Using GP to Evolve Decision Rules for Classification in Financial Data Sets

Pu Wang, Edward P.K. Tsang, Thomas Weise, Ke Tang and Xin Yao

***Abstract*—Financial forecasting is a lucrative and complicated application of machine learning. In this paper, we focus on the finding investment opportunities. We therefore explore four different Genetic Programming approaches and compare their performances on real-world data. We find that the novelties we introduced in some of these approaches indeed improve the results. However, we also show that the Genetic Programming process itself is still very inefficient and that further improvements are necessary if we want this application of GP to become successful.**

***Keywords*-Genetic programming; Decision rules; Classification; Forecasting; Finance; EDDIE; FGP; AUC; Entropy**

I. Introduction

Professional traders in the financial markets as well as private investors are always looking for efficient indicators giving information about how stock prices are likely to develop. One way to obtain such knowledge, i.e., to solve the financial forecasting problem, is by utilizing data mining techniques [1]. This way, rules or patterns in the history of stock prices can be discovered which can then be used to support investors in their decisions.

By extracting such features [2], we turn the financial forecasting problem in a classification problem, dividing stocks into such that will likely rise in value and in those which probably will not. There are many different approaches we could use for mining classifiers for this purpose, such as Artificial Neural Networks [3, 4], Learning Classifier Systems [5], Support Vector Machines [6, 7], and decision trees[8–11].

We adopt the latter approach for two reasons: First, tree-shaped rules are explicit and easily be understood, which is important in real-world applications where users will not trust suggestions coming from a system they cannot understand. Second, there exists a variety of approaches to efficiently synthesize highly-accurate decision trees. Amongst these methods, Genetic Programming (GP) is one of the most stable and efficient approaches [10–12]. Tsang, Yung, and Li [12–14] used GP in financial forecasting and discovered patterns in stock prices. Li devised FGP (Financial Genetic Programming) on the basis of the works [9, 12, 14] and successfully solved further problems in financial prediction.

We began our research with the aim to improve FGP. In this paper, we explore some methods to do so. In its original form, FGP rates the synthesized classifiers on the basis of the accuracy metric which leads to bad performance in imbalanced data sets. Therefore, we introduce a more reasonable metric for rating the decision rules, the AUC [15]. Second, Li used standard GP to evolve the decision trees. In this work, we adopt some more efficient operators[11] and introduce improvements which have shown excellent properties in classification problems in the past.

This paper is organized as follows: Section II gives an introduction into related work. In Section III, we list metrics which can be used for rating the utility of classifiers and which, therefore, are potential fitness functions for a GP process. In Section IV, we define the technologies and improvements we introduced in FGP and outline our new system (EDDIE-102). We perform various experiments with different configurations of the Genetic Programming system in Section V. On one hand, we show that our enhancements increase the ability of FGP to provide useful financial forecasts. In a critical discussion of the results, on the other hand, we also find that none of the tested configurations is satisfyingly efficient. We then explore possible reasons for this issue and draw conclusions about necessary future work in Section VI.

II. Related Work

The use of GP for evolving classifiers and predictors has a tradition of more than ten years. Since 1998, the Dynamic Data Investment Evaluator (EDDIE, [16]) has been used to discover interactions between variables in decision making processes [2, 16]. It evolves rules which can be utilized for classification or forecasting. These rules are more explicit and clearer than rules produced by other approaches, such as neural networks.

EDDIE-1 is the first implementation of EDDIE [16]. FGP-1, the third EDDIE release, fully focused on financial forecasting and is thus referred to as FGP. It creates generic decision trees which are called GDTs and which record how a forecast was arrived at. FGP-1 takes accuracy as its target performance metric in several financial problems [12–14]. Accuracy is a contentious metric to measure classifiers and

The authors are with the Nature Inspired Computation and Applications Laboratory(NICAL), School of Computer Science and Technology, University of Science and Technology of China(USTC), Hefei, Anhui 230027, China. Edward P.K. Tsang is also with the Department of Computer Science,University of Essex, Wivenho Park, Colchester CO4 3SQ, U.K. Xin Yao is also with the Center of Excellence for Research in Computational Intelligence and Applications (CERCIA) School of Computer Science, the University of Birmingham Edgbaston,Birmingham B15 2TT, U.K. (emails: wuyou308@mail.ustc.edu.cn, tweise@ustc.edu.cn, ketang@ustc.edu.cn, edward@essex.ac.uk, x.yao@cs.bham.ac.uk).

Corresponding author: Ke Tang (Phone: +86-551-3600754).

more and more researchers [10, 11, 15, 17, 18] adopt the Area under an ROC Curve (AUC) for measuring their final results. This measure is considered to be more reasonable than accuracy in comparing learning algorithms [19]. In this work, we will introduce AUC into FGP in order to improve its performance and result quality.

The C4.5 algorithm by Quinlan [8] is a famous tool for classification. Because of its greedy tree construction algorithm, it is faster than most other classification approaches. It utilizes entropy as heuristic which costs much less computation time than AUC. We will therefore also test how using entropy as classifier fitness influences the performance of our Genetic Programming system.

EDDIE-1 and FGP-1 adopt a standard EA and simple operators in the algorithms. Besides the two mentioned novelties, we also introduce the unfitness method discussed by Muni et al. [11] and a modified mutation operator into EDDIE-3.

III. Classifier Metrics and Fitness Function

A classifier performance metric is a value which rates the utility of the classifier. The larger (or smaller) this value is, the better classifier. There are various metrics for measuring classifiers and we describe the ones relevant to our work in the following.

A. Confusion Matrix and Metrics

Table I
Confusion matrix to classify two classes

TN	FP	$TN+FP$
FN	TP	$FN+TP$
$TN+FN$	$FP+TP$	Ntr

.

Table I sketches the blueprint of a confusion matrix for a binary classification problem. Here, true positive (TP) is the number of instances which are positive and which have also been classified as positive, false negatives (FN) are the instance which belong to class positive but are classified as negative, and the meaning of false positives (FP) and true negative (TN) are analogous. Ntr is the total number of instances in the data set. From the confusion matrix, the following metrics can be extracted:

$$\textit{Accuracy} = \frac{TP+TN}{Ntr} \quad (1)$$

$$\textit{True-Positive Rate}(\textit{Recall}) = \frac{TP}{FN+TP} \quad (2)$$

$$\textit{False-Positive Rate}\ (FPR) = \frac{FP}{TN+FP} \quad (3)$$

$$\textit{Rate of Failure}(RF) = \frac{FP}{FP+TP} \quad (4)$$

$$\textit{Rate of Missing chance}(RMC) = \frac{FN}{FN+TP} \quad (5)$$

Accuracy focuses on the rate of correct classification in the whole data set but does not consider the balance [20, 21] of the data set. Assume, for example, that there is a dataset with 90 positive instances and 10 negative instances. A classifier always classifying an instance as positive will then have an accuracy rating of 0.9. Although this rating is high, it is meaningless.

B. Receiver Operating Characteristics (ROC)

Receiver operating characteristics (ROC) can be used to illustrate the performance of a classifier. ROC graphs are increasingly used in machine learning and data mining research [15]. Figure 1 by Garcia-Almanza et al. [22] shows an example ROC graph. ROC graphs depict relative tradeoffs between benefits (true positives) and costs (false positives) [15]. The area under an ROC curve (AUC) [15, 17, 18] is a metric for classifier performance on binary classification considered more objective than accuracy [19]. A perfect classifier has an AUC equal to 1, a random classifier achieves around 0.5 and a classifier with an AUC less than 0.5 is a bad classifier. In the ROC graph, the top-left area denotes good performance.

For each classifier/ensemble, not only one point in the ROC graph exists but a whole curve. The idea is that a classifier returns a score when classifying an instance of the available data. The higher the score of the instance, the more likely it should be classified as positive. If there are M different scores, we can define M different thresholds t. If the score of an instance is higher than t, it is classified as positive. This way, we can create up to M different points in the ROC graph for a classifier or ensemble. The area under the curve defined by these points then is the AUC measure. Details on how AUC is computed can be found in [15].

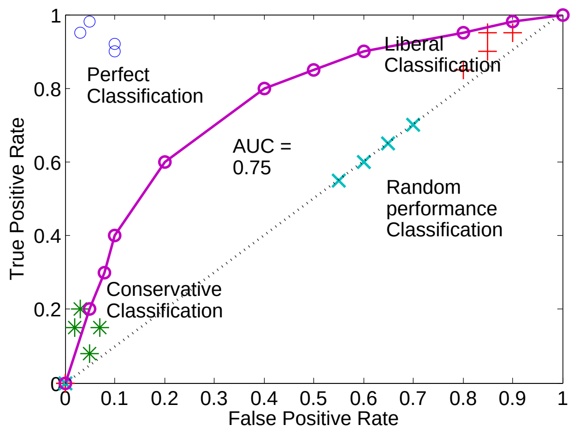

Figure 1. Receiver Operating Characteristic (ROC) space [22]

C. Fitness Function

Evolutionary algorithms use fitness functions in order to determine which candidate solutions are promising for further investigation. Since our goal is to improve the AUC metric of the evolved classifiers, we may use the AUC as fitness directly. We refer to such a fitness function as f_A.

In Li's method [12], the cost-sensitive fitness function f_C given in Equation 6 is chosen:

$$f_C = w_1 * Accuracy - w_2 * RF - w_3 * RMC \qquad (6)$$

w_1, w_2 and w_3 are the weights of the metrics. Since our target is to find the classifier with the maximum AUC, we may instead take the AUC measure as fitness function. On the other hand, we wish to explore the use of the entropy similar to C4.5, since it has a positive correlation with AUC but is much easier to compute.

$$H(X) \;=\; -\sum_{i=0}^{n} p(x_i)\log_2 p(x_i) \qquad (7)$$

Equation 7 gives the definition of the information gain [23] for a random variable X with n outcomes $\{x_i : i = 1....n\}$. The Shannon entropy, a measure of uncertainty is denoted by $H(X)$ and $p(x_i)$ is the probability mass function of the outcomes x_i.

$$p_i \;=\; \frac{P_i}{\sum_{j=0}^{1} P_j} \qquad (8)$$

$$f_E \;=\; \frac{\sum_{k=0}^{NL}\left(1+\sum_{i=0}^{1} p_i\log_2 p_i\right)\left(\sum_{j=0}^{1}(P_j-1)\right)}{Ntr-NL} \qquad (9)$$

We can easily introduce a fitness function very similar to Shannon's formula as given in Equation 9.It's a novel decision tree structure. In a decision tree T as used in our EDDIE-102 system, two numbers are assigned to each of the NL leaf node: P_0 is used to record how many negative instances arrive at it and the other (P_1) is used to record how many positive instances arrived. The range of the entropy-based fitness f_E is from 0 to 1. When $f_E = 1$, the classifier has the best training result, $f_E = 0$ means the classifier is a random one.

IV. The Improved Algorithm: EDDIE-102

The main routine of our improved approach (called *EDDIE-102*) is given in 2. It is a version of standard Genetic Programming process extended with a local search and an ensemble approach.

A. Crossover Steered by Badfitness

We use the badfitness bf of an individual T applied to the training data D as guide for parent selection as the candidate individuals for crossover or mutation. This idea stems from [11]. According to 10, the badfitness of an individual here stands for how far away it is from achieving the best fitness.

$$bf \;=\; 1 - f_E \qquad (10)$$

```
Algorithm EDDIE-102($M$, $N$)
$M$ is the maximum generation
$N$ is the size of the ensemble
BEGIN
  Let $gen = 0$
  Initialize the population using the grow method
  while ($gen < M$)
    Evaluate fitness($f_E$) of each individual
    Update archive of Best $N$ individuals
    Survival Selection : Tournament Selection by $f_E$
    Crossover and Mutation
    Adjust Individuals with Hill-Climbing
    $gen = gen + 1$
  end while
Ensemble the Best $N$ individuals on test data set
END
```

Figure 2. The EDDIE-102 Algorithm

We also utilize some of the ideas from [11]. Crossover plays important role in genetic programming. We produce offspring according to a given crossover rate. All candidate parents to be made crossover on are selected via another tournament which is one by the participating individual with the highest badfitness. This will give more chance for decision tree with lager badfitness to be changed, i.e., to likely be improved. The crossover operation swaps two subtrees from the two different individuals.

B. Modified Mutation

Subsequently, individuals are selected for mutation according to the mutation rate p_m. The mutation operation is usually a destructive process. Since we do not want to destroy the good parts of a decision tree, we create several mutated offspring of an individual in order to find one whose fitness is better than the one of the parent. Many mutation operations, however, cost much training time. In [11], a strategy to reduce computation time is adopted: 50% of the training cases are used to evaluate the fitness f_m of the mutated individual and the fitness f_o of the original individual. If f_m equals f_o, the remaining 50% of the training cases are used to get f_m. If the mutated individual is equally good or better than its parent, it is kept. Otherwise, the mutation operation is ignored with the probability p_t. In our experiments, we set $p_t = 0.5$.

C. Ensemble

Our system does not only return a single classifier as result of the optimization process, but instead uses the aggregated knowledge in the population by combining the N best classifiers to an ensemble. In this section, we describe an ensemble method to give scores to training and test cases as needed for computing the AUC metric and ROC graphs. The decision trees used by us are different from traditional decision trees, as already mentioned in the discussion of 9.

In Figure 3 we sketch such a tree where every leaf node records how many positive cases (in P_1) and negative cases (in P_0) arrive at it. The leaf nodes will then return a score S_c= P_1/(P_1+P_0).

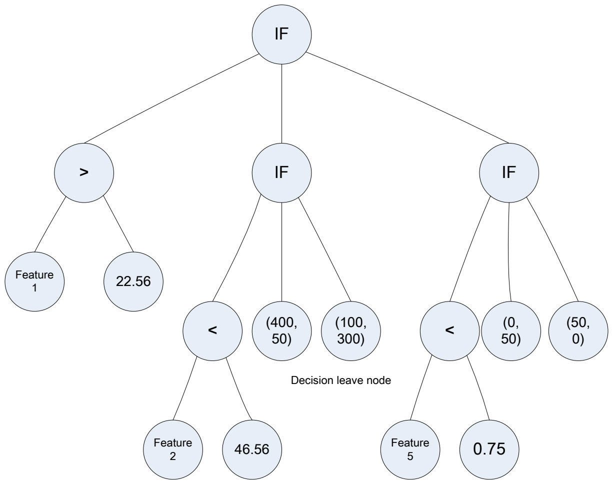

Figure 3. Decision Tree

Our ensemble method is using several individuals to decide the final score for one training or test case C, here we describe a strategy to obtain the scores as follows, where N is the number of individuals in the ensemble and $S_i(C)$ is the score achieved by the ith individual classifier.

$$Score(C) = S_i(C) : i = \arg\max_{j \in 1..N} abs(S_j - 0.5) \qquad (11)$$

V. Experiments

A. Data sets

To conduct our experiments we use four examples and four datasets defined as follows:

S&P-500 -2700 cases. This dataset stems from the Heng Seng Index. Available to us were data from the 2nd April 1963 to the 25th January 1974 (2700 data cases). The attributes of each case are composed by indicators which are derived from financial technical analysis. The indicators are based on the daily closing price. We tried to find classifiers which detect an increase of stock value of 4% in a horizon of 63 days. We divided the data into 1800 cases for training and put 900 cases into the test data set.

DJIA -2800 cases. We obtain this dataset from the closing prices of the Dow Jones Industrial Average. We choose two ranges in DJIA to our experiments. One has 2800 cases (from 7th of April 1969 to the 5th May 1980), the other consists of 3035 cases (from the 7th April 1969 to the 9th April 1981). In the first dataset (called D1), we on one hand, again aim to detect an increase of 4% in a horizon of 63 days (setup D11). On the other hand, we try to classify according to an increase of 2.2% in a horizon of 21 days (setup D12). In the second datasets, we target an increase of 2.2% in an horizon of 21 days (setting D2).

It's difficult to find patterns in the financial data sets because of its timeliness. Li used FGP-1 to find some patterns in above data sets [12], and this work is the extend of FGP-1, so we adopt these data sets to examine EDDIE-102 working in financial data sets. A horizon of 21 days is a short-term prediction and 63 days is a middle-term perdition.

B. Parameters

We adopt the three different fitness functions (f_A, f_C, and f_E as defined in Section III-C) in FGP and run it on the four data sets. First, we run FGP with the cost sensitive fitness function (f_C, FGP-1), second we use AUC as the fitness function (f_A, FGP-AUP), and third, we applied Equation 9, i.e., f_E (FGP-E). Besides FGP, we also tested our new approach EDDIE-102 which is based on the entropy fitness as well. In Table , we provide the parameters of the four algorithm configurations used.

Table II
Parameters for four algorithms

Objective	Find decision trees which has the higher AUC
Terminals	0,1with 1 representing "Positive"; 0 representing "Negative"
Function set	If-then-else , And, Or, Not, $>$, $<$, =.
Data sets	S&P500 D11 D12 D2
Algorithms	FGP-1 ,FGP-AUC ,FGP-E ,EDDIE-102
Crossover rate	0.9
Mutation rate	0.1
Parameters for GP	P(Population size) = 1000; G (Maximum generation) = 50 Number of Runs = 10
Termination criterion	Maximum of G of generation has been reached
Selection strategy	Tournament selection, Size = 4
Max depth of individual program	17
Max depth of initial individual program	3

C. Results

We compare the algorithms using four indicators,

1) the AUC on training data,
2) the AUC on the test datasets,
3) the training time, and
4) the final decision tree's size.

In all diagrams we provide as evaluation results, the blue line with the plus markers stand for the performance of FGP-1, the green line with star markers represent FGP-AUC, the red line with circle markers belongs to FGP-E, and cyan lines with dots are results obtained with EDDIE-102.

The incentive to apply all four GP algorithms to stock price prediction was to get a clear, unbiased understanding of their performance and of the hardness of the problem.

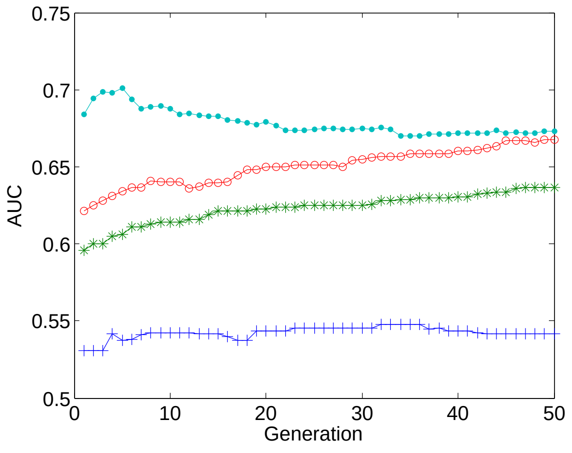

Figure 4. Training Results on S&P500

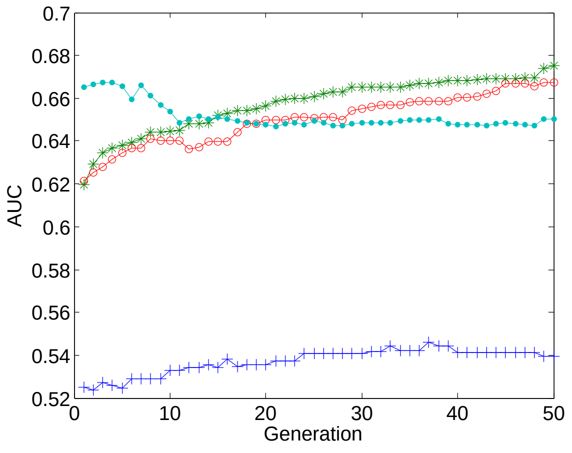

Figure 7. Training Results on DJIA D11

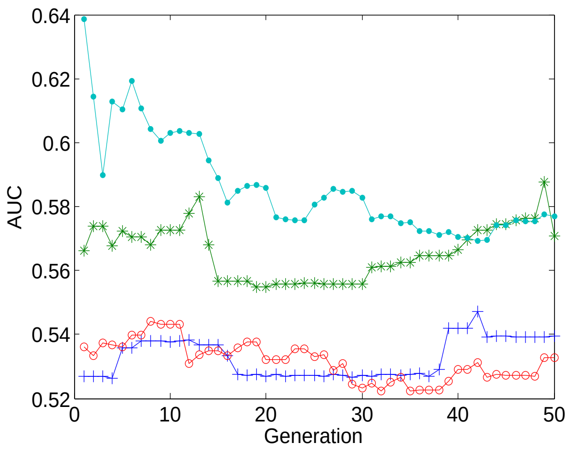

Figure 5. Test Results on S&P500

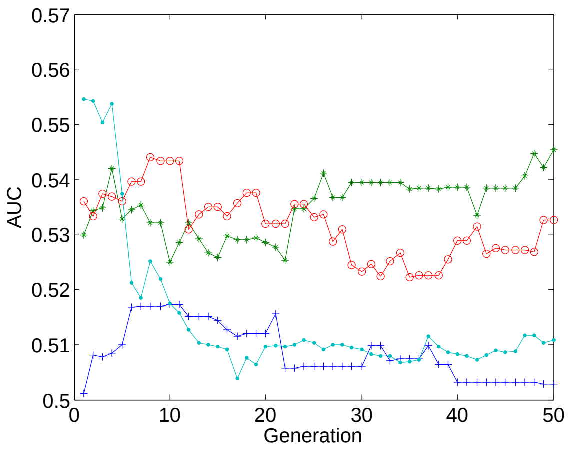

Figure 8. Test Results on DJIA D11

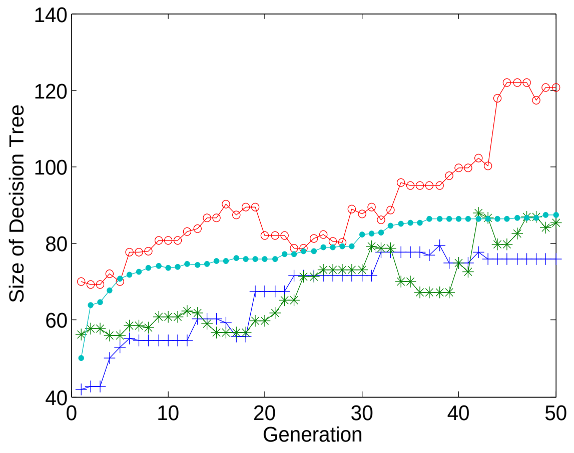

Figure 6. Decision Tree Size on S&P500

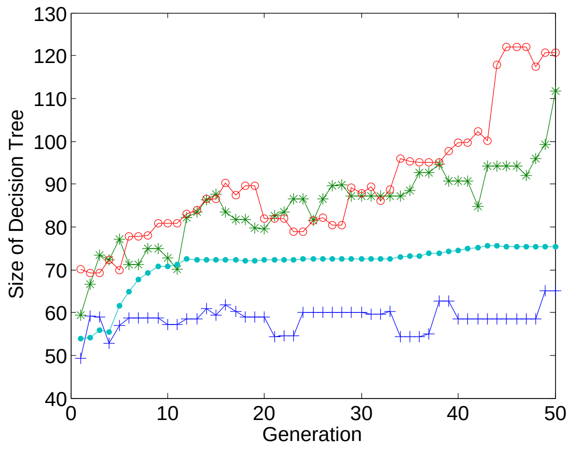

Figure 9. Decision Tree Size on DJIA D11

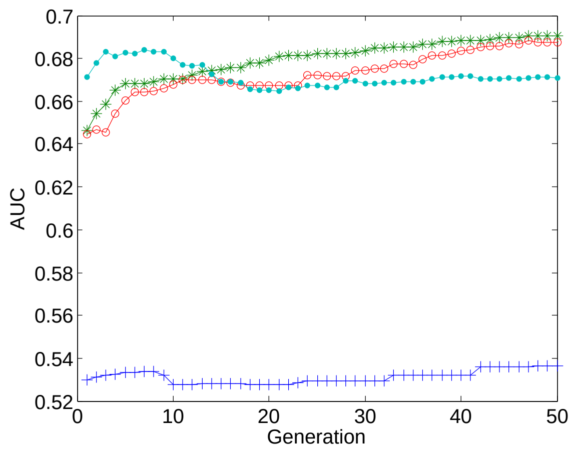

Figure 10. Training Results on DJIA D12

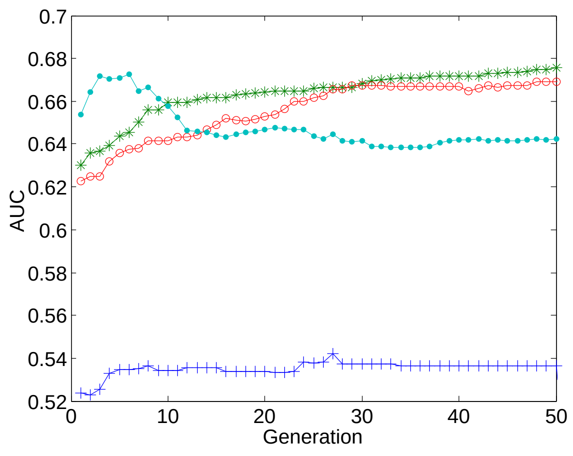

Figure 13. Training Results on DJIA D2

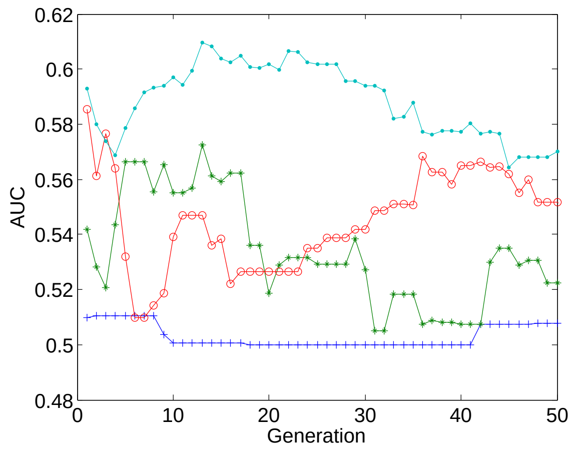

Figure 11. Test Results on DJIA D12

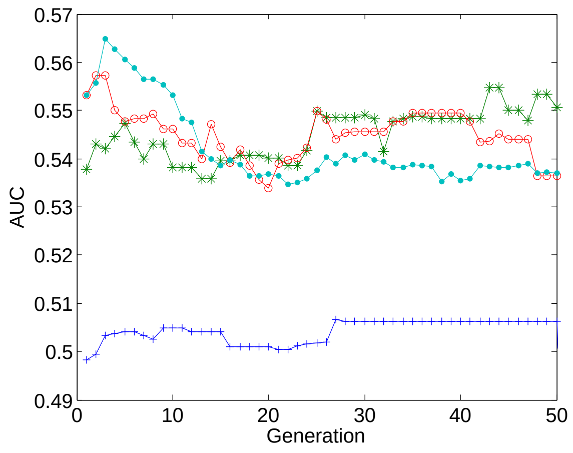

Figure 14. Test Results on DJIA D2

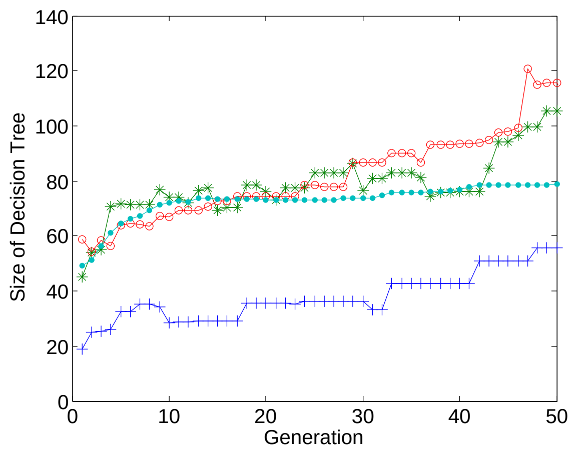

Figure 12. Decision Tree Size on DJIA D12

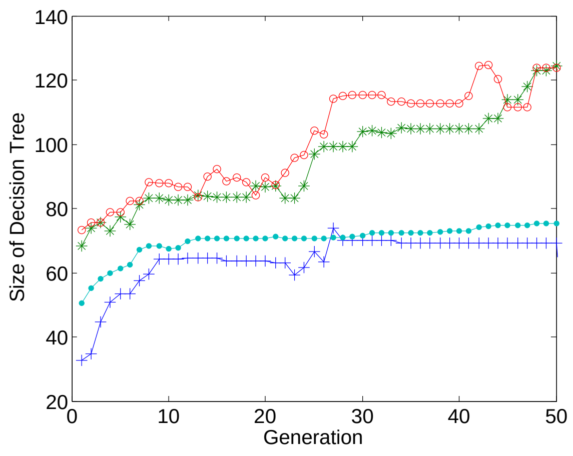

Figure 15. Decision Tree Size on DJIA D2

Table III
TRAINING TIME ON FOUR ALGORITHMS

Time [ms]	$S\&P500$	$DJIAD11$	$DJIAD12$	$DJIAD2$
FGP-1	2227339	2228840	2282016	2479239
FGP-AUC	3884162	3836702	4344654	5077303
FGP-E	2613086	2734400	2745200	2839350
EDDIE-102	4839719	5247922	5377094	6833672

The primary goal was, of course, to find an algorithm configuration which can produce better results than the de-facto standard FGP-1.

Figures 4 to 15 illustrate the average performance of the classifiers discovered over the course of the fifty generations for each of the four algorithms. If we consider the end products of the evolution, i.e., the classifiers, we can clearly see that the three new algorithms improve FGP-1 on the AUC measure on the test data.

When we compare the behaviors of the AUC measure of the evolved classifiers on the training set and those on the test set, we find a striking difference. Often, the AUC on the *training* data is increasing whereas the AUC in the *test* data behaves arbitrarily – compare, for instance, Figures 7 and 8. This process does not just set in after some time, but takes place right from the start.

There are two possible reasons for this strange behavior. First, we should be aware that financial forecasting problems are complicated by nature. There are lots of factors (directly or indirectly) influencing the stock market which are not covered by the features we can use for classification, such as political events, natural influences such as shortages of certain products and the weather, and market sentiments. The test data and the training data are therefore less correlated than in other problems. Thus, we cannot a priori expect that classifiers successful in training are also successful in practice and vice versa. Also, the prediction character of the task does not allow us to arbitrarily split the data into different test and training sets or to perform crossvalidation. Verifying the hypothesis that financial forecasting is harder than other problems will be part of our future work, in which we will compare this type of problem with data sets from the well-known UCI repository [24].

Another reason why the GP process is not really able to improve its initial, randomly created candidate solutions by much could code bloat. Figure 6, 9, 12, and 15 all show that the sizes of the decision trees grows fast. The fast tree growth may create the impression that overfitting is taking place. However, we think that this is not the case since there is almost no improvement of the AUC metric in the test data, i.e., all trees would have to be overfitted right from the start. Yet, it is well known that code bloat reduces the chance that the reproduction operations can improve (or even change) the behavior of a program [25]. Since we assume that this may be the reason for the apparent inefficiency of the optimization process, it may be necessary to apply methods which limit the tree size and/or reduce the speed of tree growth.

In Table III, we compared the four algorithms according to the required training time. Because of the use of the new multi-mutation operator and the ensemble method, EDDIE-102 has the highest time consumption. Amongst the FGP-based approaches, FGP-AUC is most expensive in terms of runtime, since computing the AUC metric is more complicated than other two fitness functions.

VI. CONCLUSION AND FUTURE WORK

From our experiments we can learn two lessons.

1) We can find classifiers for financial forecasting with Genetic Programming.
2) However, the process of evolutionary optimizing them proofed to be highly inefficient.

With EDDIE-102, for instance, we find good solutions in the first generation, but these are step-by-step ”optimized” to worse and worse performance. Therefore, we want to emphasize that it is not sufficient to just point out that a certain problem can be solved with GP. Instead, thorough research also requires us to give information on *when* GP discovers its solutions and *how* it arrives at them. This last point not only holds for classification, but for all problem areas in general. By providing this information in this paper, we aim at stirring up a discussion on how the Genetic Programming can be made more efficient for financial forecasting and what the most challenging characteristics of this problem domain are for GP.

In this paper, we extended the work of Li [12]. In his FGP-1 algorithm, the accuracy metric is used to measure the decision trees. Accuracy is not a good measure for classificatory performance. In our experiments, we therefore use AUC and it turns out that GP solutions to financial forecasting are less efficient than expected. We designed two other fitness functions for FGP and developed a new algorithm called EDDIE-102.

We tested all four resulting configurations thoroughly with experiments. According to our results, none of the algorithms can always outperform the others. Although we were able to provide some improvements with our new approaches, the problem can still not satisfyingly be solved.

In our future work, we therefore plan to further analyze what makes financial prediction problems hard and, based on this analysis, suggest a more efficient algorithm. A first step may be to impose more restrictive limits on the tree growth in our system, since all four algorithms seem to be vulnerable to bloat[26–29].

ACKNOWLEDGMENT

This work is partially supported by two National Natural Science Foundation of China grants (No. 60802036 and No. U0835002).

References

[1] M. Kantardzic, *Data mining: concepts, models, methods, and algorithms*. Wiley-Interscience, 2003.

[2] E. P. K. Tsang and J. Li, "Eddie for financial forecasting," S.-H. Chen, Ed. Genetic Algorithms and Programming in Computational Finance: Kluwer Academic Publishers, 2002, ch. 7, pp. 161—174.

[3] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[4] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge University Press, 2008.

[5] J. H. Holland, L. B. Booker, M. Colombetti, M. Dorigo, D. E. Goldberg, S. Forrest, R. L. Riolo, R. E. Smith, P. L. Lanzi, W. Stolzmann, and S. W. Wilson, "What is a learning classifier system?" *Lecture Notes in Computer Science*, vol. 1813, 2000.

[6] N. Cristianini and J. Shawe-Taylor, "An introduction to support vector machines and other kernel-based learning methods." Cambridge University Press, 2000.

[7] B. Catanzaro, N. Sundaram, and K. Keutzer, "Fast support vector machine training and classification on graphics processors," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 104–111.

[8] J. Quinlan, *C4.5: programs for machine learning*. Morgan Kaufmann, 1993.

[9] J. Li, X. Li, and X. Yao, "Cost-sensitive classification with genetic programming," *Proceedings of the Congress on Evolutionary Computation*, pp. 2114–2121, 2005.

[10] C. Zhou, W. Xiao, T. M. Tirpak, and P. C.Nelsom, "Evolving accurate and compact classification rules with gene expression programming," *IEEE Transactions on Evolutionary Computation*, vol. 7, pp. 519–530, 2003.

[11] D. P. Muni, N. R.Pal, and J. Das, "A novel approach to design classifiers using genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 183–196, 2004.

[12] J. Li, "Fgp: a genetic programming based tool for financial forecasting," Ph.D. dissertation, 2001.

[13] E. P. K. Tsang, P. Yung, and J. Li, "Eddie-automation, a decision support tool for financial forecasting," *Decision Support Systems*, vol. 37, 2004.

[14] J. Li and E. P. K. Tsang, "Investment decision making using fgp: A case study." *Proceedings of the Congress on Evolutionary Computation*, vol. 2, pp. 1253–1259, 1999.

[15] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, pp. 861–874, 2006.

[16] E. P. K. Tsang, J. Li, and J. Butler, "Eddie beats the bookies," *International Journal of Software, Practice & Experience*, vol. 28, pp. 1033–1043, 1998.

[17] J. Hanley and B. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve," *Radiology*, pp. 29–36, 1982.

[18] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recogn*, pp. 1145–1159, 2003.

[19] C. Ling, J. Huang, and H. Zhang, "AUC: a better measure than accuracy in comparing learning algorithms," *Advances in Artificial Intelligence*, pp. 991–991, 2003.

[20] N. Japkowicz *et al.*, "Learning from imbalanced data sets: a comparison of various strategies," in *AAAI workshop on learning from imbalanced data sets*, 2000, pp. 00–05.

[21] N. Japkowicz, "Class imbalances: Are we forcusing on the right issue?" *Pro.Omt'l Conf.Machine Learning,Workshop Learning from Imbalanced Data Sets II*, 2003.

[22] A. L. Garcia-Almanza, E. P. K. Tsang, and E. Galvin-Lopez, "Evolving decision rules to discover patterns in financial data sets," *Computational Methods in Financial Engineering*, pp. 239–255, 2008.

[23] C. E. Shannon, "Prediction and entropy of printed english," *The Bell System Technical Journal*, 1951.

[24] UCI, "Uc irvine machine learning respository," 2009. [Online]. Available: http://archive.ics.uci.edu.c/ml

[25] T. Weise, *Global Optimization Algorithms – Theory and Application*. www.it-weise.de, 2009.

[26] W. Langdon, "Quadratic bloat in genetic programming," in *Proceedings of the Genetic and evolutionary Computation Conference (GECCO-2000)*. Citeseer, 2000, pp. 451–458.

[27] H. Iba, "Bagging, boosting, and bloating in genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2, 1999, pp. 1053–1060.

[28] J. Daida, H. Li, R. Tang, and A. Hilss, "What makes a problem GP-hard? validating a hypothesis of structural causes," in *Genetic and Evolutionary ComputationłGECCO 2003*. Springer, 2003, pp. 205–205.

[29] L. Liu, H. Cai, M. Ying, and J. Le, "RLGP: An Efficient Method to Avoid Code Bloating on Genetic Programming," in *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, 2007, pp. 2945–2950.