International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 4 (2012) pp. 366–373

Neighborhood Evaluation in Acquiring Stock Trading Strategy Using Genetic Algorithms

Kazuhiro Matsui and Haruo Sato

Department of Computer Science, College of Engineering, Nihon University,
1 Nakagawara, Tokusada, Tamura-machi, Koriyama, 963-8642, Japan
matsui@cs.ce.nihon-u.ac.jp

Abstract: **We propose a new method to evaluate individuals in genetic algorithms (GAs) for algorithmic trading in stock markets. In our previous work, we presented an effective method to acquire trading strategy in stock markets. However, it had a tendency of overfitting in genetic searches. Our new approach, namely *neighborhood evaluation*, involves evaluation for neighboring points of genetic individuals in fitness landscape as well as themselves. Empirical results for trading simulation in the first section of the Tokyo Stock Exchange for recent eleven years show the effectiveness of the neighborhood evaluation for reducing the overfitting tendency. We discuss suitable forms of neighborhoods on the performance of the genetic searches. We also propose a new method to reduce the computational cost of our method, because the neighborhood evaluation has a disadvantage on the cost.**

Keywords: genetic algorithm, algorithmic trading, neighborhood evaluation, overfitting, fitness landscape

I. Introduction

In recent financial markets, automated trading methods, which are often referred as *algorithmic trading*, are becoming increasingly widespread. Many works are found in applications of computational intelligence methodologies in finance [1]. Evolutionary computation, such as genetic algorithm (GA)[2], is promising in these methodologies, because of their robustness, flexibility and powerful ability for search.

Various works have been done on automated trading using evolutionary computation (*e.g.* [3],[4],[5],[6],[7] and [8]). These methods used mainly *technical analysis*, which is one of the two basic approaches in trading methods. Technical analysis is an attempt for the forecast of the future direction of prices by analyzing past market data, such as price and volume.

In our previous work [8], automated trading using GAs has a tendency that difference of earned profits is large between training and testing. This tendency implies overfitting in the training phase. In other words, when a kind of anomalous point causes large profit in the training, the GA fits the point excessively. The objective of our study is to dissolve the overfitting problem in the GA training.

For this problem, we propose a new evaluation method, namely *neighborhood evaluation*. This method involves evaluation for neighboring points of a genetic individual in fitness landscape as well as itself. The aim of our method is to reduce the influence of the anomalous points in the training phase and to improve the profit in the testing phase. In this paper, we apply this method to automated trading in stock markets and verify the effectiveness of this method.

This paper is organized as follows: We summarize the concept of technical analysis in financial markets and the genotype representations for algorithmic trading in Section II. In Section III, we discuss the overfitting problem in automated trading with GAs. Section IV describes the details of neighborhood evaluation. In Section V, we show our trading method, and the empirical results are followed in Section VI. We discuss the effectiveness of our method based on the results in Section VII, and conclude this work in Section VIII.

II. Algorithmic Trading using GAs

A. Technical Analysis

Two basic approaches to analyze financial markets are known widely: technical analysis and fundamental one [1]. The former is based on the past changes of prices and volume. The latter is based on analyzing financial statements, management, and competitive advantages of companies.

Our approach is the technical analysis. It needs various indicators for trading. They are evaluated from past stock data. Generally, technical indicators have several parameters. For example, moving average has a parameter, namely *period*, which is used as the denominator of averaging calculation. We can define various derived indicators with the parameter, such as *10-day moving average*, *50-day moving average*, *etc.*. In this paper, we use many technical indicators and their parameters for automated trading in stock markets.

Many technical indicators have been proposed, but it is hard to select optimal indicators for actual algorithmic trading. Furthermore, it is also difficult to determine parameters for the selected indicators. In this paper, GAs are applied for this problem. We encode technical indicators and their parameters on chromosomes in GAs and apply the GA search for acquiring appropriate combinations of technical indicators and their parameters.

The flow of our trading method is shown in Figure 1. First, we apply the GA using stock price data for a pre-determined period. This is the training phase to extract a set of effective technical indicators and their parameters. Next, we try an

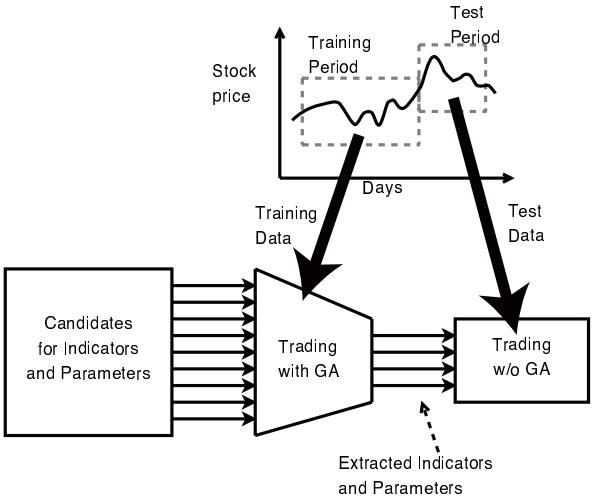

Figure. 1: Flow of trading using GA.

automated trading with the obtained set of technical indicators and their combined parameters using another stock price dataset. This is the testing phase to examine the performance of the selected indicators and their parameters. Note that we do not apply the GA in the testing phase. It is important to prevent the overfitting in the training phase for improvement of the performance in the test.

B. Genotype Representation

1) Parameter Encoding

Binary coding has been widely used to encode parameters on chromosomes in GAs, such as de la Fuente [6] and Hirabayashi [7]. However, this coding involves a shortcoming. Generally, binary coded GAs divide their search-ranges at regular intervals and assign each divided value to each binary code. It is often not suitable to divide the range at regular intervals. For examples, suppose that a binary coded GA searches the period of the moving average of prices. As shown in Figure 2, in comparing among shorter periods, such as five and six days, the difference of these periods may cause different profits in short-term trading. On the other hand, in comparing among longer periods, such as 100 and 101 days, it will be expected that the difference of profits is very little although the difference of these periods is same as one day. Thus, the simple binary coding is not always suitable for the search of parameters. We refer this conventional method of binary coding as *the direct coding* in the following sections.

Another coding method of parameters is *the indirect coding* (Matsui [8]). Generally, there often are a small set of particular values which are widely accepted to evaluate technical indicators. The indirect coding restricts the range of the genetic search to the small set of these discrete values. For examples, in the above case of the moving average of prices, we are able to restrict the search-range to a set of values $\{5, 10, 20, 50, 100\}$. This method makes the search space to be smaller than the conventional one. It have been reported that the indirect coding improved the performance of algorithmic trading in comparing with the direct coding [8].

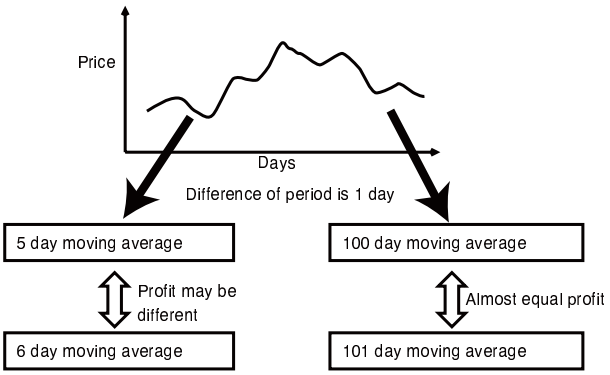

Figure. 2: A shortcoming of binary-coded GAs.

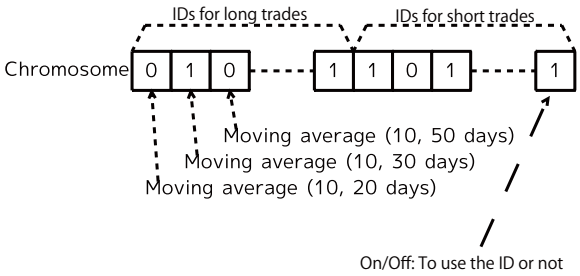

Figure. 3: Locus-based representation.

2) Indicator Representation

Representation methods of technical indicators are divided roughly into two categories [8]. The first is the locus-based representation, as shown in Figure 3. This method assigns a technical identifier (ID) to each locus, which is a bit-position on chromosomes. The ID is a combination of a technical indicator and its parameters which are represented in the indirect coding. For example, the second bit in Figure 3 is assigned to the ID *"the crossover of moving average between ten and thirty days."* When the assigned bit is "on," the corresponding ID is applied for trading. Note that technical indicators and their parameters are combined on bits and the parameters are encoded in the indirect coding.

The second type of genotype representation is the allele-based one. Figure 4 shows its concept. Each locus takes a value as an allele. In the allele-based representation, the allele takes various IDs which represent technical indicators and their combined parameters in the indirect coding. For example, Allele #1 is assigned to the ID *"the crossover of moving average between ten and twenty days"* in Figure 4.

In our comparison [8], the allele-based representation realized better performance than the locus-based one. Therefore, we adopt the allele-based one in this paper.

III. Overfitting in Conventional Methods

In our previous work [8], we have proposed a method to acquire reasonable strategy for automated stock trading. However, this method still has some problems. One of the most significant problems is the difference of profits between the training and the testing phases. The profit of the former is obtained in the GA search and the latter is earned in the automated trading by the acquired strategy. Generally, the latter

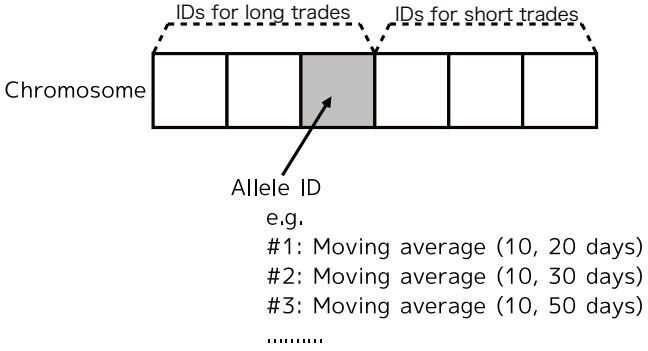

Figure. 4: Allele-based representation.

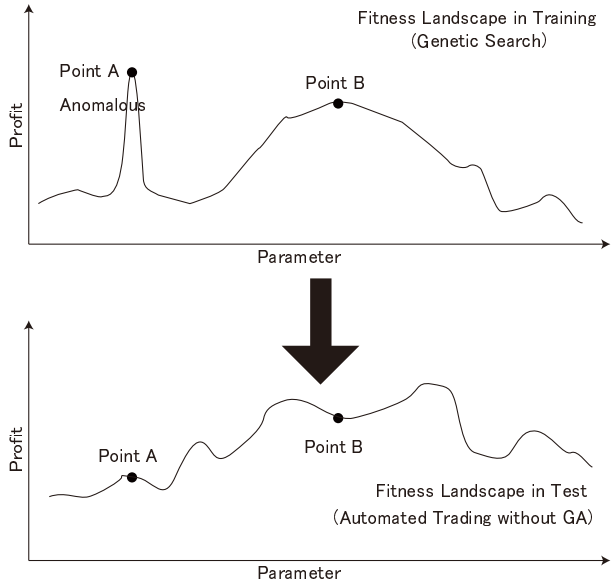

Figure. 5: Overfitting and fitness landscapes.

is much lower than the former. This difference implies a kind of overfitting in the training phase. The acquired strategy has a tendency not to match the stock price data in the testing phase because this strategy has matched the data excessively in the training.

One reason of the overfitting is the influence of a kind of anomalous point, as shown in Figure 5. In this paper, we refer to a point, which has extremely larger profit than its neighboring points, as *anomalous*. Point A is anomalous in Figure 5 (upper). In maximizing problems using ordinary GAs, it is sufficient to find the best point, which is Point A in this figure. However, in automated trading, the fitness landscapes are usually different between the training and the testing phases, because the price datasets are different in the both phases. Therefore, the optimal point in the GA search is not always the best solution in the testing phase. This causes the difference of profits between the training and the test. Figure 5 shows the difference. The upper part of this figure is an example of the fitness landscape in the training, and the lower is that in the test. The profit on Point A is much lower than the highest point on the fitness landscape of the test.

IV. Neighborhood Evaluation

A. Evaluation

We propose a new evaluation method, which is referred as the *neighborhood evaluation*, for the overfitting problem. In Figure 5 (upper), since Point B has a lower profit than Point A, conventional GAs do not select Point B for the solution. However, the neighborhood of Point B is *gradual* while that of Point A is *steep*. It is expected that the gradual profile is better than the steep one in order to keep the profit on the test because the large profit on the steep profile may be obtained by anomalous points.

From this point of view, we propose the neighborhood evaluation, which considers the neighboring points as well as the point of interest. In our previous work [8], we evaluated the fitness f_i in a point i as follows:

$$f_i = P_i^{long} + P_i^{short} \tag{1}$$

where P_i^{long} and P_i^{short} are the total sum of profit for long and short trades respectively.

On the other hand, the fitness F_i in the neighborhood evaluation is obtained by Eq. (2):

$$F_i = F_i^{long} + F_i^{short} \tag{2}$$

where F_i^{long} and F_i^{short} are the evaluation of neighborhoods for long and short trades. F_i^{long} is calculated as follows:

$$F_i^{long} = \sum_{j \in \mathrm{IndexSet}^{long}} \frac{g_{j,i}^{long}}{\bar{\eta}_{j,i}} \tag{3}$$

where $\mathrm{IndexSet}^{long}$ is a set of IDs which are involved in the genetic individual for long trades, and j is an element of the set, *i.e.*, an ID.

$$g_{j,i}^{long} = \sum_{k \in N_{j,i}} \eta_{k,i} P_k^{long} \tag{4}$$

$$\bar{\eta}_{j,i} = \sum_{k \in N_{j,i}} \eta_{k,i} \tag{5}$$

where $N_{j,i}$ is a set of neighbors of the point i on ID j, and $N_{j,i}$ also includes the point i itself. $\eta_{k,i}$ is a weight which represents the relation between the points i and k. $\bar{\eta}$ normalizes the effect of η, and P_k^{long} is the total sum of profit in the neighbor k on long trades.

B. Related Works

Several similar methods to the neighborhood evaluation have been proposed. Matsui *et al.* [9] proposed the macroscopic fitness which averages neighboring individuals in a parallel GA for medical image processing. Nikolaev *et al.* [10] proposed another method to evaluate fitness as an average of neighbors in a genetic programming. Wang *et al.* [11] also proposed a similar method to evaluate fitness as a weighted sum of similarity of neighbors for auto-body panel die-design using a GA.

The fitness evaluation using neighboring points in the landscape is similar between these methods and the neighborhood evaluation. However, the aims of the evaluation are

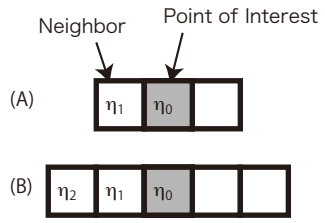

Figure. 6: One-dimensional neighborhoods.

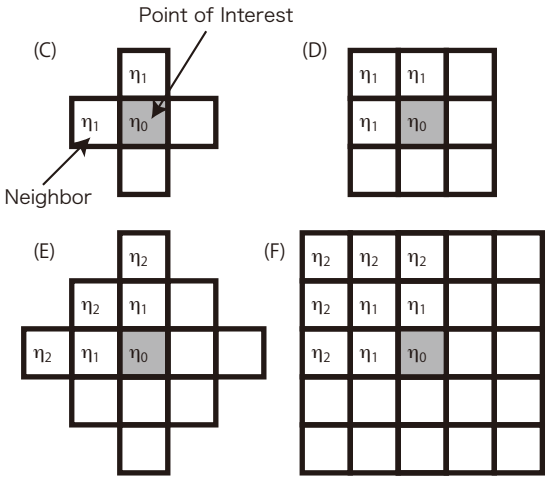

Figure. 7: Two-dimensional neighborhoods.

different. The neighborhood evaluation aims for dealing with the difference of the landscapes between the training and the test, whereas the purpose of the other methods are the improvement of the performance of the genetic searches in *only* the training landscapes.

In financial applications, various methods have been proposed to acquire trading strategies using evolutionary computation technique, such as Hryshko [3], Dempster [4], Chen [5], de la Fuente [6], and Hirabayashi [7]. These methods in common evaluate fitness for a *single* point without neighbors on the landscapes in the genetic searches. On the other hand, the neighborhood evaluation involves the evaluation of neighboring points as well as the point of interest. This is the major difference between our method and the other related works in financial applications.

C. Forms of Neighborhoods

The performance of the neighborhood evaluation may depend on the forms of neighborhoods. In this paper, we compare several forms of neighborhoods in our experiments.

We use one and two dimensional IDs in our experiments. The former has only one parameter for evaluation, and the latter has two parameters. In one dimensional IDs, we compare two types of neighborhoods, A and B, as shown in Figure 6. The type A has three points in the fitness landscape and the type B has five. These points are used in fitness evaluation. Parameters η_0, η_1, and η_2 correspond to the weights $\eta_{k,i}$ in Eq. (4). They are determined in relative position to the point of interest. In two dimensional IDs, we also compare four types of neighborhoods, C, D, E, and F, as shown in Figure 7.

D. Computational Cost

Since our method needs evaluation for many points in the fitness landscape, it is anticipated that the computational cost increases. For this problem, we propose a method to reduce the cost.

Generally, evolutionary computation like GAs has a tendency to concentrate a few small promising areas in the landscape at the final phase of the search. In such cases, some points, which have been already evaluated earlier, are often evaluated repeatedly. Thus, by storing the fitness of such the *already-evaluated* points, we can reduce the redundant evaluation. This idea is similar to the *memory cache* in computer architecture.

The procedure of our caching method is the following:

1. Initialization:
   Initialize the list L_e to store a pair (i, F_i), which is a pair of an *already-evaluated* point i and its fitness F_i.

2. Evaluation:

   (a) Scan the searching point j in the list L_e. If the point is found, go to (b). Otherwise, go to (c).

   (b) Read the pair (j, F_j) from L_e, and then, the fitness F_j is assigned to the point j. In this case, the trading simulation on the point j is *not* executed.

   (c) Run the trading simulation on the point j, and evaluate the fitness F_j. Add the pair (j, F_j) to the list L_e

V. Methods

A. Overview

The overview of our system is shown in Figure 1. The flow of our method is the following:

1. Prepare a set of past stock prices and divide it to the training and the test subsets.

2. Apply the genetic search to find effective IDs for trading on the training subset.

3. Run the automated trading, without GA, for the test subset using the IDs found in Step 2.

Note that we apply the GA only on the training phase, not on the testing one.

This flow is applied in order to maximize the evaluation value on the test subset, *not on the training one*. Thus, it is important to keep off overfitting in the training.

The trading rules in our system are applied *daily*. When a rule is matched in a day, the system opens or closes a position *at the opening price on the next day*.

B. Technical Indicators

We apply the four types of technical IDs, as shown in Table 1, in the following:

1. Simple Moving Average Crossover (SMA)
   The SMA is a simple average of closing prices for the

Table 1: Technical IDs.

Indicators	*Parameters*
Simple Moving Average (SMA)	shorter period, longer period
Exponential Moving Average (EMA)	shorter period, longer period
Bollinger Band (BB)	period, factor
Price Channel Breakout (PCB)	period

last n days. We define the SMA as follows:

$$\mathrm{SMA}_n(t) = \frac{1}{n}\sum_{i=0}^{n-1} c_{t-i}, \tag{6}$$

where c_t is the closing price at the day t, n is the parameter which determines the period to calculate the SMA.
In our experiments, we apply the following rules: For a long trade, enter when a shorter-period SMA crosses a longer-period SMA and exit when the opposite occurs. A short trade is the contrary of the long one.
The SMA is a two-dimensional ID which has the two parameters: the shorter and longer periods.

2. Exponential Moving Average Crossover (EMA)
The EMA is an exponentially weighted average of closing prices for the last n days, as follows:

$$\mathrm{EMA}_n(t) = \begin{cases} \mathrm{SMA}_n(t) & (t=0) \\ \mathrm{EMA}_n(t-1) & \\ \quad + \alpha\left(c_t - \mathrm{EMA}_n(t-1)\right) & (t \geq 1), \end{cases} \tag{7}$$

where n is the period parameter, and $\alpha\,(=2/(1+n))$ is the weight. In our experiments, we use the same crossover rule as the above SMA for trades.
The EMA is also a two-dimensional ID which has the two parameters: the shorter and longer periods.

3. Bollinger Band (BB) [12]
The BB is an indicator based on the standard deviation of the change of prices, as follows:

$$\mathrm{BB}_n(t) = \mathrm{SMA}_n(t) \pm \beta\sigma, \tag{8}$$

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=0}^{n-1}\left(c_{t-i} - \mathrm{SMA}_n(t-i)\right)^2}, \tag{9}$$

where β is the factor of the standard deviation.
In our experiments, we apply the following rules: For a long trade, enter when the closing price crosses the upper line of the BB and exit when the closing price crosses $\mathrm{SMA}_n(t)$. For a short trade, enter when the closing price crosses the lower line of the BB and the exit rule is the same as the long one.
The BB is another two-dimensional ID which has the two parameters: the period n and the factor β.

4. Price Channel Breakout (PCB) [4]
The PCB is an ID based on the trading range of the last n days, as follows:

$$\mathrm{U}_n(t) = \max\{h_{t-i} | 1 \leq i \leq n\}, \tag{10}$$

$$\mathrm{L}_n(t) = \min\{l_{t-i} | 1 \leq i \leq n\} \tag{11}$$

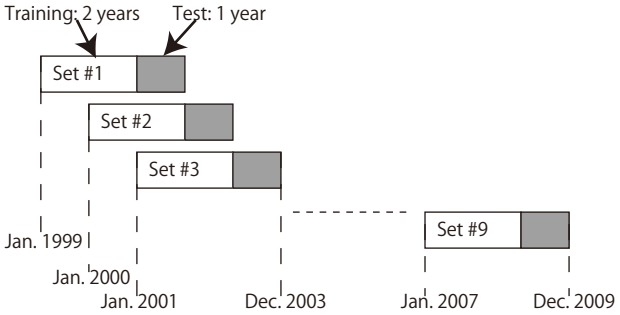

Figure. 8: Concept of sliding windows.

where $\mathrm{U}_n(t)$ is the upper bound of the PCB, $\mathrm{L}_n(t)$ is the lower bound, h_t is the highest price and l_t is the lowest one at the day t.
In our experiments, we apply the following rules: For a long trade, enter when $c_t > \mathrm{U}_n(t)$ and exit when $l_t < \frac{1}{2}\left(\mathrm{U}_n(t) + \mathrm{L}_n(t)\right)$. For a short trade, enter when $c_t < \mathrm{L}_n(t)$ and exit when $h_t > \frac{1}{2}\left(\mathrm{U}_n(t) + \mathrm{L}_n(t)\right)$.
The PCB is a one-dimensional ID which has one parameter: the period n.

VI. Experiments

A. Setups

We apply our method with eleven years of stock price data from the start day of 1999 to the end day of 2009. We select twenty companies at random from the components of the Nikkei 225, which is a stock market index for the Tokyo Stock Exchange, and these issues are used for our experiments.

A kind of sliding-window technique is applied for our experiments, as shown in Figure 8. Each window consists of training and test phases. The former is a genetic search for two years, and the latter is a trading simulation for one year just after the training. This test is applied to evaluate the set of IDs which are selected by the GA search. The total size of the windows is nine.

The initial principal is 5,000,000 JPY. The trading unit is minimal, *i.e.*, a round lot of each issue. A commission of one trade is assumed at 1,000 JPY.

The technical IDs applied in our experiments are described in the previous subsection. Each technical IDs are calculated from *daily* prices. The period for each ID in the indirect coding takes a value from a set of $\{5, 10, 15, 20, 25, 30, 50, 75, 100, 200\}$, and the factor for the Bollinger Band also takes a value from a set of $\{1.0, 1.5, 2.0, 2.5, 3.0\}$. Since SMA has two parameters, 45 IDs are defined as *5-and-10 day SMA, 5-and-15 day SMA, ..., 100-and-200 day SMA*. In same ways, 45 EMAs, 50 BBs and 10 PCBs are also defined. Thus, the total number of IDs is 150. In our genotype coding, each ID can be applied for *long* and *short* positions. Therefore, the total size of IDs is 300 finally in the indirect coding.

The setups of our GA are the following: The population size is 50 and the searching generation is 5,000. The minimal generation gap model [13], the uniform crossover and the random-replace mutation are applied.

We compare two types of fitness evaluation methods: the

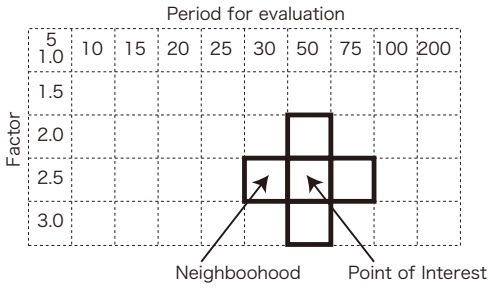

Figure. 9: Two dimensional neighborhood [Type C]: BB

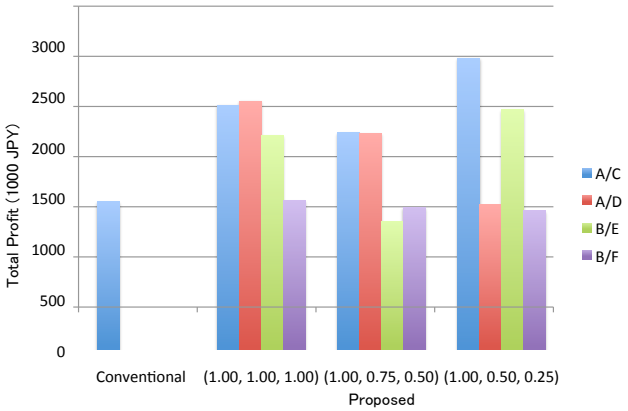

Figure. 10: Comparison of the total profit.

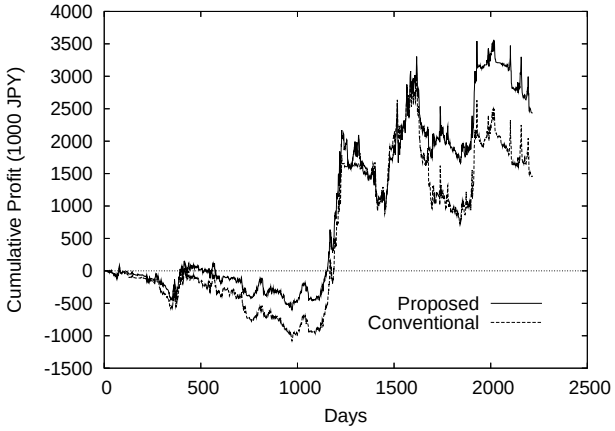

Figure. 11: Cumulative profit.

conventional and the neighborhood evaluations. In the former, the fitness of each individual is the profit obtained *without the influence of neighborhood points* in the fitness landscape. In the latter, the three categories of IDs which have two types of parameters, *i.e.*, SMA, EMA, and BB, in Table 1, are evaluated in two dimensional neighborhoods. For example, the neighborhood type C for the BB is shown in Figure 9. On the other hand, the PCB has only one parameter. In our experiments, we compare three sets of weights (η_0, η_1, η_2) in Figures 6 and 7: $(1.00, 1.00, 1.00)$, $(1.00, 0.75, 0.50)$, and $(1.00, 0.50, 0.25)$.

B. Results

First, we compare the neighborhood evaluation with the conventional evaluation, in the three sets of weights (η_0, η_1, η_2). Table 2 shows the empirical results from our experiments through the nine windows. This table summarizes the three sets of weights: $(1.00, 1.00, 1.00)$, $(1.00, 0.75, 0.50)$, and $(1.00, 0.50, 0.25)$.

In each of the experiments, we applied ten runs whose seeds of random numbers were different each other and removed the best and the worst runs from the ten. Then we averaged the remaining eight runs.

In Table 2, *Neighborhood* shows the forms of the neighborhoods. For example, *A/C* is "the type A for one-dimensional ID and the type C for two-dimensional one." *Num. of trades* is the number of trading in the testing phase. *Total profit* is the average of total profit in the tests in the eight runs, and *Profit rate* is the rate of the total profit to the initial principal. *Max draw down* is the worst loss from a consecutive loss trades. The profit and the draw down are represented in 1,000 JPY in all the tables in this paper. We show the ratio of earned profits between the testing and the training phases in the bottom row of Table 2. Since the training was applied for two years and the testing was done for one year, the former results were converted into the value for one year, and then, we calculated the ratio from the converted values.

Figure 10 is a summary of the total profit in Table 2. From Figure 10 and Table 2, it turns out that the neighborhood evaluation improves the total profit drastically for the neighborhood types A/C. The test/train ratio has also been improved. In the types A/D and B/E, the total profit depends on the weight η. Only in the types B/F, the profit of the proposed method is almost equivalent to the conventional one.

We also show the progress of cumulative profits of the two methods in Figure 11. In this plot, the *proposed* line is the types A/C for the weights $(1.00, 1.00, 1.00)$. Although the both methods suffered loss in earlier days, the loss was recovered later.

The next experiment is the comparison of the computational costs. Table 3 shows the results. The rows of *Time* are the CPU time to execute the genetic search and the trading simulation. These values are written in seconds. *Reduction ratio* is the ratio of the CPU time between *with-cache* and *without-cache*. *Cache hit* is the ratio of the hits in the cache. We use the weight set $(1.00, 1.00, 1.00)$ in this experiments, and our CPU is Intel Xeon 2.66GHz. From this result, the neighborhood evaluation can reduce the computational cost remarkably by the fitness caching.

VII. Discussion

From Table 2 and Figure 10, it turns out that the neighborhood evaluation improves the performance of trading. In particular, the total profit and the test/train ratio for the neighborhood types A/C are much superior to the conventional one.

The improvement of the test/train ratio means that the neighborhood evaluation can reduce the overfitting in the training. In order to confirm it, we analyze our trading simulation in detail. In Figure 11, the profile of both lines are very similar. However, the difference expands in the progress of trading days. In particular, it happens around the days 1600 to 1700. Since these days correspond to the year 2007, we examine the acquired sets of IDs on this window, in which the GA search is applied from 2005 to 2006 and the test is done in 2007. Table 4 summarizes the profit on this window.

Table 2: Trading Results

Methods	Conventional	Neighborhood Evaluation											
Weight	-	$(1.00, 1.00, 1.00)$				$(1.00, 0.75, 0.50)$				$(1.00, 0.50, 0.25)$			
Neighborhood	-	A/C	A/D	B/E	B/F	A/C	A/D	B/E	B/F	A/C	A/D	B/E	B/F
Num. of trades	718	614	632	625	582	624	653	614	651	665	574	614	634
Total profit	1457	2410	2450	2110	1469	2144	2136	1265	1400	2872	1431	2366	1368
Profit rate (%)	29.1	48.2	49.0	42.2	29.4	42.9	42.7	25.3	28.0	57.4	28.6	47.3	27.4
Max draw down	822	820	716	783	704	743	736	698	778	675	652	727	768
Test/Train ratio (%)	10.8	18.9	19.1	16.6	11.7	16.7	16.6	10.0	11.1	22.6	11.5	18.5	10.5

Table 3: Computational Costs

Methods	Conventional	Proposed			
Neighborhood	-	A/C	A/D	B/E	B/F
Time: w/o Cache	150	3420	3202	8007	7730
Time: w/ Cache	93	1544	1420	3579	2830
Reduction ratio (%)	38.0	54.9	55.7	55.3	63.4
Cache hit (%)	56.8	55.8	56.6	56.6	58.7

Table 4: Summary of the training in 2005-06 and the test in 2007.

Methods	Proposed	Conventional
Profit in Test	421	-709
Profit in Training	2,131	2,348

Table 5: The selected IDs in the training of 2005-06 and for the test in 2007.

Trade type	Long		Short	
	ID	Count	ID	Count
Proposed	BB 30-2.0	1	SMA 10-75	1
	BB 50-2.0	1	EMA 5-10	1
	BB 100-1.0	1	EMA 10-50	2
	BB 100-1.5	2	EMA 20-50	5
	BB 100-2.0	1	BB 15-2.5	1
	PCB 100	2	BB 50-1.5	5
Conventional	EMA 20-25	2	SMA 5-75	2
	BB 100-1.0	5	SMA 20-75	1
	BB 100-1.5	1	SMA 25-30	1
			SMA 25-50	1
			SMA 25-75	1
			SMA 30-50	1
			EMA 15-50	2
			EMA 20-50	2
			EMA 25-30	1
			EMA 25-50	1
			BB 50-1.5	2
			BB 100-1.5	1

While the both methods obtained almost equivalent profits in the training, the difference of the two methods was very large in the test. In order to survey the cause of the loss in the conventional method, we inspect the IDs selected in our experiments, as shown in Table 5. The *count* in this table means the number of occurrence of the ID in our experiments. For long trades, the conventional method has selected the *BB 100-1.0* (the period is 100 days and the factor is 1.0) for *five* times in the *eight* runs. However, the proposed method has selected the same ID in only *one* run. Thus, we investigate the influence of this ID in the training and the testing phases. In the training of 2005-06, the *BB 100-1.0* earned 4,197 thousand JPY in our GA searches. This ID was selected by the conventional method because of this large profit. In the test of 2007, however, this ID caused the loss of 920 thousand JPY. This is the reason of the loss for the conventional method in Table 4. On the other hand, the proposed method earned the slightly lower profit in the training than the conventional one in this table. The neighborhood evaluation yielded the difference of the profits because this method selected IDs with low influence of anomalous points even if the profit of the IDs was lower than other IDs in the GA training. Since the influence of anomalous points was reduced, the larger profit was obtained than the conventional one in the testing phase, as shown in Table 4.

Next, we discuss the difference of the forms of the neighborhoods. From Table 2 and Figure 10, we can find a tendency that the smaller neighborhood realized better results than the larger one. For one-dimensional neighborhoods, the type A is generally superior to the type B, and the type C is almost always better than the other types D, E, and F for two-dimensional neighborhoods. The only one exception is the type A/D for the weights $(1.00, 0.50, 0.25)$. The types A and C are the smallest forms in one and two dimensional IDs, respectively. This implies that smaller neighborhoods are promising than larger ones.

On the other hand, there is no obvious tendency on the weight η. Therefore, we have to consider further analysis on the weight.

Finally, we discuss the computational cost of our method. Since the proposed method needs evaluations on many neighboring points, the cost is very high in the neighborhood evaluation. This is a shortcoming of the method. From Table 3, however, the fitness caching is very effective to reduce the costs. The CPU time with the cache is less than the half of that without the cache.

VIII. Conclusions

We proposed the neighborhood evaluation to reduce overfitting in acquiring stock trading strategy using genetic algorithms, and we compared the proposed method with the conventional one. In our experiments, the neighborhood evaluation with small neighborhoods outperformed the conventional one in the earned profit. The test/train ratio was also improved and the overfitting problem was reduced. However,

the computational cost was a shortcoming of our method. Thus, we proposed the fitness caching to reduce the cost. In our experiments, this method was very effective to reduce the computational time. We also compared the forms of neighborhoods. The empirical results implied smaller neighborhoods were appropriate than larger ones.

Future problems are the following: Since our discussion of the forms of neighborhoods are preliminary, we need deeper analysis on this issue. The optimal value of the weights on the neighbors is open. Also, it is necessary to add other technical indicators since we used only four types of indicators in our experiments.

Acknowledgment

This work was partially supported by the Grant-in-Aid for Scientific Research (C) 20500215, Japan Society for the Promotion of Science.

References

[1] A. Brabazon and M. O'Neill, An Introduction to Evolutionary Computation in Finance, *IEEE Computational Intelligence Magazine*, 3(4), pp. 42-55, 2008.

[2] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Boston, 1989.

[3] A. Hryshko and T. Downs, An Implementation of Genetic Algorithms as a Basis for a Trading System on the Foreign Exchange Market. In *Proceedings of the Congress of Evolutionary Computation*, pp. 1695-1701, 2003.

[4] M.A.H. Dempster and C.M. Jones, A Real-time Adaptive Trading System Using Genetic Programming, *Quantitative Finance*, 1(4), pp. 397-413, 2001.

[5] S.H. Chen and C.F. Chen, Statistical Analysis of Genetic Algorithms in Discovering Technical Trading Strategies, *Advances in Econometrics: Applications of Artificial Intelligence in Finance and Economics*, 19, pp. 1-43, 2005.

[6] D. de la Fuente, A. Garrido, J. Laviada and A. Gomez, Genetic Algorithms to Optimise the Time to Make Stock Market Investment. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1857-1858, 2006.

[7] A. Hirabayashi, C. Aranha and H. Iba, Optimization of the Trading Rule in Foreign Exchange using Genetic Algorithms. In *Proc. of the 2009 IASTED Int'l Conf. on Advances in Computer Science and Engineering*, 2009.

[8] K. Matsui and H. Sato, A Comparison of Genotype Representations to Acquire Stock Trading Strategy Using Genetic Algorithms. In *Transactions on Computational Science VIII, Lecture Notes in Computer Science*, 6260, pp. 56-70, 2010.

[9] K. Matsui and Y. Kosugi, The Effect of Regularization with Macroscopic Fitness in Genetic Approach to Elastic Image Mapping, *IEICE Transactions on Information and Systems*, E81-D5, pp. 472-478, 1998.

[10] N.I. Nikolaev and V. Slavov, Inductive Genetic Programming and the Superposition of Fitness Landscapes. In *Proceedings of the 7th International Conference on Genetic Algorithms*, pp. 97-104, 1997.

[11] D. Wang, H. Zhu, and X. Liu, An Application of Genetic Algorithm for Auto-body Panel Die-design Case Library Based on Grid. In *Proceedings of the 5th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*, pp. 43-47, 2006.

[12] J.A. Bollinger, *Bollinger on Bollinger Bands*, McGraw-Hill, New York City, 2001.

[13] H. Satoh, M. Yamamura and S. Kobayashi, Minimal Generation Gap Model for GAs Considering Both Exploration and Exploitation. In *Proceedings of the 4th International Conference on Soft Computing*, pp. 494-497, 1996.

Author Biographies

Kazuhiro Matsui received his Dr. Eng. degree from Tokyo Institute of Technology, Japan, in 1997. He is currently a lecturer at College of Engineering, Nihon University, Japan. His research interests include evolutionary computation, pattern recognition, and computational intelligence.

Haruo Sato received his Dr. Eng. degree from Yokohama National University, Japan, in 1992. He is currently a professor at College of Engineering, Nihon University, Japan. His research interests include intelligent systems, decision making, and artificial intelligence.