# Can We Believe That Genetic Algorithms Would Help without Actually Seeing Them Work in Financial Data Mining?: Part 2, Empirical Tests*

Shu-Heng Chen[1] and Chien-Fu Chen[2]

[1] National Chengchi University, Taipei, Taiwan 11623
[2] National Chengchi University, Taipei, Taiwan 11623

**Abstract.** By using the data of Taiwan stock index (**TAIEX**), we conducted two primitive tests for the *no-free-lunch* (**NFL**) *property* and the *temporal stability of landscapes* over the space of trading strategies. 100,000 trading strategies are randomly generated as a sub-domain on which the random variable $r_s^t$ is defined. From the 20 possible cases studied in this paper, we found no evidence for the existence of the NFL property. Also, based on the temporal correlation coefficients computed, the landscape is rather dynamic, and there is no simple pattern for its dynamics.

## 1 Motivation and Introduction

Chen (1998) (Part I of this series) gave three important properties which can be used to predict the performance of GAs. They are the *NFL property*, the *well-ordered property* and the *existence of temporal correlation* (*the temporal stability of landscapes*). Whether or not GAs or, more precisely, a particular style of GAs, can be helpful in financial data mining crucially depends on the validation of these properties as stated in Theorems 1, 3, 6, 9, and 10 in Chen (1998). The purpose of this paper is then to give an empirical test for two out of these three properties, namely, the **NFL** property and the *temporal stability of landscapes*. To our best knowledge, the first one has never been addressed in the literature. The second one, while frequently discussed in the literature, has never been tested by any empirical measure. It is our belief that without an advanced understanding of these properties, machine learning will remain a *black box* in financial data mining.

## 2 Data Description

The raw data used in this paper are daily trading volume ($V_t$) and the highest ($H_t$), lowest ($L_t$), and closing price ($C_t$) of the daily stock index. The daily stock index used in this paper is the Taiwan Stock Price Index (TAIEX). The sampling

* Research support from MasterLink Securities Corporation is gratefully acknowledged.

period starts from May 15, 1989 to Sept. 27, 1997. There are totally 2400 observations in this sample. From this data set, six technical indices are generated. Among them, VPT, MAV and EOM are volume-related and MAP, KD-RSI, and MACD are price-related. These six technical indices are further used to generate other indices, and the whole data preprocessing process is summarized as follows.

$$\overrightarrow{\mathbf{X_t}} = \begin{bmatrix} \{V_t\} \\ \{H_t\} \\ \{L_t\} \\ \{C_t\} \end{bmatrix} \longrightarrow \overrightarrow{\mathbf{Y_t}} = \begin{bmatrix} \{VPT_t\} \\ \{MAV_t\} \\ \{EOM_t\} \\ \{MAP_t\} \\ \{KD-RSI_t\} \\ \{MACD_t\} \end{bmatrix} \longrightarrow \overrightarrow{\mathbf{Z_t}} = \begin{bmatrix} \{DMVPT_t\} \\ \{DMAV_t\} \\ \{DMAEOM_t\} \\ \{DMAP_t\} \\ \{DMARSI\%D_t\} \\ \{DEMA_t\} \end{bmatrix} \tag{1}$$

## 3 Encoding Trading Strategies with Genetic Algorithms

Each trading strategy considered in this paper has the following form:

```
( IF   (CONDS)
  THEN   (BUY)
  ELSE   ( IF   (CONDS)
             THEN   (SELL)
             ELSE   (HOLD)   )   )
```

The CONDS appearing in the trading strategy is a *predicate*. CONDS itself is a logical composition of several primitive predicates. In this paper, all CONDSes are composed of three primitive predicates. Each primitive predicate can be represented as:

$$Cond(Z) = \begin{cases} 1(True), & \text{if } \ Z \oplus z, \\ 0(False), & \text{if } \ Z \ominus z. \end{cases} \tag{2}$$

where $Z$ is an element of the time-series vector $\mathbf{Z}$ (Equation 1), and $z$ can be regarded as a *threshold* or *critical value* ($z \in \aleph$). $\oplus \in \{\geq, <\}$ and $\ominus = \{\geq, <\} - \oplus$. An example of CONDS with three primitive predicates is

$$CONDS(Z_1, Z_2, Z_3) = Cond(Z_1) \quad \vee ( \quad (Cond(Z_2) \wedge (Cond(Z_3) \quad ), \tag{3}$$

where "$\vee$" refers to the logic operator "OR", and $\wedge$ refers to "AND".

Based on the formulation above, to encode a trading strategy, we only need to encode the CONDS. And for a CONDS with three primitive predicates, that means the following three things:

– $\overrightarrow{\mathbf{z}} = (z_1, z_2, z_3)$,

**Table 1.** Binary Codes for Inequality Relation

| Logic Code (Binary Code) | $\oplus_1$ | $\oplus_2$ | $\oplus_3$ | Logic Code (Binary Code) | $\oplus_1$ | $\oplus_2$ | $\oplus_3$ |
|---|---|---|---|---|---|---|---|
| 0(000) | $\geq$ | $\geq$ | $\geq$ | 4(100) | $<$ | $<$ | $\geq$ |
| 1(001) | $<$ | $\geq$ | $\geq$ | 5(101) | $<$ | $\geq$ | $<$ |
| 2(010) | $\geq$ | $<$ | $\geq$ | 6(110) | $\geq$ | $<$ | $<$ |
| 3(011) | $\geq$ | $\geq$ | $<$ | 7(111) | $<$ | $<$ | $<$ |

- $\overrightarrow{\oplus} = (\oplus_1, \oplus_2, \oplus_3)$,
- the logical combination of the three predicates $Cond(Z_i)(i = 1, 2, 3)$.

To encode $\overrightarrow{\mathbf{z}}$, we first transform the range of the variable $Z$ into a fixed interval, say $[0, 31]$.

$$Z^* = \frac{Z - Z_{min}}{Z_{max} - Z_{min}} \times 32 \tag{4}$$

Then $Z^*$ will be transformed by assigning the largest integer that is not greater than $Z^*$ except for $Z^*_{max}$, which shall be assigned 31.

$$Z^{**} = \begin{cases} n, & \text{if} n \leq Z^* < n + 1, \\ 31, & \text{if} Z^* = 32. \end{cases} \tag{5}$$

Since there are only 32 cutoff values, each $z$ can be encoded by a 5-bit binary string and hence the vector $\overrightarrow{\mathbf{z}}$ can be encoded by a 15-bit binary string. To encode $\overrightarrow{\oplus}$, notice that each $\oplus$ has only two possibilities: $\geq$ or $<$. Therefore, a $\oplus$ can be encoded by a 3-bit binary string (Table 1). Finally, there are totally 8 logical combinations for three predicates and they can be encoded by 3-bit strings (Table 2).

In sum, a CONDS can be encoded by a 21-bit string (3 for logical combinations, 3 for inequalities, 15 for the three thresholds). Since each trading strategy is composed of two CONDSes, it can be represented by a 42-bit string. Let $\mathcal{D}$ be the collection of all trading strategies encoded as above. Then the cardinality of $\mathcal{D}$ is $2^{42}$ ($\#(\mathcal{D}) = 2^{42}$), which is more than 0.1 trillion. Since the vector $\overrightarrow{\mathbf{Z}}$ has six variables and each CONDS is made of three of them, there are totally 20 ($\binom{6}{3}$) combinations of 6 variables taken 3 at a time. Hence, there are twenty different $\mathcal{D}$s considered in this paper. Denote them by $\mathcal{D}_{ijk}$ $(i, j, k = 1, ..., 6, i < j < k)$. For example, $\mathcal{D}_{146}$ refers to the CONDSes made of the first ($DMVPT$), the fourth ($DMAP$) and the sixth ($DEMA$) variable in the vector $\overrightarrow{\mathbf{Z}}$.

To test the *no-free-lunch* and the *well-ordered* property defined in Chen (1998), one hundred thousand 42-bit strings (denoting one hundred thousand trading strategies) are randomly generated for each $\mathcal{D}_{ijk}$. Following the notations introduced in Chen (1998), let $r^t_s(d)$ $(d \in \mathcal{D}_{ijk})$ be the returns earned by following a trading strategy $d$ over the time interval $[s, t]$. To calculate $r^t_s$, we must be very specific about the contents of the actions **BUY**, **SELL**, and **HOLD**. In this paper, the return is calculated in terms of per-unit basis. So,

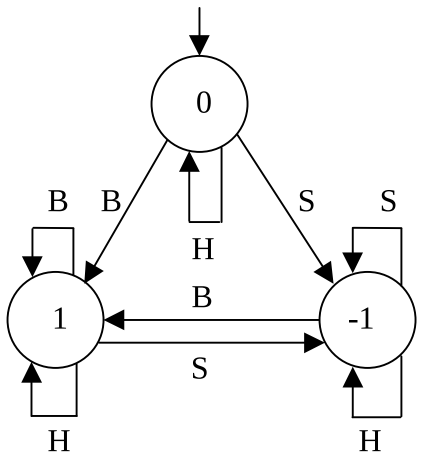

**Fig. 1.** Three-State Automaton

**Table 2.** Binary Codes for Logical Combination

| Logic Code (Binary Code) | Logical Combination of Predicates | Logic Code (Binary Code) | Logical Combination of Predicates |
|---|---|---|---|
| 0(000) | Cond 1 OR (Cond 2 AND Cond 3) | 4(100) | (Cond 1 OR Cond 3) AND Cond 2 |
| 1(001) | Cond 1 AND (Cond 2 OR Cond 3) | 5(101) | (Cond 1 AND Cond 3) OR Cond 2 |
| 2(010) | (Cond 1 OR Cond 2) AND Cond 3 | 6(110) | Cond 1 OR Cond 2 OR Cond 3 |
| 3(011) | (Cond 1 AND Cond 2) OR Cond 3 | 7(111) | Cond 1 AND Cond 2 AND Cond 3 |

**BUY** means to buy one unit of stock and **SELL** to sell one unit of stock. However, since short-selling is allowed in this paper, **BUY** could further imply *to recover short*, and **SELL** could imply *to sell short*. In the latter case, two units of stock are bought or sold at the same time. Finally, **HOLD** means no trade.

The whole rule of game may be better described by a three-state automaton depicted in Figure 1. The three states are denoted by "0", "1" and "-1". "1" refers to being *long* on one unit of stock and "-1" refers to being *short* on one unit of stock. "0" serves as both the *initial* state and the *terminal* state. In addition to these three states, there are three actions which will determine the transition to the next state, namely, "B"(**BUY**), S(**SELL**), and "H"(**HOLD**). As Figure 1 manifests, at any point in time, a trader can only be either *long* or *short* on one unit of stock. Therefore, if a trader is already on the long (short) position, then any "**BUY**" ("**SELL**") action shall be ignored. Finally, the terminal state "0" will not be reached until the clearance date, i.e., the period $t$.

Based on this three-state automaton, $r_s^t$ can be calculated as follows. First, a sequence of accumulated profits $\pi_\tau$ $(0 \leq \tau < t - s)$ is defined as follows (See also Figure 2):

$$\pi_0 = 0, \tag{6}$$

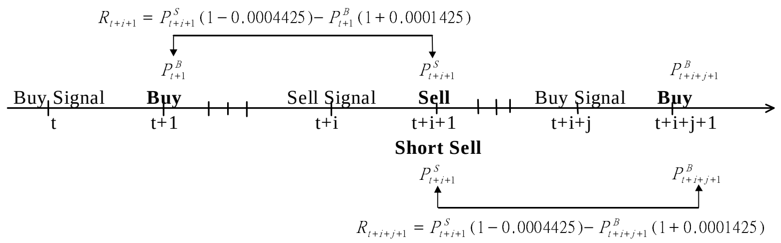

**Fig. 2.** Trades and Accumulated Returns in the Time Flow

and, for $0 < \tau \leq t - s$,

$$\pi_\tau = \begin{cases} 0, & \text{if } I_{\tau-1} = 0, I_\tau = 1, -1, \\ \pi_{\tau-1}, & \text{if } I_\tau = I_{\tau-1}, \\ \pi_{\tau-1} + P_{s+\tau}(1-c_1-c_2) - P_{s+\lambda_\tau}(1+c_1), & \text{if } I_{\tau-1} = 1, I_\tau = -1, 0 \\ \pi_{\tau-1} + P_{s+\lambda_\tau}(1-c_1-c_2) - P_{s+\tau}(1+c_1), & \text{if } I_{\tau-1} = -1, I_\tau = 1, 0 \end{cases} \tag{7}$$

where $I_\tau$ is the state at the time period $s + \tau$. As depicted in Figure 1, $I_\tau \in \{1, 0, -1\}$. $c_1$ is the tax rate of each transaction, and $c_2$ the tax rate of securities exchange income. In the case of Taiwan, $c_1 = 0.0001425$, and $c_2 = 0.0003$. $\lambda_\tau$ is an index function:

$$\lambda_\tau = \min\{\lambda \mid 0 \leq \lambda < \tau, I_\lambda \neq I_\tau\} \tag{8}$$

Given the sequence of accumulated profits $\{\pi_\tau\}_{\tau=0}^{t-s}$, $r_s^t$ is defined to be $\pi_{t-s}$:

$$r_s^t \equiv \pi_{t-s} \tag{9}$$

## 4 Testing the NFL and the Temporal Stability Property

The **NFL** property is about the convergence of the *distribution* of $r_1^t$. The test which we propose is simply to draw the *box-and-whisker plot* of $r_1^t$ over the 100,000 randomly generated strategies for each $i, j, k$ $(i, j, k = 1, ..., 6, i < j < k)$, and the results are shown in Figure 3 and Figure 4. To see whether or not there is a tendency to converge, four different values of $t$ are chosen, namely, $t = 1000, 1700, 2050$, and 2400. If the **NFL** property holds, then we may anticipate to see the distribution of $r_1^t$ shrink. However, glancing through Figure 3 and Figure 4, we find that the opposite seems to be true. In addition, there are a few interesting patterns worthy of mentioning here.

- First, the *box* part, i.e., the *interquartiles range*, does shrink in all cases. Roughly speaking, the difference of accumulated returns among the $50,000$ trading strategies closes up.
- Nevertheless, the *whisker* part, i.e., the accumulated returns of the rest of the $50,000$ trading strategies tend to spread widely.

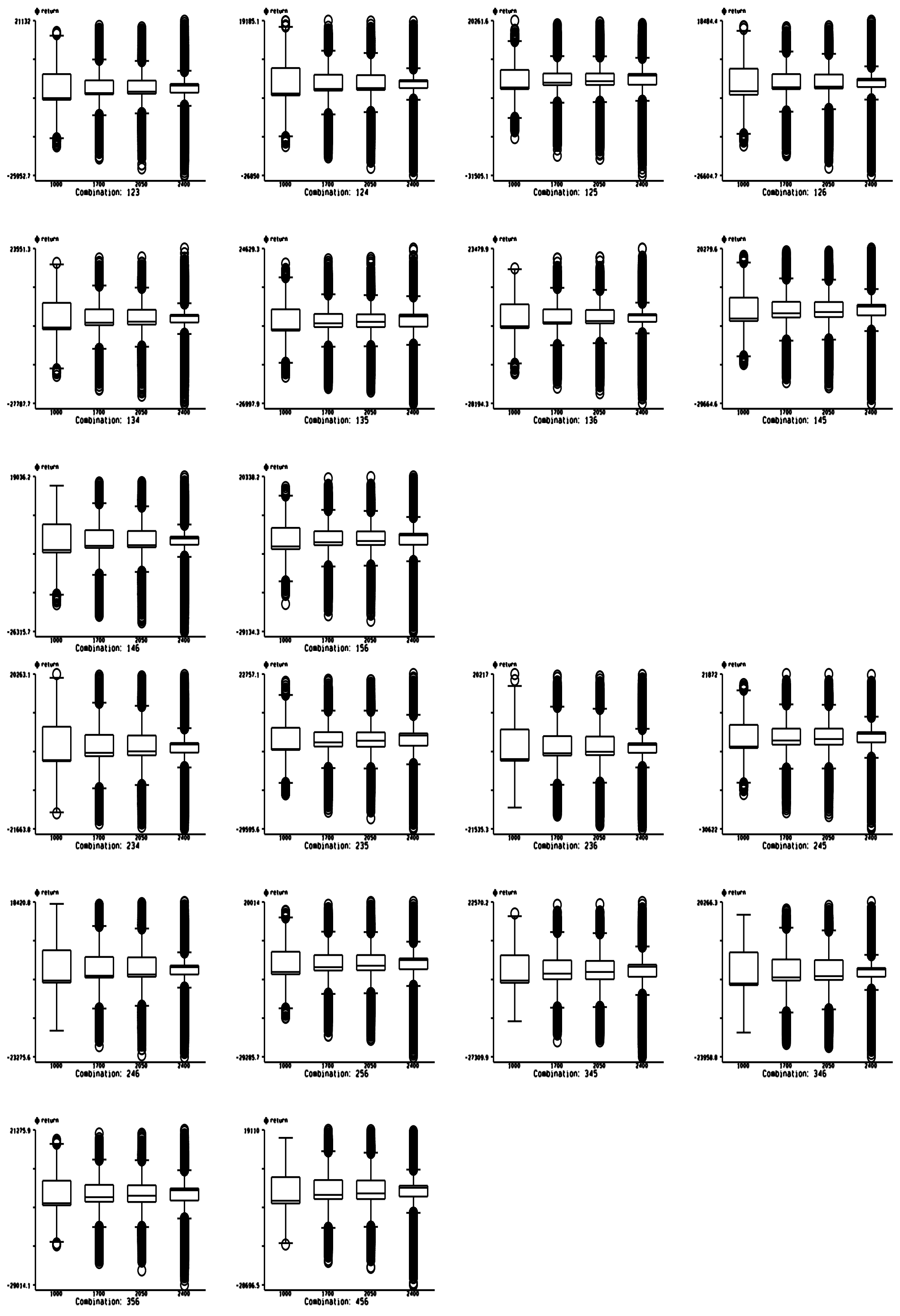

**Fig. 3.** Distribution of Accumulated Returns

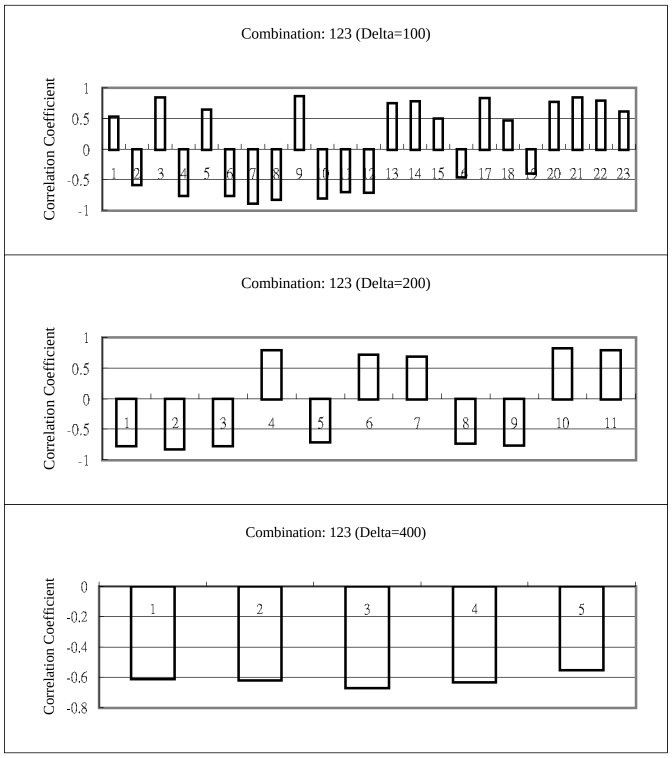

**Fig. 5.** Temporal Correlation

Thus, we have no evidence to support the **NFL** property in this limited experiment. *By Theorem 1 in Chen (1998), this result may lend support to the use of GAs in financial data mining if, statistically speaking,* $\mathcal{D}$ *is well ordered enough.* We, therefore, proceed to test the *well-ordered property*, i.e., Theorem 5 in Chen (1998). However, as simple mathematics can show, an interesting test of Theorem 5 requires a large $\Delta$; otherwise, $Corr(r_0^t, r_0^{t+\Delta})$ can be trivially close to 1 even though the $Corr(r_0^t, r_{t+1}^{t+\Delta})$ is zero. On the other hand, testing Theorem 5 also requires a large $t$. Given that our sample size is only 2400, it is quite difficult to support a large $\Delta$ and a large $t$ simultaneously. We therefore leave this test for future studies and test a related version of the well-ordered property, i.e., *temporal correlation* $(Corr(t, \Delta_1, \Delta_2))$.

We set $\Delta_1 = \Delta_2 = \Delta$, and also tried three different values of $\Delta$, i.e., $\Delta = 100, 200, 400$. The whole data set can then be divided into $k$ ($\equiv \frac{2400}{\Delta}$ non-overlapping connected intervals (In our case, $k$= 6, 12 and 24.). We computed the temporal correlation coefficient for each connected interval, i.e., we computed $Corr(t, \Delta, \Delta)$ $(t = 1, \Delta + 1, 2\Delta + 1, (k-1)\Delta + 1, k = 6, 12, 24)$. The result of $\mathcal{D}_{123}$ is depicted in Figure 5. (The other 19 cases are qualitatively similar to the one depicted.) From Figure 5, we can see that there is no general pattern for temporal correlation. For example, when $\varDelta = 400$, all correlation coefficients are *highly negative*. However, when $\varDelta = 200$ or 100, $Corr(t, \varDelta, \varDelta)$ can switch between "highly positive" and "highly negative".

While the pattern exhibited in Figure 5 is difficult to interpret, there are two points which are worth further studying. First, regardless of their signs, $Corr(t, \varDelta, \varDelta)$ are uniformly highly correlated over different values of $\varDelta$. Second, as $\varDelta$ increases, $Corr(t, \varDelta, \varDelta)$ has a tendency to be consistently negative. These statistical properties can be useful for the design of the retraining scheme. For example, using a 400-observation training sample, a 400-observation testing sample, and a 400-period retraining cycle plus choosing the *negative accumulated returns* as the fitness function may be able to generate a set of well-performed trading strategies.

## 5 Concluding Remarks

According to Chen (1998), the failure of the **NFL** property coupled with unstable temporal correlation patterns suggests two things. On one hand, GAs can be potentially useful for financial data mining. On the other hand, using GAs in a non-adaptive manner, such as Bauer (1994), may result in a disappointing performance due to the unstable landscapes. While adaptive GAs can potentially cope with dynamic landscapes and are anticipated to perform better, given the highly complex patterns of temporal correlation, the design of the *retraining scheme* is a non-trivial, if not exacting, task. In the next part, we shall see how non-trivial it is.

## References

1. Bauer, R. J. (1994), *Genetic Algorithms and Investment Strategies*, Wiley.
2. Chen, S.-H. (1998), "Can We Believe That Genetic Algorithms Would Help without Actually Seeing Them Work in Financial Data Mining?: Part I, Theoretical Foundations," *AI-ECON Research Group Working Paper Series # 9803,* National Chengchi University.

This article was processed using the LaTeX macro package with LLNCS style