A Memetic Genetic Programming with Decision Tree-based Local Search for Classification Problems

Pu Wang and Ke Tang
NICAL
School of Computer Science and Technology
University of Science and Technology of China
Hefei, Anhui 230027, China.
Email: wuyou308@mail.ustc.edu.cn
ketang@ustc.edu.cn

Edward P.K. Tsang
Department of Computer Science
University of Essex
Wivenho Park, Colchester CO4 3SQ
Email: edward@essex.ac.uk

Xin Yao
CERCIA
School of Computer Science
University of Birmingham
Edgbaston Birmingham B15 2TT, U.K.
Email: x.yao@cs.bham.ac.uk

***Abstract*—In this work, we propose a new genetic programming algorithm with local search strategies, named Memetic Genetic Programming(MGP), for classification problems. MGP aims to acquire a classifier with large Area Under the ROC Curve (AUC), which has been proved to be a better performance metric for traditionally used metrics (e.g., classification accuracy). Three new points are presented in our new algorithm. First, a new representation called statistical genetic decision tree (SGDT) for GP is proposed on the basis of Genetic Decision Tree (GDT). Second, a new fitness function is designed by using statistic information from SGDT. Third, the concept of memetic computing is introduced into SGDT. As a result, the MGP is equipped with a local search method based on the training algorithms for decision trees. The efficacy of the MGP is empirically justified against a number of relevant approaches.**

***Index Terms*—Genetic Programming; Memetic Algorithm; AUC; Classification**

I. Introduction

Classification is an important data mining task, with roots in machine learning [1–3]. It predicts the categorical attribute (class attribute) which contains two or more groups also called classes based on the values of other attributes (predicting attributes). The training set is a data set used to induce a classifier while the test set is a data set used to measure the quality of the classifier obtained. There are many different kinds of approaches to induce classifiers, such as Artificial Neural Networks [4, 5], Learning Classifier Systems [6], Support Vector Machines [7], decision trees [8, 9], and Evolutionary Algorithms (EA) for learning [10–13].

Genetic programming (GP) is an evolutionary learning approach [14]. There are a large quantity of work on GP for classification problems [13]. One of the main advantages of GP is its flexibility that allows a classification system to be represented in different ways. Linear or graphic structures can be used to construct the GP individuals. However, tree is the most common general data structure for GP representation. Tree-like structure can also be expended for three purposes. The first is GP for extracting decision trees, the second is learning Rule-Based systems, the last is learning discriminant functions [3]. Li invented Genetic Decision Tree (GDT), which is a meta-structure combines decision tree and rule based system [15] and used in the series of Evolutionary Dynamic Data Investment Evaluator (EDDIE). EDDIE is one of GP learning systems[16]. Li expended EDDIE to the third version, Financial Genetic Programming (FGP) [15] which is a successful algorithm to solve some financial forecasting problems, also works on the classification problems.

Though EDDIE and FGP are successful on some problems such as horse racing and financial forecasting, there are some weaknesses in them. First of all, the fitness functions of EDDIE and FGP guide the search towards a classifier with good overall accuracy. However, accuracy itself has been demonstrated to be an inappropriate performance metric [17]. Instead, the Area Under the ROC Curve(AUC) is now well acknowledged as a better performance metric because it is insensitive to class distribution and misclassification costs [18]. Given their fitness functions, EDDIE and FGP might be ineffective for producing classifiers with large AUC. Second, the fitness function in FGP cannot evolve GDTs efficiently for a long generation. Typically, the algorithm stops generating better solutions after a few number of generations. The last is that hill-climbing in FGP is not good enough for local search. Local search in FGP just adjusts the cut surfaces by moving them following their previous direction in the sample space. It is limit to help us to divide the whole sample space into the positive and negative sample subspaces. We need the local search which has the strong power at cutting sample space. A splitting operator is used to enhance the local search in the search process.

To further improve FGP and adapt it to achieve classifiers with large AUC, we proposed the Memetic GP (MGP) in this paper. First, we modify GDT into statistic GDT (SGDT) which gives output by the information collected in the training phase. Second, collected information is used to design new fitness function which has stronger positive relationship with AUC. Third, Memetic [19] idea is emphasized in this paper. A splitting operator strengthens the exploitation in the search process to make up for the hill-climbing (shifting operator) in FGP. The efficacy of MGP has been evaluated and compared with FGP and three other algorithms.

This paper is organized as follows: Section II gives an introduction of some related work. In Section III , we show our new representation for our algorithm. Section IV describes the new fitness function which is designed for the new representation and traditional decision tree. Section V will show the new local search operator for SGDT. Section VI gives our algorithm. Experiments and results are described in Section VII. In Section VIII, we give the conclusions and future work.

II. Related Work

Using GP to evolve classifiers has been developed over 15 years. Since 1995, Evolutionary Dynamic Data Investment Evaluator (EDDIE) [16] had been invented as a system using GP to evolve decision rules to help investors with the evaluation of investment opportunities. EDDIE is an umbrella under which a number of programs have been developed. EDDIE-1 is the first implementation invented by Butler and Tsang, and be used for forecasting in horse racing [16]. Tsang and Li [15, 20] enhanced EDDIE and applied it to financial forecasting, to obtain the third EDDIE release, FGP-1 (also called EDDIE-3). FGP-1 used GP to evolve the GDT as the classifier that syncretizes decision tree and rule.

Fig. 1 to Fig. 3 show three kinds of representations. They are discriminant function [3], classification rule [1, 21, 22] and decision tree [14, 23]. All of them can be used for classification problems. Here, A_i is the attributes in the data sets, and C_j is the label or class ($0 \leq i \leq$ number of attributes in the training data, $0 \leq j \leq$ number of classes in the training data).

Discriminant function is a mathematical expression in which different kinds of operators are applied to the attributes of a data instance that must be classified. Classification rule is also a formalism for representing classifiers that contain logical conditions. Decision tree is one of that most common representations as a classifier which combines several classification rules. For binary classification problems, a single discriminant function with a threshold can be enough, or a single classification rules or a single decision tree. To solve the n-class classification problem, we need $n-1$ discriminant functions or classification rules. We also can use a single discriminant function with $n-1$ thresholds or a single decision tree. However, all of these classifiers will return a class label for a test instance.

GDT is the combination of decision tree and classification rule. Fig. 4 shows a GDT which has several selector branches. And a test instance will get a class label from a GDT classifier. GDT can be split into several decision trees, it means GDT can be efficacy as several decision trees and is better than decision tree. Though GDT has its advantages, it also returns a class label to a test instance, that is weak for getting a good AUC [9]. As mentioned in Section I, it will use SGDT to give every test instance a probability score which a instance is classified to positive class with.

Guiding the evolutionary search in FGP-1 is a cost sensitive fitness function which is designed for the inventors to make financial investment strategies. But it is not suitable for the classifier to get a large AUC. AUC is considered to be more

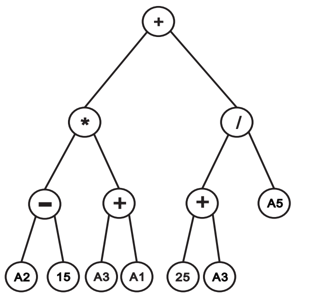

Fig. 1. GP individual representing a discriminant function

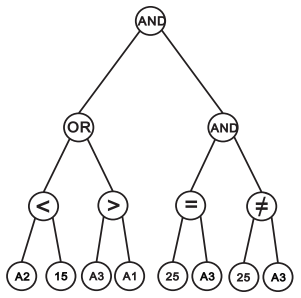

Fig. 2. GP individual representing a classification rule

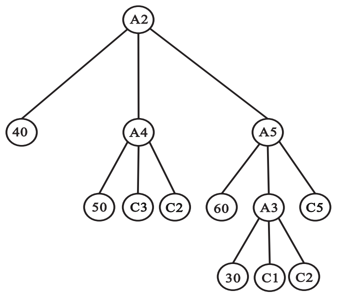

Fig. 3. GP individual representing a decision tree

appropriate than accuracy in comparing learning algorithms [18, 24]. In this work, we will introduce AUC into FGP in order to improve its performance and result quality. Many metrics are involved as the fitness function to evolve the GP classifier such as $accuracy$, $G\text{-}mean$ and $F\text{-}measure$ [3].

The C4.5 algorithm by Quinlan [8] is a famous tool for classification. Traditional decision tree is encoded as the C4.5 classifier which is constructed by greedy algorithm, and it is much faster than most other classification approaches. C4.5 utilizes information gain as the heuristic search strategy which leads to pure leaf nodes. This costs much less computation time than AUC. On the other hand, information gain has positive relationship with AUC, so there is an idea to design a new fitness function from the information gain. The new fitness function has strong positive relationship with AUC and costs less time. It will be talked about this in Section IV.

Memetic Algorihtms (MA) not only pay attention to evolution but also emphasize individual learning. In recent years, many dedicated MAs have been exploited to solve different kinds of problems [25, 26]. FGP-1 used hill-climbing as its local search operator which is not very effective, so we will develop a new local search operator to enhance the search process. The construction of C4.5 also has a splitting strategy which are more focused on deep exploiting. We also involve the splitting strategy in our new algorithm. It is used to divide the search space into more subspaces and get better results. Because of our novel design, a SGDT's fitness is expected to be better when the splitting strategy works on it successfully.

III. New Representation-SGDT

SGDT has the same structure as GDT, and the only difference between them is that SGDT will not give a categorical class label to a test instance. The SGDT will collect the information while it is trained. Every decision leaf node records the instances which have fallen into it, and the SGDT estimates the probability to the test instances according to these information. Such a minor change facilitates our proposal of a novel fitness function for FGP. Fig. 5 shows a SGDT.

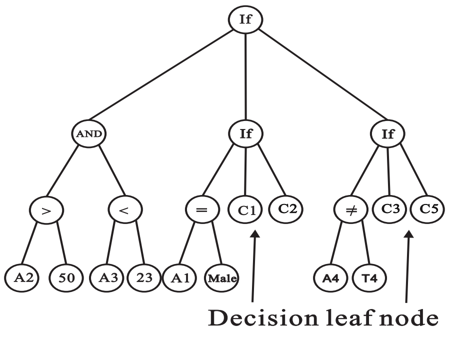

Fig. 4. A GDT individual

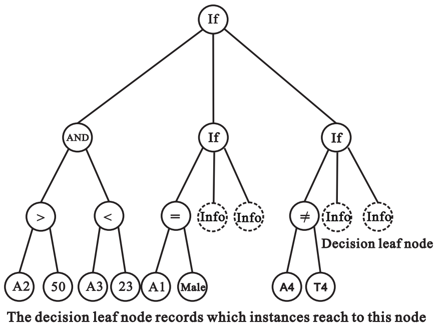
If
AND
If
If
>
<
=
Info
Info
≠
Info
Info
Decision leaf node
A2
50
A3
23
A1
Male
A4
T4
The decision leaf node records which instances reach to this node

Fig. 5. A SGDT individual

IV. Fitness Function

Fitness function plays an important role in evolutionary algorithms (EAs). It represents how an individual adapts to the requirements. Selection operator in EAs most takes the basis of the fitness function, so fitness function can guide the search and decide the final result to a certain degree. There are several metric from confusion matrix such as $accuracy$, $precision$, $recall$, $specificity$, rf and rmc , they are referred to [15, 22]. Many fitness functions for GP are designed by using these metrics. $F\text{-}measure$, introduced in [27], is the harmonic mean of $precision$ and $recall$. $G\text{-}mean$ [28] is the geometric mean of $specificity$ and $recall$. F_{cs} is the linear combination of $accuracy$, rf and rmc.

$$F-measure = \frac{2 \times recall \times precision}{recall + precision} \tag{1}$$

$$G-mean = \sqrt{recall * specificity} \tag{2}$$

$$F_{cs} = w_1 * accuracy + w_2 * rf + w_3 * rmc \tag{3}$$

Eq. 2, Eq. 3 are very common metrics as the fitness function to evaluate the performance of the classifier, and Eq. 3 is a cost sensitive method. These are arbitrary prediction based metrics. Taking them as the fitness function to evolve the classifier, it is hard to get a good result, because they have so weak relationship with AUC.

A. Our new fitness function

AUC are not taken as the fitness function though it is our target object, because it costs a lot of time to compute AUC of an individual. The upper bound of complexity of AUC computation is O($m$log$m$), where m is the size of data set [24]

$$p(l,k) = \frac{P(l)[k]}{\sum_{i=1}^{2} P(l)[i]} \tag{4}$$

$$F_E(x) = \frac{\sum_{\forall leaves\ l \in x} \left(1 + \sum_{k=1}^{2} p(l,k)\log_2 p(l,k)\right)\left(\sum_{k=1}^{2} P(l)[k] - 1\right)}{|A| - |leaves \in x|} \tag{5}$$

We want to design a fitness function which has a strong positive relationship with AUC and also has a low computation complexity. A fitness function based on the entropy method from the traditional decision tree was invented. The range of Eq. 5 is from 0 to 1. When $F_E(x) = 1$, the x has the maximum AUC with 1, $F_E(x) = 0$ means x is a random classifier, here x is a SGDT, $P(l)[i]$ is the number of the instances with label i ($i = 1$ or 2) have fallen in the lth leaf in classifier x. The computation complexity of this $F_E(x)$ is O($m + n$), n is the size of decision leaf nodes in x, $|A|$ is the size of training data set. Every instance will get a score from the SGDT by the information of the decision leaf node which it has arrived at, and the score measures the probability of the instance belonging to a certain class. $F_E(x)$ pays more attention on the detail of the partition of the sample space and the probability of the output. This leads to a stronger and positive relationship with AUC than $F\text{-}measure$, $G\text{-}mean$ and F_{cs}.

V. Local Search

In the left hand of Fig. 6, the circles are positive instances, and the crosses are negative instances. A classifier (SGDT) is instant of two blue solid lines which divide the sample space into three subspaces. All the instances in S_1 belong to positive set, where S_1 is a pure space, and so is S_3. S_2 is not so pure because positive and negative instances are mixed. Our target is to divide the data space into several subspaces, make every subspace as pure as possible and minimize the number of subspaces.

A. Shifting

In the left hand of Fig. 7, it does shifting operators twice, and the solid lines are moved to dotted lines. In the right hand of Fig. 7, in the red circle, it does a shifting operator on a SGDT. Shifting is the hill-climbing operator in [15]. Hill-climbing operator in FGP is used to adjust the thresholds in

GDT or SGDT. In Fig. 7 threshold T_4 is modified as $\underline{T_4'}$, this operator is expected to make every subspace more pure by adjusting the cutting surfaces.

B. Splitting

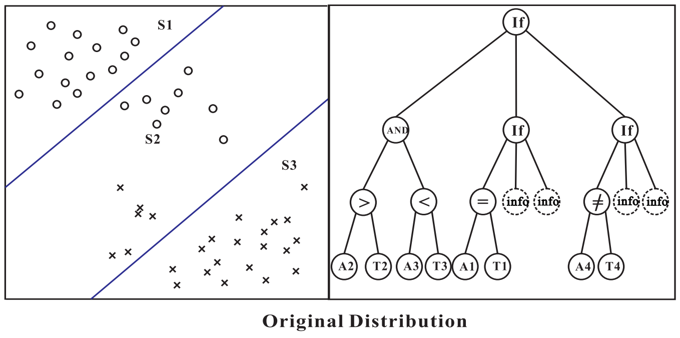

Fig. 6. Original SGDT

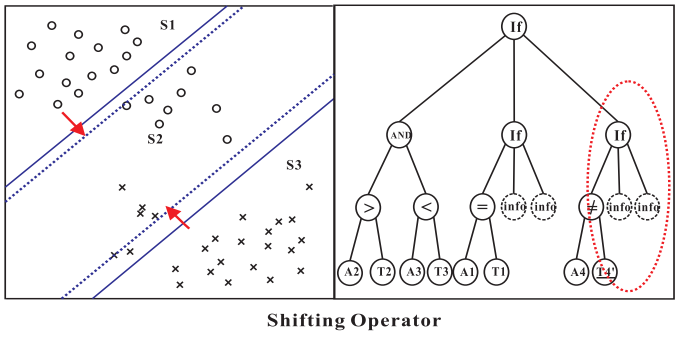

Fig. 7. Shifting on SGDT

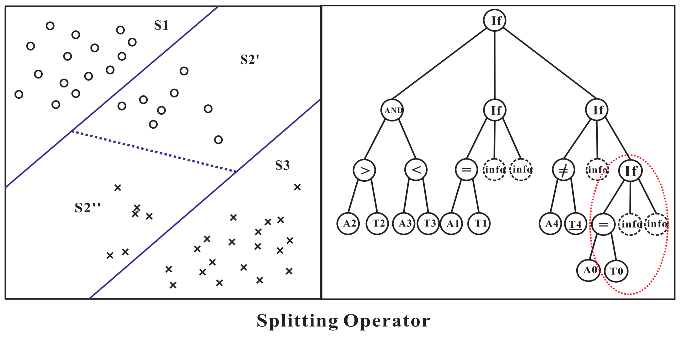

Fig. 8. Splitting on SGDT

Splitting is another local search operator. It is used to split a subspace into two or more subspaces. It works on one subspace and keeps others. In the left hand of Fig. 8, it does shifting operators once, a dotted lines divides S_2 into two subspaces which are pure. In the right hand of Fig. 8, a splitting operator is done on a SGDT that in red circle. Splitting operator is the reinforcement of shifting. When shifting operator is not efficient for an individual, splitting operator will be used to divide the subspace which lead to increment of subspaces number.

Generally, the space with more instances has more probability to be selected to do splitting operator. When a space with fewer instances is divided into several subspaces; , it will result in overfitting for the classification problems easily. On the other hand, a pure space is not necessary to be divided, so it needs to define how pure a space is should to be done by splitting operator. If a space has information gain greater then ST (ST = 0.1 in our paper), the splitting operator will be called.

A splitting operator works on a subspace or a decision leaf node. It adopts splitting ideas from C4.5. There are several instances in the decision leaf node, and splitting operator can choose the likely feature and threshold which can get the maximum information gain. The fitness of a SGDT will increase when splitting operator is worked on it successfully.

VI. MGP Algorithm

The proposed on SGDT, we have designed a new fitness function which has a positive relationship with AUC, and also involve some memetic techniques to improve SGDT performance. A Memetic Genetic Programming (MGP) for classification problems is described as Algorithm 1.

```
Algorithm 1 MGP($M$,$D$)
Require: $M \geq 0 \vee D \neq null$
 1: $M$ is the maximum generation
 2: $D$ is the data set
Ensure: $MGP$
 3: Let $gen = 0$
 4: Initialize the population using the ramped-half-and-half
    method
 5: while $gen \leq M$ do
 6:     Evaluate fitness($F_E(x)$) of each individual
 7:     Update the best individual TBest
 8:     Survival Selection + Crossover
 9:     Shifting operator
10:     Splitting operator
11:     gen = gen + 1
12: end while
```

Algorithm 1 is similar as the common GP algorithms. It uses ramped-half-and-half method [14] to initialize the population. This method can ensure the diversity of the initial population. The evaluation of the individual or classifier is using our new fitness function $F_E(x)$ in Section IV. Survival selection and crossover are combined to construct the son population. Shifting operator and splitting operator are adopted on the son individuals. Every individual does deep exploiting by these two local search operators and is kept as the new parent individual in the next generation. MGP always keep the best individual in the evolving process and take the best one as the final classifier to class the test instances.

VII. Experimental Studies

A. Data Sets

Ten Data sets are selected from UCI [29] and described in Table I. In this paper, we focus on the binary classification problems, so all the data sets are 2-class problems. *Haberman*, *german*, *hepatitis*, *spambase* and *bands* are imbalanced data sets, others are balanced data sets.

TABLE I
10 UCI Data Sets

Data Set	No. of features	Class Distribution
haberman	3	81:225
german	24	700:300
hepatitis	19	32:123
monks-1	36	1669:1527
hill-valley	100	612:600
spambase	57	1813:2788
mammographics	5	445:516
bands	36	228:312
crx	15	307:383
heart	13	150:120

B. Experimental Design

To evaluate our ideas on new fitness function F_E and the splitting operator, we compare our new algorithm with three other algorithms. They are showed in Table III. All the algorithms have the same framework of GP. FGP comes from Li's work [15]. We design GGP ($G\text{-}mean$ GP) and EGP (Entropy GP) to make the proof procedure more clear and smooth. The difference between EGP (Entropy GP) and FGP or GGP is the encoding of individuals and fitness function. F_E is better than F_{cs} or F_{Gmean} when the result of EGP is better than FGP and GGP. The efficiency of the splitting operator can be proved by comparing MGP with EGP. C4.5 is the tree structure classifier, and the original idea of our fitness function and splitting operator, so it is taken in the compared list. In Table II, it provides the parameters of the four algorithms 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	10 UCI data sets
Algorithms	C4.5 ,FGP-1 , GGP , EGP, MGP
Crossover rate	0.9
Mutation rate	0.1
Parameters for GP	P(Population size) = 100; G (Maximum generation) = M Number of Runs : 5 fold crossvalidation 20 times
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

TABLE III
Four algorithms to be compared

	Encoding	Fitness function	Shifting	Splitting
FGP	GDT	F_{cs}	Yes	No
GGP	GDT	$G\text{-}mean$	Yes	No
EGP	SGDT	F_E	Yes	No
MGP	SGDT	F_E	Yes	Yes

1) Fixed Generations Experiment: To briefly evaluate the performance of MGP over the other compared algorithms, we simply run all the algorithms with the same generations (150 in our study). And every algorithm runs 5-fold cross-validation 20 times on all data sets. It should be noted that different algorithms may need different number of generation to obtain the optimal solution (i.e., classifier). By using the same number of generations for all the compared algorithms, we aim to analyze the whole evolutionary behavior of them rather than simply comparing the quality of the best classifiers achieved by each algorithm. All the final results are shown from Fig 9 to Fig. 18. The blue lines with crosses represent the performance of MGP, the green lines with circle markers stand for EGP, the cyan lines with plus markers belong to GGP and the red lines with snow markers are results obtained with FGP. From these figures, it can be found that our MGP performs significantly better than other algorithms on all the training data sets. Furthermore, it can also be observed that our MGP performs better than other algorithms on most of the test sets in two points: First, the process of converging to the obtained best solutions of MGP is much shorter than others. Second, the best AUC obtained by GP among the pre-defined generations is always the largest of all the compared algorithms.

Though MGP can get the best solutions quickly on all of the test data sets, except the *haberm* data set, it could be found that the overfitting of MGP exists on several data sets (e.g. *german*, *crx* and *haberman*). The reason of the overfitting is our splitting operator which divides the subspace into several smaller subspaces. It means that splitting operator leads to increment of subspaces and decrement of number of instances in one subspace. The overfitting comes out when there are too many decision leaf nodes with few training instances. The parameter ST in Section V is used to decide whether the splitting operator work on a decision leaf node. If ST is too strict small like 0, splitting operator will be done on all decision leaf node, that is not so good. If ST is too large like 1, splitting operator will not be done at all, and MGP equals EGP. There are three methods can be used to avoid the overfitting. The first is that we can adjust ST which controls the search depth of the splitting operator. Pruning strategy in C4.5 also can be employed for preventing the overfitting which is the second method. It is not necessary for different algorithms to run with the same generations. Generation number M is

TABLE IV
PERFORMANCE ON 10 DATA SETS I

	C4.5	EGP	FGP	GGP	MGP
bands	**0.7456 ± 0.0021**	0.7004 ± 0.0505	0.5399 ± 0.0556	0.6488 ± 0.0489	0.7412 ± 0.0553
crx	0.8551 ± 0.0016	0.9068 ± 0.0249	0.8591 ± 0.0357	0.8636 ± 0.0332	**0.9073 ± 0.0254**
german	0.6536 ± 0.0022	0.7081 ± 0.0342	0.5175 ± 0.0351	0.6714 ± 0.0536	**0.7180 ± 0.0336**
haberman	0.6396 ± 0.0043	0.6297 ± 0.0763	0.5066 ± 0.0425	**0.6398 ± 0.0668**	0.6127 ± 0.0744
heart	0.7854 ± 0.0046	**0.8220 ± 0.0584**	0.7564 ± 0.0659	0.7624 ± 0.0646	0.8178 ± 0.0540
hepatitis	0.6438 ± 0.0168	0.7285 ± 0.1093	0.5908 ± 0.1054	0.7050 ±0 0.1202	**0.7299 ± 0.1099**
hill-valley	0.5000 ± 0.0000	0.5018 ± 0.0215	0.5009 ± 0.0139	0.4990 ± 0.0325	**0.5765 ± 0.0092**
mammographic	0.8766 ± 0.0005	**0.8896 ± 0.0197**	0.8276 ± 0.0360	0.8473 ± 0.0346	0.8846 ± 0.0247
monks1	0.7522 ± 0.0048	0.8596 ± 0.1196	0.5121 ± 0.0996	0.7503 ± 0.0525	**1.0000 ± 0.0000**
spambase	0.9372 ± 0.0001	0.8393 ± 0.0508	0.7614 ± 0.0716	0.7658 ± 0.0430	**0.9543 ± 0.0063**

Fig. 9. bands result

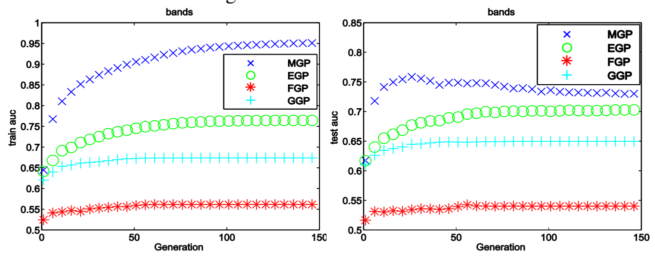

Fig. 10. crx result

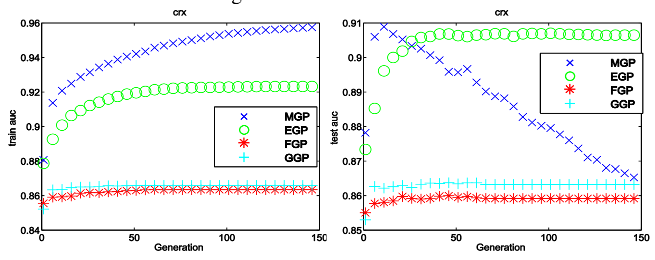

Fig. 11. german result

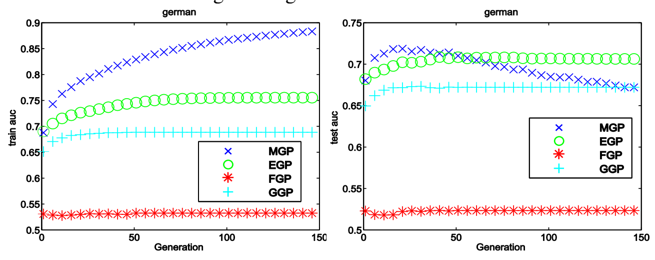

Fig. 12. haberman result

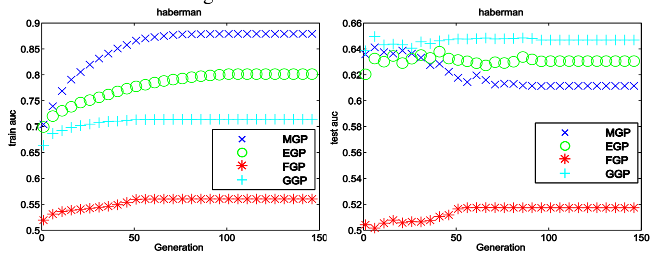

Fig. 13. heart result

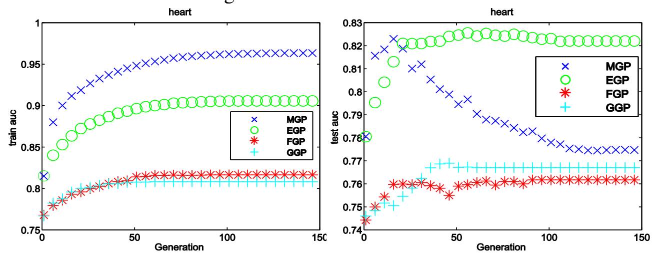

Fig. 14. hepatitis result

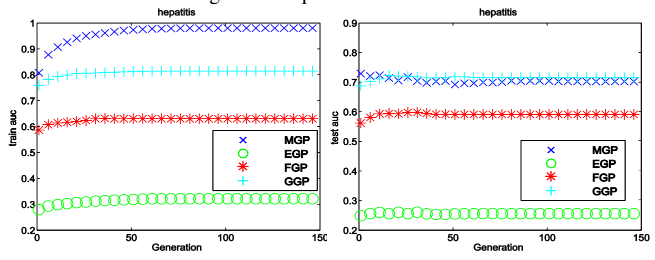

Fig. 15. hill-valley result

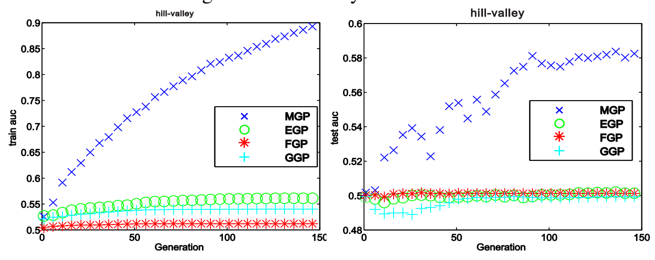

Fig. 16. mammographic result

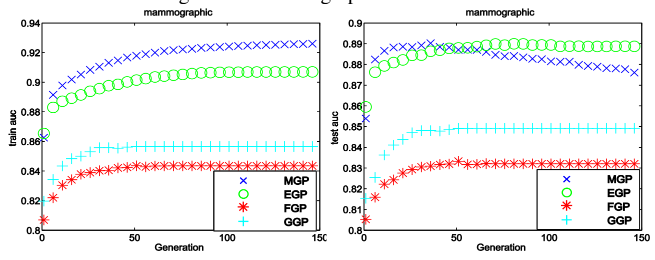

Fig. 17. monks1 result

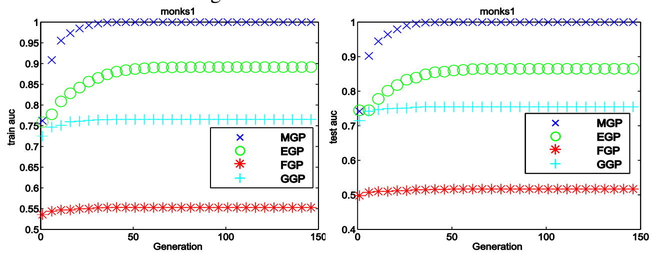

Fig. 18. spambase result

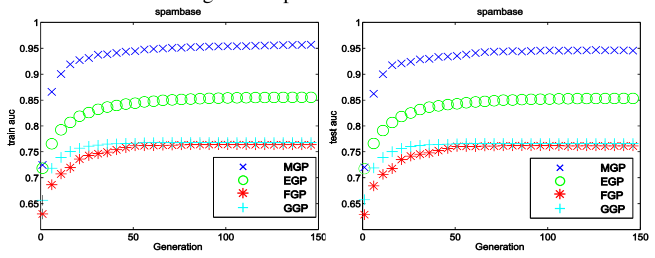

TABLE V
TERMINAL GENERATIONS FOR FOUR ALGORITHMS

	FGP	EGP	MGP	GGP
bands	51	82	78	45
crx	8	95	14	36
German	9	51	20	34
haberman	25	70	67	11
Heart	49	27	18	30
hepatitis	50	38	13	8
hill-valley	15	125	143	72
mammographic	20	93	70	38
monks1	25	49	72	21
spambase	112	134	145	112

TABLE VI
PERFORMANCES ON 10 DATA SETS II

	C4.5	EGP	FGP	GGP	MGP
C4.5		2-2-6	8-1-1	4-3-3	1-2-7
EGP			9-1-0	7-3-0	1-5-4
FGP				0-4-6	0-0-10
GGP					0-1-9
MGP					

taken as a normal parameter in our paper and adjust it to different algorithms for different data sets. This is the third strategy, actually it is adopted into MGP as our next subsection.

2) Dynamic Generations Experiment: In this experiment, M (number of generations of GP) is not taken as a fixed parameter in these algorithms. M is a normal parameter in MGP. Different M should be used to MGP to solve different data sets. There is a very common method in machine learning to adjust the parameter for classifier that can use it to choose the suitable M for MGP. 5-fold cross-validation 20 times are done on all data sets. 80% instances are stratified sampling as the new data set(SD) then do 5-fold cross-validation once on D and take parameter (M) with best result in SD as the final parameter to do the 5-fold cross-validation 20 times on the whole data set. Table V gives the terminal generations for four algorithms by above idea.

Table IV describes the AUC of five algorithms on 10 data sets. Every grid contains an average value on of 100 results and its standard deviation. The bold number is the best in the row. Table VI is the result by using wilcoxon test [30] with a significance level of 5%. A $x-y-z$ in a grid of Table VI means the algorithm in column is worse than algorithm in the row z times, better$x$ times, and there is no difference between them y times.

From Table IV, if we assign a score S to the algorithms on these data sets, score equals number of top1 divides the total number of data sets, $S(C4.5) = 0.1, S(EGP) = 0.2, S(FGP) = 0, S(GGP) = 0.1, S(MGP) = 0.6$. For these data sets, we can claim that MGP is better than EGP, EGP is better than GGP and C4.5 which are better than FGP, and MGP is the best algorithm.

In Table VI, we want to prove two things: one is the fitness function 5 is efficient and the other is that our splitting operator is good in the search strategy. The grid of $9-1-0$ with column FGP and row EGP, and the grid of $7-3-0$ with column GGP and row EGP show EGP is much better than FGP and GGP. The only difference between EGP and FGP or GGP is the fitness function (also the encoding of individual). This means that our new fitness function 5 works in the GP for classification problems. The next grid of $1-5-4$ with column MGP and row EGP means MGP loses 1 goal, wins 4 goals and draws 5 goals in 10 matches, also means MGP is better than EGP. Splitting operator is adopted in MGP but EGP and there is no difference between them except this. We can argue that splitting operator is an efficient and successful operator in our paper.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an improved version of FGP, namly MGP. In comparison to FGP, MGP is equipped with three novel components. First, SGDT is the new tree structure for GP to construct individuals for classification problem. The second is our new fitness function F_E which comes from information gain in C4.5 and the statistic information from the SGDT individual. The third is the splitting operator which also uses the splitting strategy in C4.5.

To verify the efficacy of F_E and splitting operator, EGP, FGP, GGP and MGP are compared in the experimental section. According to the results, EGP is better than FGP and GGP, which means F_E can work in the GP for classification problems. Splitting operator is also good because MGP is better than EGP.

In this paper, we found that MGP will fall into overfitting more easily than other GP-based algorithms. The reason is that we take the splitting operator which is a strong local search. The cross-validation method had been used to overcome the overfitting, and it worked well in partial data sets in the second experiment. The criteria of splitting should be an important parameter in the search process. In the future work, some pruning techniques are simple and efficacy methods which can be adopted into the MGP.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China under Grants 60802036, U0835002, and 61028009, and by the Engineering and Physical Sciences Research Council (EPSRC Grant EP/D052785/1).

References

[1] J. Han and M. Kamber, *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.

[2] P. Tan, M. Steinbach, V. Kumar *et al.*, *Introduction to data mining*. Pearson Addison Wesley Boston, 2006.

[3] P. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 2, pp. 121–144, 2010.

[4] C. Bishop, *Neural networks for pattern recognition*. Oxford University Press, USA, 1995.

[5] B. Ripley, *Pattern recognition and neural networks*. Cambridge Univ Pr, 2008.

[6] J. Holland, L. Booker, M. Colombetti, M. Dorigo, D. Goldberg, S. Forrest, R. Riolo, R. Smith, P. Lanzi, W. Stolzmann *et al.*, "What is a learning classifier system?" *Learning Classifier Systems*, pp. 3–32, 2000.

[7] N. Cristianini and J. Shawe-Taylor, *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge Univ Press, 2000.

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

[9] F. Provost and P. Domingos, "Tree induction for probability-based ranking," *Machine Learning*, vol. 52, no. 3, pp. 199–215, 2003.

[10] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *Evolutionary Computation, IEEE Transactions on*, vol. 4, no. 4, pp. 380–387, 2002.

[11] W. Au, K. Chan, and X. Yao, "A novel evolutionary data mining algorithm with applications to churn prediction," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 6, pp. 532–545, 2004.

[12] P. Wang, E. P. Tsang, T. Weise, K. Tang, and X. Yao, "Using gp to evolve decision rules for classification in financial data sets," *Special Session on Evolutionary Computing of the 9th IEEE International Conference on Cognitive Informatics (ICCI 2010)*, pp. 722–727, Jul. 9 2010.

[13] A. Freitas, "A survey of evolutionary algorithms for data mining and knowledge discovery," *Advances in Evolutionary Computation*, pp. 819–845, 2002.

[14] J. Koza, *Genetic programming: on the programming of computers by means of natural selection*. The MIT press, 1992.

[15] L. Jin, "FGP: A genetic programming based financial forecasting tool," 2000.

[16] E. Tsang, J. Li, and J. Butler, "EDDIE beats the bookies," *Software: Practice and Experience*, vol. 28, no. 10, pp. 1033–1043, 1998.

[17] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *Information Theory, IEEE Transactions on*, vol. 14, no. 1, pp. 55–63, 2002.

[18] 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.

[19] Y. Ong, M. Lim, N. Zhu, and K. Wong, "Classification of adaptive memetic algorithms: A comparative study," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 36, no. 1, pp. 141–152, 2006.

[20] E. Tsang, P. Yung, and J. Li, "EDDIE-Automation, a decision support tool for financial forecasting," *Decision Support Systems*, vol. 37, no. 4, pp. 559–565, 2004.

[21] A. Freitas, *Data mining and knowledge discovery with evolutionary algorithms*. Springer Verlag, 2002.

[22] A. Garcıa-Almanza, E. Tsang, and E. Galván-López, "Evolving Decision Rules to Discover Patterns in Financial Data Sets," *Computational methods in financial engineering: essays in honour of Manfred Gilli*, p. 239, 2008.

[23] E. Tsang, J. Li, S. Markose, H. Er, A. Salhi, and G. Iori, "EDDIE in financial decision making," *Journal of Management and Economics*, vol. 4, no. 4, 2000.

[24] T. Fawcett, "An introduction to ROC analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[25] Y. Ong, P. Nair, A. Keane, and K. Wong, "Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems," *Knowledge Incorporation in Evolutionary Computation*, pp. 307–332, 2004.

[26] M. Lim, Y. Yuan, and S. Omatu, "Extensive testing of a hybrid genetic algorithm for solving quadratic assignment problems," *Computational Optimization and Applications*, vol. 23, no. 1, pp. 47–64, 2002.

[27] N. Jardine and C. van Rijsbergen, "The use of hierarchic clustering in information retrieval," *Information storage and retrieval*, vol. 7, no. 5, pp. 217–240, 1971.

[28] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: one-sided selection," in *Proc. of 14th Intl. Conf. on Machine Learning*. Citeseer, 1997, pp. 179–186.

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

[30] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.