Toward an Effective Implementation of Genetic Algorithms in Financial Data Mining: Retraining plus Validating *

Shu-Heng Chen[1], Chien-Fu Chen[2] and Ching-Wei Tan[3]

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

Abstract. Given the evidence of dynamic landscapes shown in Chen and Chen (1998), this paper evaluates the contribution of the *retraining scheme* and the *validation scheme* to financial data mining. While experimental results show that these two schemes can be helpful, knowledge about the effective implementation of these schemes, the setting of parameter values in particular, remains to be discovered.

1 Motivation and Introduction

Chen (1998a) highlighted several key underlying features which may contribute to successful financial data mining. Two of these features were actually tested in Chen and Chen (1998). They found that, while the **NFL** property fails to hold (good news for financial data mining), the landscape is dynamic in a highly complex manner (bad news, unfortunately!). The implications of these mixed results are two-fold. First, GAs can be potentially helpful. Second, due to the complex dynamics of landscapes, the design of an effective GA is far from trivial.

In the literature, there are at least two approaches to coping with dynamic landscapes. The first one is to make GAs *adaptive*. The second is to prevent GAs from *overfitting* the data or to enhance the *generalization capability* of GAs. We consider both approaches correct directions to follow when the landscape is *time-variant*. In particular, when the pattern of the landscape dynamics are complex or *stochastic*, then the second approach, well accepted in the area of machine learning, may be more promising. Techniques such as the *early stopping validation method* (Nelson and Illingworth, 1991) and *v-fold cross validation* (Hjorth, 1994) are frequently used to tackle the issue of overfitting.

In this paper, we run a competition among three different styles of GAs, namely,

- the ordinary genetic algorithm (**OGA**),
- the recursive genetic algorithm (**RGA**),
- the validated recursive genetic algorithm (**VGA**).

* Research support from MasterLink Securities Corporation is gratefully acknowledged.

The purpose of this competition is two-fold. First, from the **OGA** to **RGA**, we would like to see whether there is any advantage of using *the retraining scheme* to cope with the dynamic landscape. The word "Recursive" used here has exactly the same meaning as the "recursive" used in the *recursive estimators*. Second, from the **OGA** to **VGA**, we would like to see the significance of hybridizing a *retraining scheme* with a *validation scheme*. In sum, given the statistical features found in Chen and Chen (1998), this paper attempts to give an empirical test for the effectiveness of using the *retraining scheme* and the *validation scheme* to cope with the dynamic landscape.

2 The Ordinary Genetic Algorithm (OGA)

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 to work with this string. In Chen and Chen (1998), we have already shown how a trading strategy can be parameterized, and how this parameterized trading strategy can be encoded by a binary string. Specifically, each trading strategy in Chen and Chen (1998) is a 42-bit string. The size of their population space $\mathcal{D}$ is hence 2^{42}. Given this structure, genetic algorithms can be briefly described as follows.

- The genetic algorithm maintains *a population of individuals*,

$$P_i = \{d_1^i, ..., d_n^i\} \tag{1}$$

  for iteration i, where n is *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 *data structure* **Z** described in Chen and Chen (1998), Equation 1.
- Each trading strategy d_j^i is evaluated by the *fitness* given by Equations (4)-(7) in Chen and Chen (1998).
- (**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*.

$$\mathcal{P}_s(P_i) = (s_1(P_i), s_2(P_i), ..., s_n(P_i)) \tag{2}$$

  where

$$s_k : \{\binom{\mathcal{D}}{n} \rightarrow \mathcal{D}\}, \tag{3}$$

  $k = 1, 2, ..., n$, 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.

Table 1. Tableau of OGA

Number of Generation	1000
Population Size	100
Selection Scheme	Rank-Based Selection
Rank Minimum	0.75
Crossover Style	One-Point
Crossover Rate	0.6
Mutation Rate	0.001

- (**Crossover:**) We use *one-point crossover* c_k, which creates new individuals by combining parts from two individuals.

$$\mathcal{P}_c(P_i) = (c_1(P_i), c_2(P_i), ..., c_{\frac{n}{2}}(P_i)) \tag{4}$$

  where

$$c_k : \mathcal{D} \times \mathcal{D} \rightarrow \mathcal{D} \times \mathcal{D}, \tag{5}$$

  $k = 1, 2, ..., \frac{n}{2}$.
- (**Mutation:**) We use *bit-by-bit mutation* m_k, which creates new individuals by a small change in a single individual.

$$\mathcal{P}_m(P_i) = (m_1(P_i), m_2(P_i), ..., m_n(P_i)) \tag{6}$$

  where

$$m_k : \mathcal{D} \rightarrow \mathcal{D}, \tag{7}$$

  $k = 1, 2, ..., n$.

– After some number of generations, say T, the algorithm terminates–it is hoped that a representative from the last population can represent a near-optimum solution. The representative trading strategy considered in this paper is the one whose fitness value is the median of the sample, i.e.,

$$d^T_{med} = arg\{median[Fitness(P_T)]\} \tag{8}$$

The detailed description of each genetic operators $\mathcal{P}_s$, $\mathcal{P}_c$, $\mathcal{P}_m$ can be found in Bauer (1994). The control parameters employed to run the OGA is given in Table 1.

3 Retraining Scheme: The RGA

The recursive genetic algorithm is an *adaptive version of the OGA*, i.e., we endow the OGA with a *retraining scheme*. The retraining scheme considered here is motivated by Chen and Lin (1997). Chen and Lin (1997) studied two versions of RGAs, one with *unlimited* memory size and the other with *fixed* memory size. While each version has its theoretical support, we only consider the one with *fixed* memory size here. This version has two characteristics.

– Firstly, it has a *periodical retraining schedule*, and the period of a retraining cycle is *predetermined*. In other words, it will routinely restart OGA learning after a fixed number of periods, say n_2.
– Secondly, upon retraining, the data set is *refreshed* first. It will include all the new observations which are not available in the last training. But, since the memory size is fixed, this inclusion is inevitably accompanied by the exclusion of an equal number of the most outdated observations.

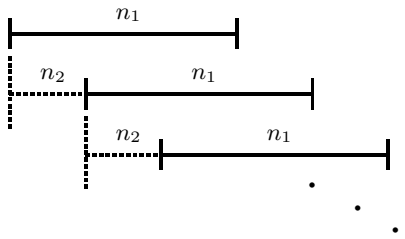

Fig. 1 The Data Structure of RGA

Figure 1 depicts the data structure used in the RGA. Here, n_1 refers to the memory size and n_2 the period of a retraining cycle. For convenience, we will denote the RGA with parameters n_1 and n_2 by RGA(n_1, n_2).

4 Generalization-Enhancing Scheme: The VGA

In the RGA, the termination criterion for a run is determined by the parameter "*Number of Generation*" (Table 1), and in this paper it is arbitrarily set to be 1000. It is not clear whether or not the choice of this number is *optimal*. To avoid *overlearning* (*poor generalization*) or *underlearning*, the *early stopping validation method* is adopted here. The procedure of this method is described below.

1. We first divide available data into the training and validation sets.
2. We then use the training set to train and run the OGA initially for m_1 generations.
3. Set $N_{GEN} = m_1$.
4. Use the validation set to calculate the fitness of $P_{N_{GEN}}$. Denote it by $V - Fitness(P_{N_{GEN}})$.
5. Run another m_2 generations.
6. Set $N_{GEN} = N_{GEN} + m_2$.
7. Use the validation set to calculate the fitness of $P_{N_{GEN}}$.
8. Compute the difference: $Diff = Median[V-Fitness(P_{N_{GEN}})]-Median[V-Fitness(P_{N_{GEN}-m_2})]$

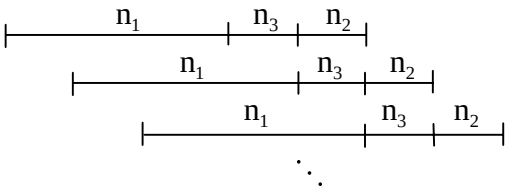

Fig. 2. The Data Structure of the VGA

9. Stop if $Diff \leq 0$; otherwise go back to Step 5.

Figure 2 depicts the data structure used in the VGA. In addition to n_1 and n_2 introduced in the RGA, n_3 is the size of the validation set. In sum, the content of a VGA is concretized by the five parameters: n_1, n_2, n_3, m_1 and m_2, and we will use $VGA(n_1, n_2, n_3, m_1, m_2)$ to denote a specific VGA.

5 Performance Competition

Evaluating the performance of different trading strategies is by no means a simple task. Chen (1998b) classified all performance criteria into two kinds: one which explicitly incorporates the *efficient frontier* in the test, and one which does not. For example, the *Sharpe ratio* and *Jessen's alpha* are criteria of the first kind, and *the mean return* is a criterion of the second. While from the theoretical viewpoint the criteria based on the efficient frontier seem to be more rigorous, many practitioners still only focus on the mean return. This paper only uses the accumulated return at the termination date (9/27/97) as the performance criterion. Volatility or risk has not been taken into account at this stage.

The accumulated return was defined in Chen and Chen (1998) (Equations 4-7). Given this criterion, the r_1^{600} (Equation 7) of the OGA, RGAs and VGAs are computed. As described in Chen and Chen (1998), there are 20 different data sets of D_{ijk} $(i < j < k,\ i, j, k = 1, ..., 6)$. So, given a GA, we first compute the r_1^{600} for each D_{ijk}. We then average the r_1^{600} over these 20 sets of D_{ijk}. The results are given in Table 2.

Among all GAs considered, the non-adaptive GA is the worst performer. By Theorem 6 of Chen (1998), the non-adaptive GA (OGA) is doomed to fail if $\mathcal{D}$ *is not well-ordered enough*. Since the empirical results found in Chen and Chen (1998) show evidence of this possibility, the poor performance of the OGA is not surprising. This result lends support to the use of the *retraining scheme* and the *validation scheme* when landscapes are dynamic. For example, by using RGAs, r_1^{600} can increase up to -159.52 from -660.28, and by using VGAs, r_1^{600} can further increase to 593.63. However, Table 2 also cautions us about the fact that *the difference made by using the retraining scheme and the generalization-enhancing scheme is very sensitive to the setting of parameters*. For example, setting n_2 at 24 can generally lead to a higher accumulated return than others.

Table 2. The Accumulated Return of GAs: 8/29/95- 9/27/97

Style of GAs	r_1^{600}	Rank
OGA	-660.28	24/30
RGA (72, 6)	-642.81	23/30
RGA (72, 12)	-333.41	19/30
RGA (72, 24)	-159.52	16/30
RGA (72, 40)	-352.77	20/30
VGA (72, 24, 24, 200, 100)	431.71	9/30
VGA (72, 24, 24, 200, 100)[∗,20]	9.68	15/30
VGA (72, 24, 24, 200, 100)[∗,10]	201.14	13/30
VGA (72, 24, 24, 50, 10)	364.14	11/30
VGA (72, 24, 24, 10, 10)	247.12	12/30
VGA (72, 40, 40, 200, 100)	-528.93	22/30
VGA (72, 24, 24, 200, 10)	-302.63	18/30
VGA (72, 24, 24, 100, 10)	-254.24	17/30
VGA (72, 40, 40, 10, 10)	593.63	8/30

$RGA(n_1, n_2)$ and $VGA(n_1, n_2, n_3, m_1, m_2)$ are defined in Sections 3 and 4 respectively. $[*, n]$ indicates that, in the initial generation, n strategies are generated by using experts' knowledge. The third column gives the the rank of GAs among 30 selected trading strategies, including 14 GAs given in this table and 16 strategies from experts.

Nevertheless, because of limited trials, it is difficult to give a rigorous evaluation of the setting of parameter values.

To see the relative performance of GAs, we also include 16 trading strategies suggested by experts in the MasterLink Securities Corporation. The last column of Table 2 gives the rank of GAs among these 30 trading strategies. While GAs rank in the middle, the average accumulated return of the 14 GAs (-99.08) significantly beats that of the 16 experts' rules (-727.03).

6 Concluding Remarks

The experiments conducted in this study show that when landscapes are dynamic, the use of the retraining scheme (RGA) and the generalization-enhancing scheme (VGA) can be potentially helpful. However, to take full advantage of these two schemes, we have to know more about the setting of parameter values. It remains a challenging task to demonstrate that those parameter values are functions of some extractable statistical properties. Until these functional relations can be constructed, GAs will still be a black box.

References

1. Chen, S.-H. (1998a), "Can We Believe That Genetic Algorithms Would Help without Actually Seeing Them Work in Financial Data Mining?: Part I, Theoretical

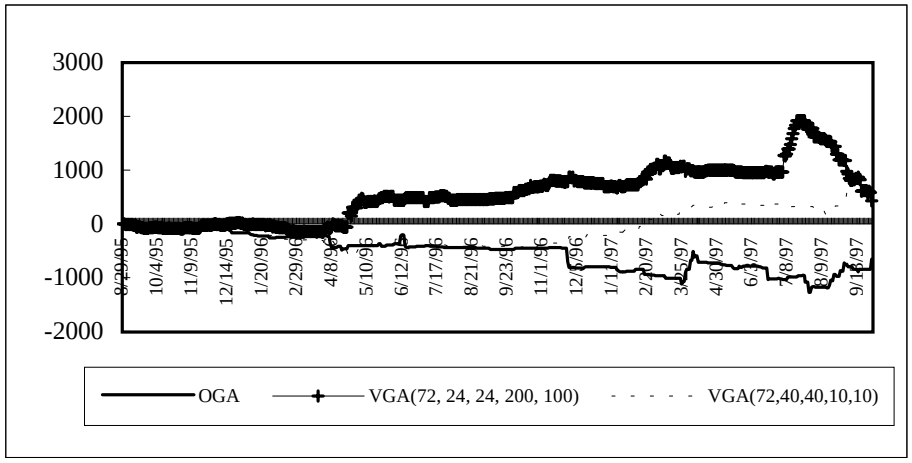

Fig. 3. The Accumulated Returns of GAs

Foundations," *AI-ECON Research Group Working Paper Series* # 9803, National Chengchi University.

2. Chen, S.-H. (1998b), "Criteria to Evaluate the Success of Data Mining in Finance and the Validity of Efficient Market Hypothesis," *AI-ECON Research Group Working Paper Series* # 9806, National Chengchi University.
3. Chen, S.-H, and C.-F Chen (1998), "Can We Believe That Genetic Algorithms Would Help without Actually Seeing Them Work in Financial Data Mining?: Part II, Empirical Tests," *AI-ECON Research Group Working Paper Series* # 9804, National Chengchi University.
4. Chen, S.-H. and W.-Y. Lin (1997), "Financial Data Mining with Adaptive Genetic Algorithms," in T. Philip (ed.), *Proceedings of the ISCA 10th International Conference*, pp. 154-159.
5. Hjorth, J. S. U. (1994), *Computer Intensive Statistical Methods Validation, Model Selection, and Bootstrap*, London: Chapman & Hall.
6. Nelson, M. C. and W. T. Illingworth (1991), *A Practical Guide to Neural Nets*, Reading, MA:Addison.

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