Genetic Algorithms, Trading Strategies and Stochastic Processes: Some New Evidence from Monte Carlo Simulations

Shu-Heng Chen
AI-ECON Research Group
Department of Economics
National Chengchi University
Taipei, Taiwan 11623
chchen@nccu.edu.tw

Wei-Yuan Lin
AI-ECON Research Group
Department of Economics
Soochow University
Taipei, Taiwan 110
k8888@mbc1.scu.edu.tw

Chueh-Iong Tsao
AI-ECON Research Group
Department of Applied Mathematics
National Chengchi University
Taipei, Taiwan 11623
g5751002@grad.cc.nccu.edu.tw

Abstract

In this paper, the performance of canonical GA-based trading strategies are evaluated under different time series. Two classes of time series model are considered, namely, *linear ARMA model* and *bilinear model*. Unlike many existing applications of computational intelligence in financial engineering, for each performance criterion, we provide a rigorous asymptotic statistical test based on Monte Carlo simulation. As a result, this study provides us with a thorough understanding about the effectiveness of canonical GAs for evolving trading strategies under these two classes of time series.

1 Motivation

Over the past few years, genetic algorithms (GAs) have gradually become a standard tool for enhancing investment decisions.[1] Nevertheless, the *statistical foundation* of these applications has not been well established. This does not mean that we lack empirical studies to support the effectiveness of GAs in investment decisions, which, in effect, we have many. What concerns us, however, is the *robustness* of those empirical results. For example, if GAs are effective for the investment in foreign exchange markets, would the same result apply to stock markets? We fail to see how this kind of issue can be seriously addressed without building the empirical tests upon a solid statistical foundation.

In this paper, a statistical approach to evaluating GAs is proposed. Instead of testing Market A and/or Market B, we are asking what make GAs successful in Market A, but not in Market B. Statistically speaking, the general question is: what are the *statistical properties* that distinguish Market A from Market B? For example, the exchange rate data in Market A may be a *linear time series*, while the stock return data in Market B may be a *bilinear time series*, and that distinguishes these two markets. If GAs can work well with only linear time series, but not with bilinear time series, then the effectiveness of GAs observed in Market A would not apply to Market B. Therefore, our statistical approach is to evaluate GAs with different statistical properties, say, different time series, and see how well they behave in each class of time series.

In this study, two classes of time series are chosen to illustrate this statistics-based evaluation. The first class is the linear stationary time series, known as the *AutoRegressive-MovingAverage* (**ARMA**) *model* and the second one is the *bilinear model*. Both classes has been frequently employed to model financial time series. For example, linear ARMA models are found to be quite useful in *high-frequency financial data*, and bilinear models are often used to model the nonlinear dependence in both low- and high-frequency data. Of course, these two classes are not the only time series models studied in finance. Models such as the volatility-clustering process, long-memory process, and chaotic process are also among the issues frequently tackled by finance people. However, since the purpose of this paper is to evaluate GAs with our proposed statistical framework, we therefore restrict ourselves to the most basic classes of models.

The rest of the paper is organized as follows. In Section 2, we first formalize the trading strategy as a mathematical problem. We then show how, this problem can be difficult to solve analytically, and point out the use of GAs as both a *numerical* and *machine learning tool* to attack this problem. Section 3 provides a brief description of the version of canonical GAs used in this

[1] Due to the size limit, we would not include the work of other authors in this active area, the interested reader is referred to a survey article by Chen (1998).

paper. Experimental designs are detailed in Section 4, followed by the experimental results in Section 5 and some concluding remarks in Section 6.

2 The Mathematics of Trading Strategies

2.1 Trading Strategy as a Binary String

A *trading strategy* d is formally defined as a mapping:

$$d : \Omega \rightarrow \{0, 1\}. \tag{1}$$

In this paper, Ω is assumed to be a collection of *finite-length binary strings*. This simplification can be justified by a *data-preprocessing procedure*, to be exemplified below, which *transforms* the raw data into *binary strings*. The *range* of the mapping d is simplified as a 0-1 action space. In terms of simple *market-timing* strategy, "1" means to "*buy*" and "0" means to "*wait*". Here, for simplicity, we are only interested in *day trading*. So, "*buy*" means to buy it at the opening time and sell it at the closing time.

More specifically, each *trading strategy* considered in this paper has the following form:

(**IF** (CONDS)
THEN (BUY AND SELL [**DAY TRADING**])
ELSE (WAIT))

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:

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

where Z, in our application, can be considered as a time series variable indexed by t, e.g., r_{t-1}, r_{t-2}, etc, and a can be regarded as a *threshold* or *critical value* ($a \in \aleph$, a set of integers). $\oplus \in \{\geq, <\}$ and $\ominus = \{\geq, <\} - \oplus$. An example of CONDS with three primitive predicates is

$$CONDS(r_{t-1}, r_{t-2}, r_{t-3}) = Cond(r_{t-1}) \vee(\ (Cond(r_{t-2}) \wedge (Cond(r_{t-3}) \ ), \tag{3}$$

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:

Table 1: Binary Codes for Inequality Relation

Code	$\oplus_1$	$\oplus_2$	$\oplus_3$	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)	$<$	$<$	$<$

- $\vec{a} = (a_1, a_2, a_3)$,
- $\vec{\oplus} = (\oplus_1, \oplus_2, \oplus_3)$,
- the logical combination of the three predicates $Cond(r_{t-i})(i = 1, 2, 3)$.

To encode $\vec{a}$, we first transform the range of the variable Z, $[Z_{min}, Z_{max}]$, 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 a_i can be encoded by a 5-bit binary string and hence the vector $\vec{a}$ can be encoded by a 15-bit binary string. To encode $\vec{\oplus}$, notice that each $\oplus$ has only two possibilities: $\geq$ or $<$. Therefore, a $\vec{\oplus}$ 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). So, each trading strategy can be represented by a 21-bit string. Let $\mathcal{D}$ be the collection of all trading strategies encoded as above. Then the cardinality of $\mathcal{D}$ is 2^{21} ($\#(\mathcal{D}) = 2^{21}$), which is more than 2 million.

2.2 Primitive Predicate

Let r_t denote the rate of return of an financial asset at the time interval $[t-1, t]$. Suppose the stochastic process of r_t is *strictly stationary* and denote the joint density of r_{t-1} and r_t by $f(r_{t-1}, r_t)$. The mathematics of the trading strategy can then be best understood from the simplest trading strategy whose CODS has only one primitive predicate,

$$Cond(r_{t-1}) = \begin{cases} 1(True), & \text{if } \ r_{t-1} \geq a, \\ 0(False), & \text{if } \ r_{t-1} < a. \end{cases} \tag{6}$$

Table 2: Binary Codes for Logical Combination

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

Denote this strategy by d_a. Motivated by d_a, we shall decompose the range of (r_{t-1}, r_t) into four subspaces, namely,

$$A_a \equiv \{(r_{t-1}, r_t) \mid r_{t-1} \geq a, r_t \geq 0\}, \tag{7}$$

$$B_a \equiv \{(r_{t-1}, r_t) \mid r_{t-1} < a, r_t \geq 0\}, \tag{8}$$

$$C_a \equiv \{(r_{t-1}, r_t) \mid r_{t-1} < a, r_t < 0\}, \tag{9}$$

$$D_a \equiv \{(r_{t-1}, r_t) \mid r_{t-1} \geq a, r_t < 0\}. \tag{10}$$

Given the trading strategy d_a, these four areas have different meanings for the investors. For area A_a, the investor will invest ($r_{t-1} \geq a$) and will earn a positive return from that investment ($r_t \geq 0$). For area B_a, the investor will not invest ($r_{t-1} < a$) and will hence miss the chance to make a profit ($r_t \geq 0$). For area C_a, the investor will not make an investment ($r_{t-1} < a$) and will avoid the chance of suffering a loss ($r_t < 0$). For area D_a, the investor will invest, but will suffer a loss. The accumulated returns of d_a over n periods can be represented as follows.

$$\begin{aligned} \pi_n &= \prod_{t=1}^{n}(1+r_t) \\ &= \prod_{A_a}(1+r_t)\prod_{B_a}(1+r_t)\prod_{C_a}(1+r_t)\prod_{D_a}(1+r_t) \\ &= \prod_{A_a}(1+r_t)\prod_{B_a}(1+0)\prod_{C_a}(1+0)\prod_{D_a}(1+r_t) \\ &= \prod_{(r_{t-1},r_t)\in A_a}(1+r_t)\prod_{(r_{t-1},r_t)\in D_a}(1-\mid r_t \mid) \end{aligned} \tag{11}$$

By taking logarithm on both sides of Equation (5), we have

$$\ln(\pi_n) = \sum_{(r_{t-1},r_t)\in A_a}\ln(1+r_t) + \sum_{(r_t,r_{t-1})\in D_a}\ln(1-\mid r_t \mid) \tag{12}$$

Since π_n is a random variable, the objective function for the investor can be the expected profit rate, i.e., $E(\ln(\pi_n))$, where

$$\begin{aligned} E(\ln(\pi_n)) = n &\int_a^{\infty}\int_0^{\infty} f(r_{t-1}, r_t)dr_t dr_{t-1} \\ &\times \int_a^{\infty}\int_0^{\infty}\ln(1+r_t)f_{A_a}(r_{t-1}, r_t)dr_t dr_{t-1} \\ &+ n\int_a^{\infty}\int_{-\infty}^{0} f(r_{t-1}, r_t)dr_t dr_{t-1} \\ &\times \int_a^{\infty}\int_{-\infty}^{0}\ln(1+r_t)f_{D_a}(r_{t-1}, r_t)dr_t dr_{t-1} \end{aligned}$$

Given this objective function, the market timing problem can be defined as the following optimization problem.

$$\max_a E(\ln(\pi_n)) \tag{13}$$

The first order necessary condition is

$$\begin{aligned} &\frac{\partial E(\ln(\pi_n))}{\partial a} \\ &= n\int_0^{\infty}\frac{\partial \int_a^{\infty}\ln(1+r_t)f(r_{t-1}, r_t)dr_{t-1}}{\partial a}dr_t \\ &\quad + n\int_{-\infty}^{0}\frac{\partial \int_a^{\infty}\ln(1+r_t)f(r_{t-1}, r_t)dr_{t-1}}{\partial a}dr_t \\ &= -n\int_0^{\infty}\ln(1+r_t)f(a, r_t)dr_t \\ &\quad -n\int_{-\infty}^{0}\ln(1+r_t)f(a, r_t)dr_t \\ &= -n\int_{-\infty}^{\infty}\ln(1+r_t)f(a, r_t)dr_t \\ &= -nF(a) = 0 \end{aligned} \tag{14}$$

where

$$F(a) = \int_{-\infty}^{\infty}\ln(1+r_t)f(a, r_t)dr_t \tag{15}$$

Solving the first-order condition will result in the optimal value of a, i.e.,

$$a^* = F^{-1}(0), \quad if \quad F^{-1}(0) exists. \tag{16}$$

From Equation (16), to get a^*, we have to know the the inverse function of $F(a)$, which in general can only be solved numerically. In this case, GAs can be used as a *numerical technique* to solve this problem. In addition, to get a^*, we also have to know the density function of $f(r_{t-1}, r_t)$, which can only be inferred from the historical data. In this case, GAs are used as a *machine learning* tool to get an estimate of this joint density. Therefore, in the trading-strategy problem, GAs are used simultaneously as a *numerical technique* and a *machine learning* tool to get the critical parameter a^*.

2.3 Composite Predicate

When the trading strategy is a logical combination of many predicates, such as the application case in Chen and Lin (1998), the mathematics can be quite complicated. In general, let us consider a decision composed of a logical combination of k primitive predicates, of which each predicate assigns a threshold value a_k to the historical return r_{t-k}. Let α be the vector $(a_1, a_2, ..., a_k)$, and r_t^k be the vector $(r_{t-1}, r_{t-2}, ..., r_{t-k})$. Moreover, let D be the region where the composite predicate is satisfied, and D' be the region where the composite predicate is not satisfied. Then the four regions as defined in Equations (7) to (10) can be generalized as follows:

$$A_\alpha \equiv \{(r_t, r_t^k) \mid r_t^k \in D, r_t \geq 0\} \tag{17}$$

$$B_\alpha \equiv \{(r_t, r_t^k) \mid r_t^k \in D', r_t \geq 0\} \tag{18}$$

$$C_\alpha \equiv \{(r_t, r_t^k) \mid r_t^k \in D', r_t < 0\} \tag{19}$$

$$D_\alpha \equiv \{(r_t, r_t^k) \mid r_t^k \in D, r_t < 0\} \tag{20}$$

The optimization problem of market timing can then be generalized as

$$\max_\alpha E(\ln(\pi_n)), \tag{21}$$

where

$$\begin{aligned} &E(\ln(\pi_n)) \\ &\quad = n \int_{A_\alpha} ... \int \int_0^\infty f(r_t^k, r_t) dr_t dr_{t-1} ... dr_{t-k} \\ &\times \int_{A_\alpha} ... \int \int_0^\infty \ln(1+r_t) f_{A_\alpha}(r_t^k, r_t) dr_t dr_{t-1} ... dr_{t-k} \\ &+ \ n \int_{D_\alpha} ... \int \int_{-\infty}^0 f(r_t^k, r_t) dr_t dr_{t-1} ... dr_{t-k} \\ &\times \int_{D_\alpha} ... \int \int_{-\infty}^0 \ln(1+r_t) f_{D_\alpha}(r_t^k, r_t) dr_t dr_{t-1} ... dr_{t-k} \end{aligned}$$

As in the case of the primitive predicate, GAs are used as a numerical technique as well as a machine learning algorithm to solve this more complicated objective function.

3 Evolving Trading Strategies

The main idea of using GAs to evolve trading strategies is to encode the variable one wants to optimize, e.g., the *trading strategy*, as a *binary string* and work with it. In the previous section, we have already shown how a trading strategy can be parameterized, and how this parameterized trading strategy can be encoded by a binary string. In the illustration above, each trading strategy is a 21-bit string. The size of their search space $\mathcal{D}$ is hence 2^{21}. Given this structure, genetic algorithms can be briefly described as follows.

- The genetic algorithm maintains *a population of individuals*
  $$P_i = \{d_1^i, ..., d_n^i\} \tag{22}$$
  for iteration i, where n is the *population size*. Usually, n is treated as fixed during the whole evolution. Clearly, $P_i \subset \mathcal{D}$.
- Each individual d_j^i represents a trading strategy at hand, and is implemented with the *historical data* r_{t-1}, r_{t-2}, and r_{t-3} (Equation 3).
- Each trading strategy d_j^i is evaluated by the *fitness* given by Equation (11).
- (**Selection Step**):
  Then, a new generation of population (iteration $i+1$) is formed by randomly selecting individuals from P_i in accordance with a *selection scheme*.
  $$\mathcal{P}_s(P_i) = (s_1(P_i), s_2(P_i), ..., s_n(P_i)) \tag{23}$$
  where
  $$s_k : \{\binom{\mathcal{D}}{n} \to \mathcal{D}\}, \tag{24}$$
  $k = 1, 2, ..., n$, and $\binom{\mathcal{D}}{n}$ is the set of all populations whose population size is n.
- (**Alteration Step**):
  Some members of the new population undergo transformations by means of *genetic operators* to form new solutions.
  - (**Crossover:**) We use *one-point crossover* c_k, which creates new individuals by combining parts from two individuals.
    $$\mathcal{P}_c(P_i) = (c_1(P_i), c_2(P_i), ..., c_{\frac{n}{2}}(P_i)) \tag{25}$$
    where
    $$c_k : \mathcal{D} \times \mathcal{D} \to \mathcal{D} \times \mathcal{D}, \tag{26}$$
    $k = 1, 2, ..., \frac{n}{2}$.
  - (**Mutation:**) We use *bit-by-bit mutation* m_k, which creates new individuals by a small change in a single individual.
    $$\mathcal{P}_m(P_i) = (m_1(P_i), m_2(P_i), ..., m_n(P_i)) \tag{27}$$
    where
    $$m_k : \mathcal{D} \to \mathcal{D}, \tag{28}$$
    $k = 1, 2, ..., n$.

The GA described above is a very simple version of GAs, which we shall call it the ordinary genetic algorithm (**OGA**). The control parameters employed to run the OGA is given in Table 3.

Table 3: Table of OGA

Number of Generation	1000
Population Size (n)	50
Total Trails	5000
String Length	21
Selection Scheme	Rank-Based Selection
Rank Min	0.75
Crossover Style	One-Point
Crossover Rate	0.6
Mutation Rate	0.001
Generation Gap	1.0

Table 4: Data Generating Processes: ARMA

Code	Model	Parameters
L-1	ARMA(1,0)	ρ_1 = 0.3
L-2	ARMA(1,0)	ρ_1 = 0.6
L-3	ARMA(2,0)	ρ_1 = 0.3, ρ_2 = -0.6
L-4	ARMA(2,0)	ρ_1 = 0.6, ρ_2 = -0.3
L-5	ARMA(0,1)	θ_1 = 0.3
L-6	ARMA(0,1)	θ_1 = 0.6
L-7	ARMA(0,2)	θ_1 = 0.3, θ_2 = -0.6
L-8	ARMA(0,2)	θ_2 = 0.6, θ_2 = -0.3
L-9	ARMA(1,1)	ρ_1 = 0.3, θ_1 = -0.6
L-10	ARMA(1,1)	ρ_1 = 0.6, θ_1 = -0.3
L-11	ARMA(2,2)	ρ_1 = 0.4, ρ_2 = -0.4 θ_1 = 0.4, θ_2 = 0.4
L-12	ARMA(2,2)	ρ_1 = 0.6, ρ_2 = -0.3 θ_1 =-0.3, θ_2 = -0.6
L-13	White Noise	$Gaussian(0, 0.1)$

4 Experimental Design

4.1 Linear Time Series Models

The first class of models we consider is the *linear time series model*, known as the *AutoRegressive-MovingAverage* (**ARMA**) model. The general form of the $ARMA(p, q)$ model is:

$$r_t = \sum_{i=1}^{p} \rho_i r_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t, \qquad (29)$$

where $\epsilon_t \overset{iid}{\sim} N(\mu, \sigma^2)$. In all the simulations conducted, μ was set at 0 and σ^2 was set to be 0.01. Thirteen $ARMA(p, q)$ models were tested. The parameters of these thirteen $ARMA(p, q)$ are detailed in Table 4.

4.2 Bilinear Models

The second class of models considered is the *bilinear model*. The general form of the bilinear model, $BL(p, q, u, v)$ is:

$$r_t = \sum_{i=1}^{p} \rho_i r_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \sum_{l=1}^{u} \sum_{m=1}^{v} \psi_{lm} r_{t-l} \epsilon_{t-m} + \epsilon_t, \qquad (30)$$

where $\epsilon_t \overset{iid}{\sim} N(\mu, \sigma^2)$, and $\mu = 0$ and $\sigma^2 = 0.01$. Eight bilinear models were tested, and their parameters are given in Table 5.

Table 5: Data Generating Processes: Bilinear

Code	Model	Parameters					
		ρ_1	θ_1	ψ_{11}	ψ_{12}	ψ_{21}	ψ_{22}
BL-1	BL(0,0,1,1)	0	0	0.6	0	0	0
BL-2	BL(0,0,1,1)	0	0	0.3	0	0	0
BL-3	BL(0,1,1,2)	0	0.3	0	0.6	0	0
BL-4	BL(0,1,1,2)	0	0.6	0	0.3	0	0
BL-5	BL(1,0,2,1)	0.3	0	0	0	0.6	0
BL-6	BL(1,0,2,1)	0.6	0	0	0	0.3	0
BL-7	BL(1,1,2,2)	0.3	0.3	0	0	0	0.3
BL-8	BL(1,1,2,2)	0.3	0.3	0	0	0	0.6

4.3 Monte Carlo Simulation

Each $ARMA$ and bilinear model was used to generate 1000 time series, each with 105 observations ($\{r_t\}_{t=1}^{105}$). For each series, the first 70 observations ($\{r_t\}_{t=1}^{70}$) were taken as the training sample, and the last 35 observations ($\{r_t\}_{t=76}^{105}$) were used as the testing sample. The OGA was then employed to extract trading strategies from these training samples. These strategies were further tested by the testing samples, and the resulting *accumulated returns* were calculated. In the mean time, accumulated returns of the buy-and-hold strategy were also calculated as a benchmark. Let $\pi_{i,j}^1$ be the accumulated returns of the buy-and-hold (B&H) strategy for the series i ($i = 1, 2, ..., 1000$) under Model L-j ($j = 1, 2, ..., 13$) or BL-j ($j = 1, 2, ...6$), and $\pi_{i,j}^2$ be the accumulated returns of using the OGA in series i under Model L-j or BL-j.

The issue which we shall address is, given the set of observations $S_j (\equiv \{\pi_{i,j}^1, \pi_{i,j}^2\}_{i=1}^{1000})$, j=1,2,...,13(6) , to decide *whether the ordinary genetic algorithm can statistically significantly outperform the buy-and-hold strategy under a fixed Model L-j or BL-j*.

4.4 Performance Criteria and Test Statistics

Our evaluation is based on the following statistics:

- the sample winning probability, i.e.,

$$\hat{p}_w = \frac{\#\{(\pi_{i,j}^1, \pi_{i,j}^2) \mid \pi_{i,j}^2 > \pi_{i,j}^1\}}{1000} \qquad (31)$$

  where # refers to the *cardinality* (size) of the set.

- the sample mean, i.e.,

$$\bar{\pi}_j^1 = \frac{\sum_{i=1}^{1000} \pi_{i,j}^1}{1000} \quad (32)$$

$$\bar{\pi}_j^2 = \frac{\sum_{i=1}^{1000} \pi_{i,j}^2}{1000} \quad (33)$$

- z_s, the test statistic of the null hypothesis

$$H_0 : \hat{p}_w = 0.5,$$

- z_π, the test statistic of the null hypothesis

$$H_0 : \pi_j^2 = \pi_j^1.$$

- the *ideal* sample mean,

$$\bar{\pi}_j^* = \frac{\sum_{i=1}^{1000} \pi_{i,j}^*}{1000} \quad (34)$$

- the *Sharpe-ratio*,

$$s_j^1 = \frac{\bar{\pi}_j^1}{\hat{\sigma}_j^1}, \quad s_j^2 = \frac{\bar{\pi}_j^2}{\hat{\sigma}_j^2}, \quad (35)$$

where

$$\hat{\sigma}_j^k = \sqrt{\frac{\sum_{i=1}^{1000}(\pi_{i,j}^k - \bar{\pi}_j^k)^2}{1000}}, \quad k = 1, 2 \quad (36)$$

- z_r, the test statistic of the null hypothesis

$$H_0 : s_j^2 - s_j^1 = 0$$

The sample winning probability, $\hat{p}_w$, estimates the *probability* that the OGA can outperform B&H. In other words, it tells us, by randomly picking up an ensemble from the environment $L - j$ ($BL - j$), the probability that the OGA can beat B&H. The sample mean $\bar{\pi}_j^1$ and $\bar{\pi}_j^2$ estimates, on the average, the numerical difference between the OGA and B&H in terms of accumulated returns. Of course, a rigorous analysis requires a statistical test for both sample statistics. Since our random sample size is 1,000, based on the *central limit theorem*, we can have a standard normal (Z) test for both statistics. z_w tells us whether the OGA is superior to B&H significantly, while z_π tells us whether this superiority deserves our investment in the OGA.

Although these four statistics can give a rigorous evaluation of the relative performance of the OGA to B&H, they are not sufficient to answer how well the OGA can solve the optimization problem defined in Equation (14). In fact, to answer this question, we have to know the highest *post*-accumulated returns one may possibly have, i.e., the accumulated returns which can be earned by an *omniscient* trader. Denoting the accumulated returns earned by this omniscient trader, $\pi_{i,j}^*$, we therefore calculate the sample mean of these *ideal* accumulated returns, π_j^*.

Table 6: Performance Statistics of the OGA and B&H in ARMA Models:

Code	$\bar{\pi}^1$	$\bar{\pi}^2$	$\dot{\pi}$	$\bar{\pi}^*$	$\tilde{\pi}$	$\hat{p}_w$	z_w	z_π
L-1	1.198	1.355	13%	4.388	30%	0.732	16.56	6.33
L-2	1.992	2.868	43%	6.658	43%	0.859	32.62	13.67
L-3	0.845	2.265	167%	5.480	41%	0.976	98.35	42.98
L-4	1.123	1.185	65%	5.170	35%	0.896	41.02	27.08
L-5	1.103	1.269	15%	4.241	29%	0.713	14.89	7.63
L-6	1.199	1.775	48%	5.166	34%	0.861	32.99	20.61
L-7	0.853	1.633	91%	5.104	32%	0.926	51.46	39.97
L-8	1.065	1.522	42%	5.285	28%	0.848	30.65	21.58
L-9	0.898	1.229	36%	4.128	29%	0.812	25.25	24.55
L-10	1.452	1.538	5%	4.783	32%	0.721	15.58	2.12
L-11	1.306	2.588	98%	6.957	37%	0.927	51.90	30.43
L-12	0.721	2.167	200%	6.189	35%	0.991	164.40	47.39
L-13	1.005	0.985	-2%	4.257	23%	0.579	5.05	-1.26
L-13	0.983	0.993	1%	3.881	25%	0.606	6.85	0.67

One criterion which has been frequently ignored by machine learning people in finance is the *risk* associated with a trading rules. Normally, a higher profit known as the *risk premium* is expected when the associated risk is higher. Without taking the risk into account, we might exaggerate the profit performance of a highly risky trading rule. Therefore, to evaluate the performance of our GA-based trading rule on a risk-adjusted basis, we employed the well-known *Sharpe ratio* as another performance criterion (Sharpe, 1966). Sharpe ratio s is defined as the excess return divided by a risk measure. The higher the Sharpe ratio, the higher the return or the lower the risk. Here, we used the sample return as the excess return and the sample standard deviation as the risk measure, and Equation (35) gives the Sharpe ratio of the B&H and OGA.

To see whether these two Sharpe ratios are statistically different, we need a formal test for the *Sharpe-ratio differential*, $d(= s_j^2 - s_j^1)$. However, to our best knowledge, this test does not exist in the literature. We, therefore, propose such a test by using the *Slutzky's theorem*, and this test is given in Theorem 1.

Theroem 1: Let $(X_i, Y_i) \overset{iid}{\sim} h(x, y)$, $i = 1, 2, ..., n$ with

$$\begin{bmatrix} E(X_i) \\ Var(X_i) \\ \frac{E(X_i-\mu)^3}{\sigma^3} \\ \frac{E(X_i-\mu)^4}{\sigma^4} \end{bmatrix} = \begin{bmatrix} \mu \\ \sigma^2 \\ \delta \\ \gamma \end{bmatrix}, \begin{bmatrix} E(Yi) \\ Var(Y_i) \\ \frac{E(Y_i-\nu)^3}{\tau^3} \\ \frac{E(Y_i-\nu)^4}{\tau^4} \end{bmatrix} = \begin{bmatrix} \nu \\ \tau^2 \\ \xi \\ \eta \end{bmatrix}$$

$$\begin{bmatrix} \frac{E(X_i-\mu)(Y_i-\nu)}{\sigma\tau} \\ \frac{E(X_i-\mu)^2(Y_i-\nu)}{\sigma^2\tau} \\ \frac{E(X_i-\mu)(Y_i-\nu)^2}{\sigma\tau^2} \\ \frac{E(X_i-\mu)^2(Y_i-\nu)^2}{\sigma^2\tau^2} \end{bmatrix} = \begin{bmatrix} \rho \\ \theta \\ \psi \\ \phi \end{bmatrix}$$

Furthermore, let $\bar{X}_n$ and $\bar{Y}_n$ be the sample means of X and Y and let S_n^2 and T_n^2 be the sample variances of X and Y with the sample size n. Then

1.
$$d_n = \frac{\bar{X}_n}{S_n} - \frac{\bar{Y}_n}{T_n} \tag{37}$$
is a consistent estimator of $d (= \frac{\mu}{\sigma} - \frac{\nu}{\tau})$.

2.
$$Z_r = \frac{\sqrt{n}(d_n - d)}{\omega} \xrightarrow{d} Z \sim N(0,1) \tag{38}$$
where
$$\omega^2 = 2(1-\rho) + \frac{\mu}{\sigma}(\theta - \delta) + \frac{\nu}{\tau}(\psi - \xi) - \frac{\mu\nu}{\sigma\tau}(\frac{\phi - 1}{2}) + \frac{\mu^2}{\sigma^2}(\frac{\gamma - 1}{4}) + \frac{\nu^2}{\tau^2}(\frac{\eta - 1}{4}) \tag{39}$$

3.
$$Z_{r,n} = \frac{\sqrt{n}(d_n - d)}{\omega_n} \xrightarrow{d} Z \sim N(0,1) \tag{40}$$
where
$$\omega_n^2 = 2(1-\rho_n) + \frac{\mu_n}{\sigma_n}(\theta_n - \delta_n) + \frac{\nu_n}{\tau_n}(\psi_n - \xi_n) - \frac{\mu_n\nu_n}{\sigma_n\tau_n}(\frac{\phi_n - 1}{2}) + \frac{\mu_n^2}{\sigma_n^2}(\frac{\gamma_n - 1}{4}) + \frac{\nu_n^2}{\tau_n^2}(\frac{\eta_n - 1}{4}) \tag{41}$$
and
$$\begin{bmatrix} \mu_n & \nu_n & \rho_n \\ \sigma_n^2 & \tau_n^2 & \theta_n \\ \delta_n & \xi_n & \psi_n \\ \gamma_n & \eta_n & \phi_n \end{bmatrix} \xrightarrow{p} \begin{bmatrix} \mu & \nu & \rho \\ \sigma^2 & \tau^2 & \theta \\ \delta & \xi & \psi \\ \gamma & \eta & \phi \end{bmatrix}$$

The test $z_{r,n}$ is then computed in accordance with Equations (40) and (41).

5 Experimental Results

Table 6 summarizes the five statistics defined in the previous section. There are several interesting features. First, from the statistics $\hat{p}_w$ and z_w, it can be inferred that, in accumulated returns, the probability that the OGA can beat B&H *is significantly greater than 0.5*. For models with linear signals (L-1 - L-12), the winning probability $\hat{p}_w$ ranges from 0.713 (L-5) to 0.991 (L-12). What seems a little puzzling is that, even in the case of *white noises*, the OGA outperformed B&H, though with much lower winning probabilities p_s (0.579 and 0.606). This may be due to the fact that a pseudo random generator can actually generate a series with signals *when sample size is small.* For example, Chen and Tan (1998) showed that, when the sample size is 50, the probability of having signals in a series generated from a pseudo random generator is about 5%, while that probability can go to zero when the sample size is 1000. Therefore, suppose that the OGA can win in all these exceptional ensembles and tie with B&H in other normal ensembles, then $\hat{p}_w$ can still be significantly greater than 0.5.

Table 7: Performance Statistics of GAs and B&H in Bilinear Models:

Code	$\bar{\pi}^1$	$\bar{\pi}^2$	$\dot{\pi}$	$\bar{\pi}^*$	$\tilde{\pi}$	$\hat{p}_w$	z_w	z_π
BL-1	1.253	1.126	-10%	4.398	25%	0.491	-0.57	-6.78
BL-2	1.151	1.064	-7%	4.228	25%	0.517	1.08	-4.66
BL-3	1.302	1.830	41%	5.341	34%	0.861	17.78	11.50
BL-4	1.186	1.356	14%	4.449	31%	0.745	17.78	6.95
BL-5	1.260	1.419	13%	4.539	31%	0.747	17.97	5.07
BL-6	2.292	3.143	37%	7.226	44%	0.877	36.30	9.89
BL-7	1.841	2.471	34%	6.448	38%	0.848	30.65	8.83
BL-8	1.602	2.287	43%	5.894	39%	0.870	34.79	19.57

Second, by directly comparing $\bar{\pi}^1$ with $\bar{\pi}^2$, we can see that the OGA outperformed B&H *numerically* in all linear $ARMA(p,q)$ models except in one case of white noises. To have a more practical comparison, we also calculate the excess return of the OGA over B&H, $\dot{\pi}$, as follows:
$$\dot{\pi} \equiv \frac{\bar{\pi}^2 - \bar{\pi}^1}{\bar{\pi}^1}, \tag{42}$$
and the results are exhibited in the fifth column of Table 6. From this table, we can see that GAs beat B&H anywhere from 5% ($L-10$) to more than 200% ($L-12$). From z_π, numerical differences are also statistically significant. The statistics z_π range from 2.12 to 47.39, which implies substantially different profits. However, for the two cases of white noises, the OGA are not statistically different from B&H.

Third, to see how effectively the OGA can solve the optimization problem defined in Equation (14), we divide realized accumulated returns by potential accumulated returns,
$$\tilde{\pi} \equiv \frac{\bar{\pi}^2}{\bar{\pi}^*}. \tag{43}$$
By definition, $\tilde{\pi}$ is between 0 and 1. A higher $\tilde{\pi}$ implies a more effective solution obtained from the OGA. From Table 6, we can see that even our simplest version of GAs can realize 30% to 40% of potential returns.

The bilinear model is a little more complicated than the ARMA model. It is used to capture more complex phenomena, such as bursts and asymmetry. As Table 7 reveals, this class of nonlinear time series is indeed more difficult for our OGA. In fact, out of the eight testing beds, the OGA lost twice (BL-1 and BL-2) to B&H. While in these two cases the losing probability $\hat{p}_w$ is not significantly different from 0.5, their accumulated returns are significantly lower. For the other six testing beds, the OGA continuously took the lead. Nevertheless, the excess return, $\dot{\pi}$, has a narrower and lower range as opposed to ARMA models. Finally, in terms of the effectiveness index, $\tilde{\pi}$, the OGA did not perform much differently from the ARMA models. In both cases, $\tilde{\pi}$ ranges roughly from 25% to 40%.

However, as mentioned earlier, we should not judge the performance of the OGA solely by the profitability criterion. The risk is a main concern in business practice. We, therefore, also calculated the Sharpe ratio, a risk-adjusted profitability criterion, and put the results on Table 8. It is interesting to notice that in almost all cases the Sharpe-ratio differential (d) is positive. In other words, the OGA can easily outperform B&H when the risk is taken into account. Furthermore, the test of this differential also showed that this advantage of the OGA is significant.

Table 8: The Sharpe Ratio of GAs and B&H in ARMA and Bilinear Models:

Code	Model	s^1	s^2	d_n	$z_{r,n}$
L-1	ARMA(1,0)	1.002	1.668	0.666	9.37
L-2	ARMA(1,0)	0.473	0.807	0.334	8.86
L-3	ARMA(2,0)	1.877	2.113	0.236	3.19
L-4	ARMA(2,0)	1.006	1.899	0.893	14.11
L-5	ARMA(0,1)	1.169	1.975	0.806	10.71
L-6	ARMA(0,1)	0.815	1.379	0.564	4.36
L-7	ARMA(0,2)	2.057	2.445	0.388	4.79
L-8	ARMA(0,2)	1.054	1.818	0.764	12.82
L-9	ARMA(1,1)	2.727	2.806	0.080	0.75
L-10	ARMA(1,1)	0.689	1.120	0.431	4.33
L-11	ARMA(2,2)	0.723	1.392	0.668	10.67
L-12	ARMA(2,2)	3.530	2.273	-1.257	-11.83
L-13	ARMA(0,0)	1.562	2.682	1.120	14.39
L-13	ARMA(0,0)	1.505	2.529	1.024	11.57
BL-1	BL(0,0,1,1)	1.568	2.248	0.679	8.02
BL-2	BL(0,0,1,1)	1.450	2.405	0.955	10.83
BL-3	BL(0,1,1,2)	1.062	1.620	0.559	4.78
BL-4	BL(0,1,1,2)	0.699	1.596	0.897	9.03
BL-5	BL(1,0,2,1)	0.863	1.575	0.713	10.82
BL-6	BL(1,0,2,1)	0.445	0.791	0.346	8.06
BL-7	BL(1,1,2,2)	0.625	1.194	0.570	5.61
BL-8	BL(1,1,2,2)	0.477	0.686	0.209	2.21

6 Conclusions

By using the Monte Carlo simulation, this study tested the effectiveness of the OGA in evolving trading strategies under different stochastic processes. The performance criteria employed are the winning probability, accumulated returns and Sharpe ratio. It is found that the OGA uniformly outperformed B&H in all the series when only linear signals were present. When there was no signal (white noise), the OGA did not make much difference. When there were *only* nonlinear signals (BL-1 and BL-2), the OGA even performed worse than B&H. For other mixed processes which contained both linear and nonlinear signals, the OGA still beat B&H because it could effectively extract at least some linear signals. We then compared GAs with an omniscient trader, and we founded that GAs could realize 25% to 40% of potential returns. Finally, it is found that the OGA outperformed B&H in almost all cases when the evaluation is based on the Sharpe ratio, i.e., the risk-adjusted return. This study only tested the simplest version of GAs. The results obtained here may be best regarded as a *lower bound* of GAs' capability in evolving trading strategies. It is expected that a more sophisticated design can efficiently extract nonlinear signals and thus improve the effectiveness index $\tilde{\pi}$.

Acknowledgement

Research support from NSC grant No.85-2415-H-004-001 is gratefully acknowledged. The authors are also grateful for the helpful comments from two anonymous referees.

References

Chen, S.-H. (1998), "Evolutionary Computation in Financial Engineering: A Road Map of GAs and GP," *Financial Engineering News,* Vol. 2, No. 4. Also available from the website:
http://www.fenews.com/1998/v2n4/chen.pdf

Chen, S.-H. and W.-Y. Lin (1998), "Two Ways to Improve Genetic Algorithms in Financial Data Mining: Sell Short with Recursive GAs," in *Proceedings of the Seventh International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Vol. II, pp. 1090-1097.

Chen, S.-H. and C.-W. Tan (1998), "Some Evidences of the Brief Signals in Financial Times Series: An Examination based on Predictive Stochastic Complexity," *AI-ECON Research Group Working Paper*, National Chengchi University.

Sharpe, W. F. (1966), "Mutual Fund Performance," *Journal of Business*, Vol. 39, No. 1, pp.119-138.