] - H[4] >= C[5] - H[1]) AND (L[2] >= O[3])
KO short	(L[5] >= O[3]) AND (H - L[3] > C - H[4])

Table 4 list the agents (logical formulas of prediction algorithms based on candlestick patterns) which have gained the highest fitness function values on the in-sample periods of a given time series. These agents are evaluated both on in-sample and on out-of-sample periods of a given time series using the benchmark according to expressions (3), (4), and (5). The results of the evaluation are shown in Table 5.

Table 5
Benchmark results of the fittest agents both on in-sample (INS) and on out-of-sample (OOS) periods

Type of agent	Geometric mean		Variation coefficient	
	INS	OOS	INS	OOS
AA long	1.00**266**	1.00**143**	1.984	2.646
AA short	1.00**195**	1.00**210**	2.149	2.645
BA long	1.00**132**	1.00**018**	2.024	1.884
BA short	1.00**125**	1.00**050**	1.737	1.500
CAT long	1.00**060**	1.00**242**	1.734	1.957
CAT short	1.00**021**	1.00**296**	1.997	2.240
DIS long	1.00**158**	1.00**099**	1.968	1.605
DIS short	1.00**113**	1.00**008**	1.783	1.318
GE long	1.00**045**	0.99858	1.694	1.611
GE short	1.00**103**	1.00**462**	1.989	2.293
IBM long	1.00**070**	1.00**286**	1.509	1.111
IBM short	1.00**110**	0.99974	1.714	1.246
KO long	1.00**148**	1.00**068**	1.274	0.897
KO short	1.00**073**	1.00**031**	1.511	1.062

The INS values of geometric mean and variation coefficient benchmark the quality of the fittest agents resulted from the GA. The same INS time series that is used for the training of the multi-agent system is also used for the calculation of the benchmark.

The OOS values of the geometric mean and the variation coefficient benchmark the same agents but now operating in an **unknown environment**. The unknown environment for the agents is the OOS period of given time series.

By comparing the INS and OOS values in Table 5 we can assess the **quality of** proposed stock price **prediction algorithms** (i.e. agents) found by the GA. Some of the agents work better in the evaluation than in the training, that is, 'AA short', 'CAT long', 'CAT short', 'GE short', and 'IBM long'.

The OOS geometric mean of the 'AA short' agent is greater than the INS one. That means the average profit per trade reached in the evaluation period is greater than the one reached at the end of the GA evolution. The value of 1.00210 at 'AA short' OOS results means the average profit of the set of trades done on the AA daily time series is 0.21% per trade.

Conclusions

We apply genetic algorithms (GA) to construct candlestick pattern logical formulas. Then we use the constructed patterns as stock price prediction algorithms. The quality of prediction algorithms is measured by a newly designed stock price prediction algorithms benchmark.

Both in-sample and out-of-sample benchmark results confirm that some of genetically evolved agents (i.e. prediction algorithms based on candlestick pattern logical formulas) are doing well at the stock price time series prediction.

The fittest short agent for the GE stock price time series resulting from the GA gained the highest benchmark evaluation as for the out-of-sample test. The value of 1.00462 means the average profit of trades done by 'GE short' from January 2009 to December 2011 is 0.46% per trade.

The GA we have used was easy to converge, although the GA parameters were not necessarily optimal. Our experiment has demonstrated the feasibility of using GA for stock price prediction algorithms construction.

Acknowledgement

The author wishes to thank Prof. Jozef Kelemen from the Institute of Computer Science, Silesian University in Opava for consultations on agents and multi-agent systems, Assoc. Prof. Petr Sosík from the Institute of Computer Science, Silesian University in Opava for his supervision and reading several versions of the manucript, and Prof. Dušan Marček from the Institute of Computer Science, Silesian University in Opava for consultations on financial time series forecasting.

This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070). Research was partially funded by the Silesian University in Opava under the Student Funding Scheme, project no SGS/7/2011.

References

[1] Allen, F., Karjalainen, R.: Using Genetic Algorithms to Find Technical Trading Rules. *Journal of Financial Economics* **51** (1999) 245-271

[2] Bauer, R. J.: *Genetic Algorithms and Investment Strategies*. Wiley, New York, 1994

[3] Box, G. E. P., Jenkins, G. M., Reinsel, G. C.: *Time Series Analysis: Forecasting and Control*. Wiley, New York, 2008

[4] Chen, J.-S.: Trading Strategy Generation Using Genetic Algorithms. *Asian Journal of Information Technology* **4** (2005) 310-322

[5] Chen, S.-H.: *Genetic Algorithms and Genetic Programming in Computational Finance*. Springer, Berlin, 2002

[6] Chen, Y., Mabu, S., Shimada, K., Hirisawa, K.: A Genetic Network Programming with Learning Approach for Enhanced Stock Trading Model. *Expert Systems with Applications* **36** (2009) 12537-12546

[7] Deboeck, G.: *Trading on the Edge*: *Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets*. Wiley, New York, 1994

[8] Graham, B., Zweig, J.: *The Intelligent Investor: The Definitive Book on Value Investing*. Harper Business Essentials, New York, 2003

[9] Jakel, M.: Stock Investment Recommendation Making Supported by Case Based Reasoning. In: Kelemen, J., Nehéz, M.: *Proceedings of 6th International Workshop on Knowledge Management, Bratislava 2011*. Vysoká škola managementu, Bratislava, 2011

[10] Kelemen, J.: Agents from Functional-Computational Perspective. *Acta Polytechnica Hungarica* **3** (2006) 37-54

[11] Kelemen, J., et al.: *Knowledge in Context: Few faces of the knowledge society*. Iura Edition, Bratislava, 2010

[12] Koza, J. R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, 1992

[13] Lee, C.-H., Liu, A.: A Financial Decision Supporting System Based on Fuzzy Candlestick Patterns. In: *Proceedings of 9th Joint Conference on Information Sciences*. Atlantis Press, Taiwan, 2006, pp. 1289-1292

[14] Lee, C.-H., Liaw, Y.-C., Hsu, L.: Investment Decision Making by Using Fuzzy Candlestick Pattern and Genetic Algorithm. In: *Proceedings of 2011 IEEE International Conference on Fuzzy Systems*. IEEE, Taiwan, 2011, pp. 2696-2701

[15] Marček, M., Pančíková, L., Marček, D.: *Ekonometria a soft computing* (in Slovak). EDIS, Žilina, 2008

[16] Marček, M.: *Viacnásobná štatistická analýza dát a modely časových radov v ekonómii* (in Slovak). Silesian University, Opava, 2009

[17] Marshall, B. R., Young, M. R., Rose, L. C.: Candlestick Technical Trading Strategies: Can They Create Value for Investors? *Journal of Banking & Finance* **30** (2006) 2303-2323

[18] Mitchell, M.: *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, 1998

[19] Nison, S.: *Japanese Candlestick Charting Techniques: A Contemporary Guide to the Ancient Investment Techniques of the Far East*. 2nd edition, New York Institute of Finance, New York, 1991

[20] Williams, L.: *Long-Term Secrets to Short-Term Trading*. Wiley, New York, 1999

[21] Wooldridge, M.: *An Introduction to MultiAgent Systems*. 2nd edition. Wiley, New York, 2009

[22] Yahoo! Finance [Online] Available: http://finance.yahoo.com/ [2012/01/20]