Learning Ensemble of Decision Trees through Multifactorial Genetic Programming

Yu-Wei Wen and Chuan-Kang Ting
Department of Computer Science and Information Engineering and
Advanced Institute of Manufacturing with High-tech Innovations
National Chung Cheng University
Chia-Yi 621, Taiwan
Email: {wyw100u, ckting}@cs.ccu.edu.tw

***Abstract*—Genetic programming (GP) has received considerable successes in machine learning tasks such as prediction and classification. Ensemble learning enables the collaboration of multiple classifiers and effectively improves the classification accuracy. Learning an ensemble of classifiers with GP can simply be achieved by repeated runs of GP; however, the computational cost will be multiplied as well. Recently, multifactorial evolution was proposed to concurrently solve multiple problems with a single population. This study utilizes the multifactorial evolution and designs a multifactorial genetic programming (MFGP) for efficiently learning an ensemble of decision trees. In the MFGP, each task is associated with one run of GP. The multifactorial evolution enables MFGP to evolve multiple GP classifiers for an ensemble in a single run, which saves a substantial amount of computational cost at repeated runs of GP. The experimental results show that MFGP can learn an ensemble with comparable accuracy, precision, and recall to conventional ensemble learning methods, whereas MFGP requires much less computational resource. The satisfactory outcomes validate the advantages of MFGP in ensemble learning.**

***Index Terms*—Multifactorial evolution, ensemble learning, genetic programming, decision tree.**

I. Introduction

Decision tree is a well-known machine learning technique and has successfully solved many classification and prediction problems, e.g., stock market trend prediction, medical diagnosis, and rainfall forecasting [1–6]. In a real-valued decision tree, each node comprises a condition associated with a specific feature, a comparison operator, and a threshold value. Entropy and Gini index are two common measures for selection of nodes in a decision tree. The process of building a decision tree is usually greedy, which suffers from being trapped in local optima. Inspired by natural evolution, evolutionary algorithms (EAs) manipulate a population of candidate solutions to search for the optimal solutions. Several dialects of EAs, e.g., genetic algorithm, evolution strategy, and genetic programming, have been designed and successfully solved various search and optimization problems. In particular, genetic programming (GP) [7] has shown to be effective in seeking the optimal decision tree by using evolutionary operators such as selection, crossover, and mutation. GP has achieved a considerable amount of results better than conventional decision tree methods [8–12].

Beyond a single classifier, ensemble learning uses a number of classifiers to improve the classification accuracy. Ensembles of diverse classifiers generally exert positive effects on accuracy and reduce the overfitting situation. To construct an ensemble, the methods such as boosting, bagging, feature reduction, and data sampling are used to segment data for training. The learning algorithms are then applied on these data subsets to generate multiple classifiers, from which the final decision can be made by majority voting or weighted voting.

As for GP, the classifiers of an ensemble can be obtained by repeated runs of GP in that its stochastic nature leads to convergence to different niches (corresponding to different classifiers) over the search space [8–10, 13, 14]. Brameier and Banzhaf [15] compared several fusion methods for a set of classifiers evolved by GP. Zhang and Bhattacharyya [16] applied multiple GP models on data subsets for classification among large-scale data. The repeated runs of GP, however, substantially increase the computational cost at ensemble learning.

Recently, Gupta et al. [17] proposed the multifactorial optimization (MFO), which aims to solve *multiple* problems concurrently with a single population. The multifactorial evolutionary algorithm uses a real-coded chromosome representation to unify the representations for numerical and combinatorial optimization. Accordingly, a chromosome can simultaneously represent the candidate solutions for different types of problems. The MFO greatly increases the efficiency of evolutionary algorithms in dealing with more than one problem; in addition, it can achieve better solution quality than solving a single problem.

In view of these advantages, this study proposes the multifactorial genetic programming (MFGP), which introduces multifactorial evolution to GP for ensemble learning. The MFGP evolves multiple GP classifiers concurrently with one single population to save the computational cost of ensemble learning. That is to say, the MFGP consumes a similar amount of computation to standalone GP but generates multiple classifiers for the ensemble at once.

The remainder of this paper is organized as follows. Section 2 reviews the related work. The proposed MFGP is elucidated in Section 3. Section 4 presents the experimental results. Finally we draw conclusions of this work in Section 5.

978-1-5090-0623-6/16/$31.00

Algorithm 1 Genetic Programming (GP)

```
Initialize the population
Evaluate individuals
repeat
    repeat
        Select parents from the population
        Recombine parents to generate offspring
        Evaluate offspring
        Add offspring to the subpopulation
    until Subpopulation is filled
    Select the individuals to survive
until Termination criterion is satisfied
```

II. Related Work

Decision tree can generally be learned by a top-down splitting mechanism based on entropy or Gini index. It can also be learned by a global search algorithm such as GP. Training decision tree with information gain or Gini index is done by a top-down process that computes and compares entropy or classification error to determine a node. This learning process is usually greedy and may lead to missing the global optimal decision tree. The top-down tree building methods ordinarily have lower cost than GP; however, decision trees trained by GP have higher classification accuracy [11].

The ensemble technique has shown to be effective to improve the classification accuracy of GP-based decision trees [13, 14, 18]. Nonetheless, building an ensemble of decision trees requires several times of training, which induces higher computational cost than building a single decision tree. For example, an ensemble of five GP-based decision trees requires 5 runs of GP, while each GP evolution is a computationally expensive process. Parallel GP is able to efficiently learn an ensemble of classifiers on distributed computation environments [8, 9]. However, the total computational cost remains unchanged.

Multifactorial evolution facilitates evolutionary multitasking, which can simultaneously solve multiple optimization tasks with one population in a single evolution. Gupta et al. [17] proposed the multifactorial evolutionary algorithm (MFEA) to realize the concept of evolutionary multitasking. With a user-defined chromosome mapping method, MFEA can solve multiple tasks concurrently and obtain even better solution quality than a single task does.

III. Multifactorial Genetic Programming

This study presents the MFGP aiming to efficiently learn an ensemble of GP classifiers. Evolutionary multitasking allows MFGP to learn an ensemble with much less computational resource than traditional GP requires. This section introduces GP for classification and then describes the proposed MFGP in detail.

A. Genetic Programming for Classification

Genetic Programming uses genetic operators, i.e., selection, crossover, and mutation, to evolve tree-represented chromosomes for optimal performance on given tasks. Algorithm 1 shows the pseudocode of GP. Compared to other EAs, GP is featured in the use of trees for representation. For example, GP uses parse tree and decision tree to represent candidate solutions, whereas genetic algorithm uses strings (e.g., Boolean string, integer string, and real-valued vector).

In the light of tree structure, one-point crossover is widely used in GP. This crossover operator randomly selects a subtree in each parent and then exchanges the two subtrees. Figure 1 illustrates the crossover operation in GP. Mutation is another genetic operator used in GP. The mutation operator randomly chooses a subtree and replaces it with a random one (cf. Fig. 2). In GP, crossover is viewed as a major operator, whereas mutation is a minor operator and seldom performed.

For classification, the fitness function of GP can be defined

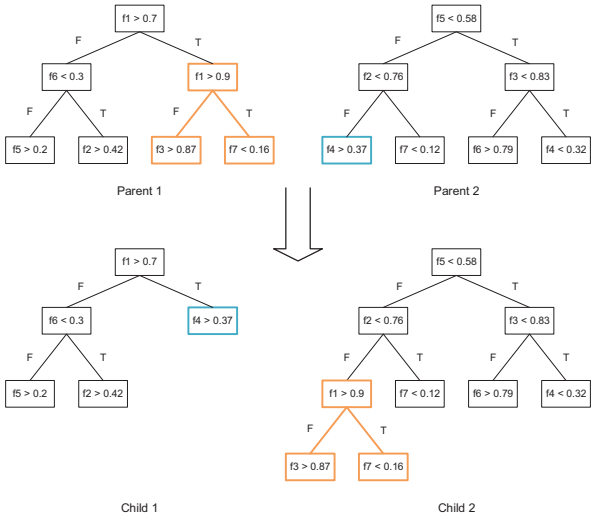


Figure 1. Crossover in GP

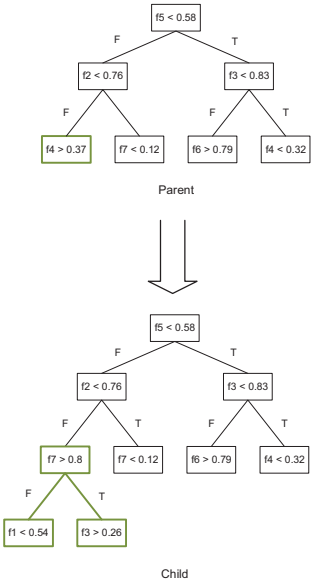


Figure 2. Mutation in GP

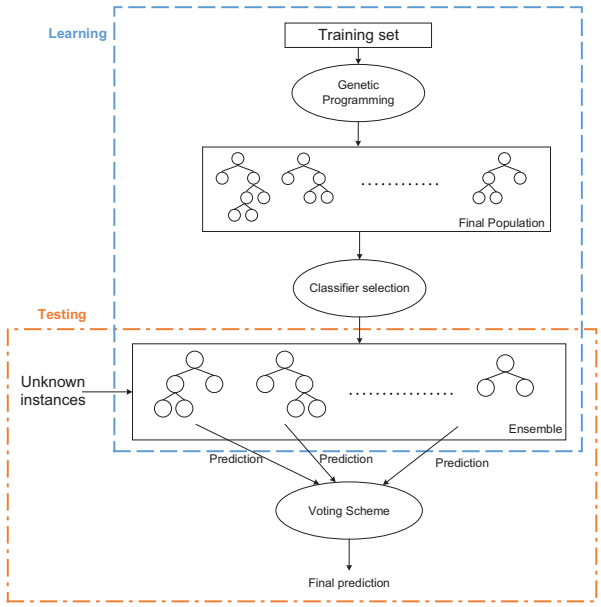


Figure 3. Flowchart of GPEn1

by the training accuracy. The fitness of individual $\boldsymbol{x}$ is calculated by

$$f(\boldsymbol{x}) = \frac{\text{number of instances correctly classified by } \boldsymbol{x}}{\text{number of instances in the training set}}. \quad (1)$$

A node of real-valued decision tree indicates a condition and the paths to other nodes. The condition comprises a feature, a comparator, and a threshold value. To use GP to find the optimal decision tree, the candidate trees are represented as individuals. Then GP applies the evolutionary scheme and genetic operators to evolve the individuals toward the optimal decision tree [11, 14].

Ensemble is promising for enhancement of GP performance. The members of an ensemble can be obtained by applying selection strategies on the final population of GP. Selection of decision trees is similar to setting weights for each decision tree: Both are associated with the importance of each decision tree in the ensemble. The present study denotes this ensemble construction approach as GPEn1. Specifically, GPEn1 selects the n decision trees with highest prediction accuracy to construct the ensemble, where n represents the ensemble size. Figure 3 shows the process of GPEn1.

Another approach uses repeated runs of GP to construct the ensemble. For example, feature selection and data sampling, such as random forest, can be used to segment the original data into several subsets; then GP is applied to train decision trees on each data subset. The resultant classifiers from all data subsets can then form an ensemble. Without data segmentation, repeated runs of GP on the same training data can also result in different decision trees due to the stochastic nature of GP. In this way, an ensemble is formed by the best decision trees obtained from multiple runs of GP. The method of running GP repetitively to construct the ensemble is denoted as GPEn2. Figure 4 illustrates the process of GPEn2.

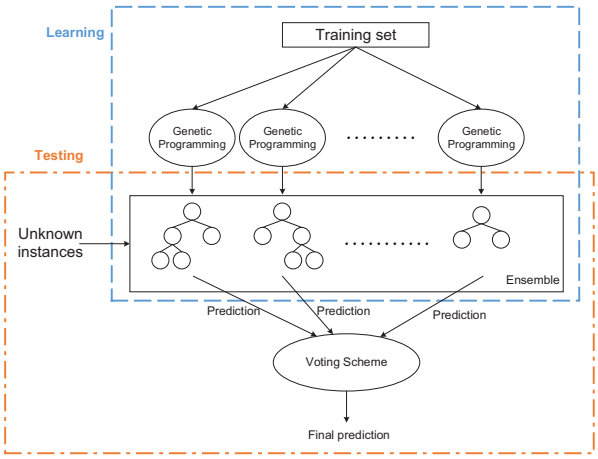


Figure 4. Flowchart of GPEn2

B. Multifactorial Genetic Programming

Multifactorial evolutionary algorithm (MFEA) is capable of solving multiple optimization problems simultaneously. This study proposes a novel MFEA, i.e., the MFGP, to concurrently generate multiple classifiers for ensemble learning. In MFEA, a problem to be solved is regarded as a task. Following this notion, the MFGP formulates the learning of classification model as an optimization problem of maximizing the accuracy. Learning a decision tree is thus regarded as a task in the MFGP.

To learn an ensemble of K decision trees, the MFGP initializes K tasks and evolves them simultaneously toward the optima. The objective function of the j-th task T_j is defined by $f_j : X_j \to \mathbb{R}$, where X_j denotes the search space of T_j. In addition to fitness values, MFEA defines four measures for evaluation of individuals with respect to different tasks.

Definition 1 (Factorial Cost)**.** The factorial cost ψ_j^i of an individual p_i on task T_j is defined by

$$\psi_j^i = f_j^i - \delta_j^i, \quad (2)$$

where f_j^i denotes the prediction accuracy of p_i on T_j, and δ_j^i is the penalty for over-complex individual.

Note that the factorial cost in MFGP is defined to be maximized, which is different from the original definition in [17].

Definition 2 (Factorial Rank)**.** The factorial rank r_j^i is defined as the performance rank associated with factorial cost ψ_j^i.

For a pair of individuals having the same factorial cost on T_j, random tie-breaking is used to distinguish the two individuals.

Definition 3 (Scalar Fitness)**.** The scalar fitness φ_i of individual p_i is defined by the inversion of the best rank over all tasks:

$$\varphi_i = \frac{1}{\min_{j\in\{1,\ldots,K\}} r_j^i}. \quad (3)$$

Algorithm 2 Multifactorial Genetic Programming (MFGP)

```
Initialize the population
Evaluate individuals on all tasks
Compute factorial rank and skill factor
repeat
    repeat
        Randomly select parents from the population
        Perform assortative mating          ▷ Algorithm 3
        Vertical cultural transmission      ▷ Algorithm 4
        Evaluate factorial cost on the task of skill factor
        Add offspring to the subpopulation
    until Subpopulation is filled
    Update factorial rank, scalar fitness, and skill factor
    Select the individuals to survive
until Termination criterion is satisfied
```

Definition 4 (Skill Factor)**.** The skill factor τ_i is the task that p_i performs best among all tasks. Formally, the skill factor of p_i is defined by

$$\tau_i = \underset{j\in\{1,\ldots,K\}}{\operatorname{argmin}} r_j^i . \tag{4}$$

The above measures are used to evaluate individuals considering multiple optimization problems. Note that the measure values need to be updated in each generation.

According to the principle of natural evolution and biocultural model, the MFGP applies assortative mating, recombination (crossover or clone), vertical cultural transmission, and survival selection, to evolve candidate solutions into the optima for all given tasks. In the beginning, MFGP generates the initial population randomly, in which every individual is evaluated on all tasks and accordingly its factorial rank and skill factor are computed. Then MFGP conducts the evolutionary process: In each generation, random parent selection is applied and followed by assortative mating, recombination, and vertical cultural transmission to generate offspring. In the reproduction process of MFGP, if the two randomly selected parents have the same skill factor, the assortative mating performs crossover to produce offspring, which will inherit the skill factor from their parents. Otherwise, crossover is performed with a predetermined probability and the offspring will inherit skill factor from one of its parents. Afterward, offspring are evaluated according to their own skill factor. The $(\mu + \lambda)$ survival selection is adopted with respect to the scalar fitness. Algorithms 2–4 present the framework, assortative mating, and vertical cultural transmission of MFGP, respectively.

In MFEA, the population is implicitly partitioned into K sub-populations by the skill factor and assortative mating. To diversify the decision trees obtained from the K subpopulations, the MFGP further scrambles the dataset for each task by randomly mapping the feature indexes. For example, Table I shows that, for Task 1, its feature number one corresponds to the fourth feature (feature D) in the dataset, number two corresponds to the first feature (feature A), and so on. The three tasks will then work on the datasets with different feature sequences, helping to generate diverse decision trees. Figure 5 presents the flowchart of MFGP.

Algorithm 3 Assortative mating

For two randomly selected parents p_a and p_b:

```
if τ_a == τ_b then
    Recombine p_a and p_b to generate c_a and c_b
else if Mating is probabilistically enabled then
    Recombine p_a and p_b to generate c_a and c_b
else
    Clone p_a and p_b to generate c_a and c_b, respectively
end if
```

Algorithm 4 Vertical cultural transmission

Child c_a has either one parent p_a or two parents p_a and p_b:

```
if c_a has two parents then
    c_a inherits skill factor randomly from p_a or p_b
else
    c_a inherits skill factor from p_a
end if
```

IV. Experiment Results

This study conducts several experiments to examine the performance of MFGP for ensemble learning in the classification problems. The test datasets include Australian Credit Data (ACD), German Credit Data (GCD), Liver Disorders Data Set (LD), and Pima Indians Diabetes Data Set (PID) from the UCI Machine Learning Repository [19]. Table II summarizes the properties of these four test datasets. In the experiments, all data values are normalized to $[0, 1]$. The performance metrics, to wit, accuracy, precision, and recall, are obtained from 10-fold cross-validation and each fold includes 10 trials of test algorithms.

This study uses three decision trees for an ensemble in the experiments. The maximum depth of decision tree is set to 14 in order to prevent GP from bloating. The test algorithms include GP, GPEn1, GPEn2, and MFGP. For a classification problem, GP uses the resultant best individual (a single decision tree) as the classifier, GPEn1 adopts the best three individuals of a single run to form an ensemble, GPEn2 conducts GP three times for the ensemble consisting of three best individuals, and MFGP enables concurrent evolution of three GP to form the ensemble through a single run. Table III lists the parameter setting for the four test algorithms in the experiments.

Table I
An example of random mapping of feature indexes for different tasks in the dataset

	A	B	C	D	E
Task 1	2	4	5	1	3
Task 2	3	2	4	1	5
Task 3	3	1	5	2	4

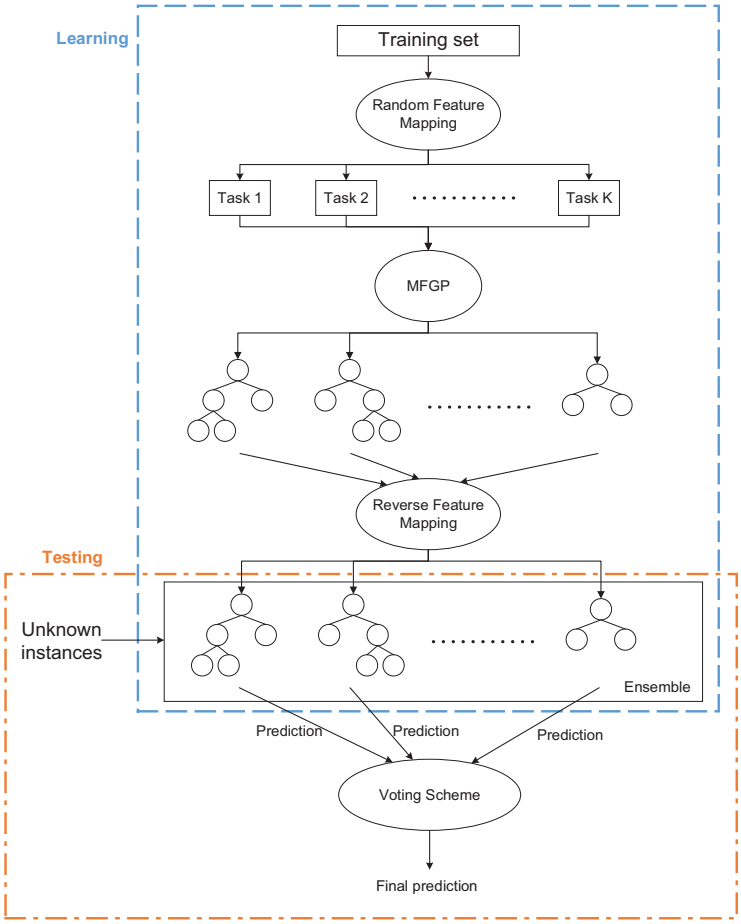


Figure 5. Flowchart of MFGP

Table II
PROPERTY OF DATASETS

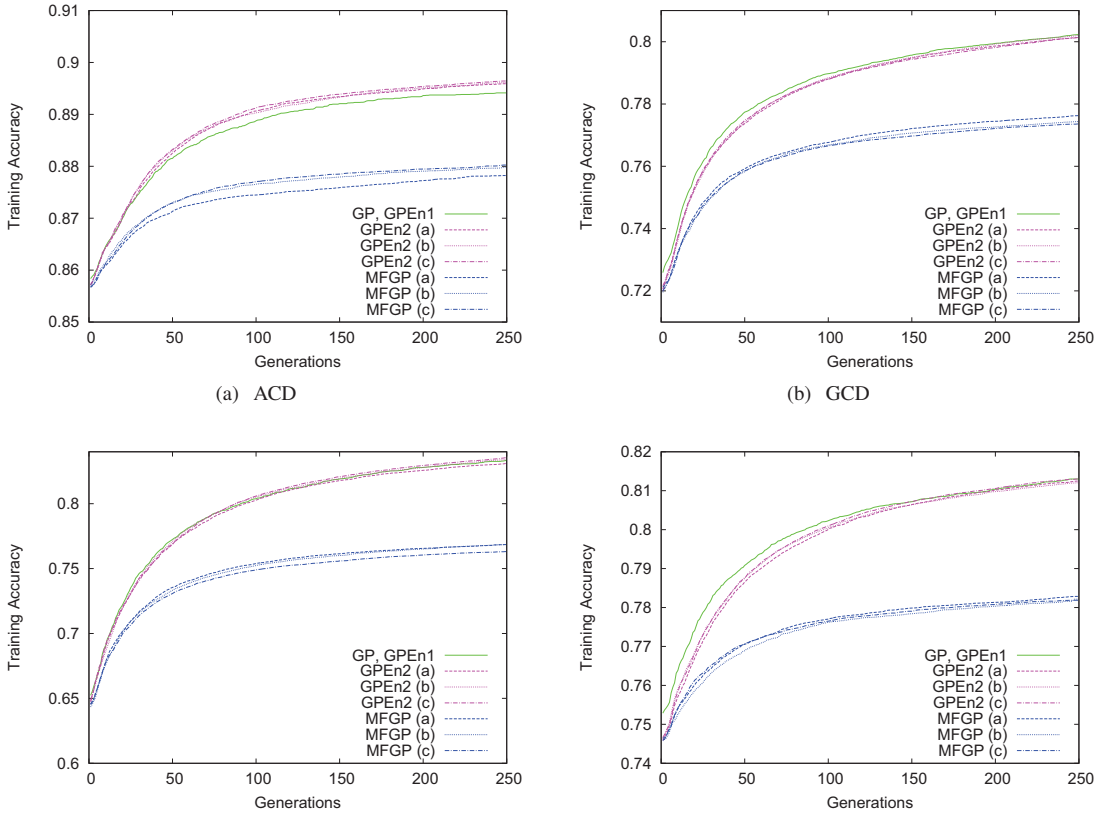
Dataset	Volume	#Features	#Labels	#Label1	#Label2
ACD	690	14	2	387	303
GCD	1000	20	2	700	300
LD	345	6	2	200	145
PID	768	8	2	500	268

Table III
PARAMETER SETTING

Parameters	GP	GPEn1	GPEn2	MFGP
Population size	2000			
Parent selection	Random selection			
Crossover	One-point crossover with $p_c = 1.0$			
Survival selection	$(\mu + \lambda)$			
Selection criterion	Fitness	Fitness	Fitness	Scalar
#Generations	250			
Voting scheme of ensemble	–	Majority	Majority	Majority
#Decision trees in an ensemble	1	3	3	3
#Runs of evolution	1	1	3	1
Cross-validation	10-fold cross-validation			

Figure 6 shows the convergence of training accuracy against generations for GP, GPEn1, GPEn2, and MFGP, where the indexes (a), (b), and (c) of GPEn2 denote the three runs of GP, and those of MFGP denote the three tasks. The results show that GP, GPEn1, and GPEn2 have similar convergence speed, which is reasonable since they carry out separate runs of GP. By contrast, MFGP converges slower and has lower training accuracy than GP, GPEn1, and GPEn2 due to the effect of multitasking in MFGP.

Next, we investigate the classification accuracy of the four test algorithms on the datasets. Although MFGP has slower convergence and lower training accuracy than the other three algorithms, Table IV and Fig. 7 show the advantages of MFGP in test accuracy, which is the key indicator of classification performance. The results show that MFGP and GPEn2 both outperform GP and GPEn1 in terms of test accuracy on all datasets. The test accuracy of MFGP and GPEn2 is very close; nonetheless, MFGP requires only $1/3$ number of fitness evaluations of GPEn2. Through multifactorial evolution, MFGP gains comparable classification accuracy but requires much lower computational cost than GPEn2 does. Moreover, MFGP achieves higher accuracy than GP and GPEn1 do under a similar number of fitness evaluations.



(a) ACD (b) GCD

(c) LD (d) PID

Figure 6. Variation of training accuracy of GP, GPEn1, GPEn2, and MFGP against generations

Considering the fact that accuracy may be misleading if the data set is unbalanced, we further examine the precision and recall of the four algorithms. Precision measures the number of correctly classified positive instances against the number of instances classified as positive. High precision is desirable for reducing the number of negative instances that are incorrectly predicted as positive instances. On the other hand, recall measures the number of correctly classified positive instances against the number of positive instances. High recall is desirable for reducing the number of positive examples that are incorrectly predicted as negative examples. According to Table IV and Fig. 7, MFGP and GPEn2 achieve higher precision and recall than GP and GPEn1, except GP gains highest recall on ACD. This tendency is similar with the results of test accuracy. Noteworthily, MFGP constructs the ensemble and attains the solution quality (i.e., accuracy, precision, and recall) comparable to GPEn2 but requires only the computational cost of a single run like GP and GPEn1. These satisfactory outcomes show the great advantages of MFGP in ensemble learning.

V. Conclusions

Multifactorial evolution is a powerful mechanism enabling EAs to solve multiple problems at the same time. This paper introduces multifactorial evolution to GP as the MFGP to reduce the computational cost at ensemble learning. In the MFGP, each task is associated with one run of GP. Through multifactorial evolution, the MFGP can achieve multiple GP classifiers to form the ensemble in a single run.

This study carries out experiments to examine the effectiveness of the proposed MFGP. The experimental results show that MFGP achieves classification accuracy, precision, and recall comparable to the ensemble generated by multiple runs of GP, and yet the MFGP requires only the computational cost of a single run. These satisfactory outcomes show the advantages of MFGP in ensemble learning. They also validate the utility of multifactorial evolution.

For future work, it is promising to apply MFGP to other ensemble techniques such as bootstrap aggregating, dimension reduction, and data sampling. Advanced design of MFGP operators is also helpful to improve its performance.

Acknowledgment

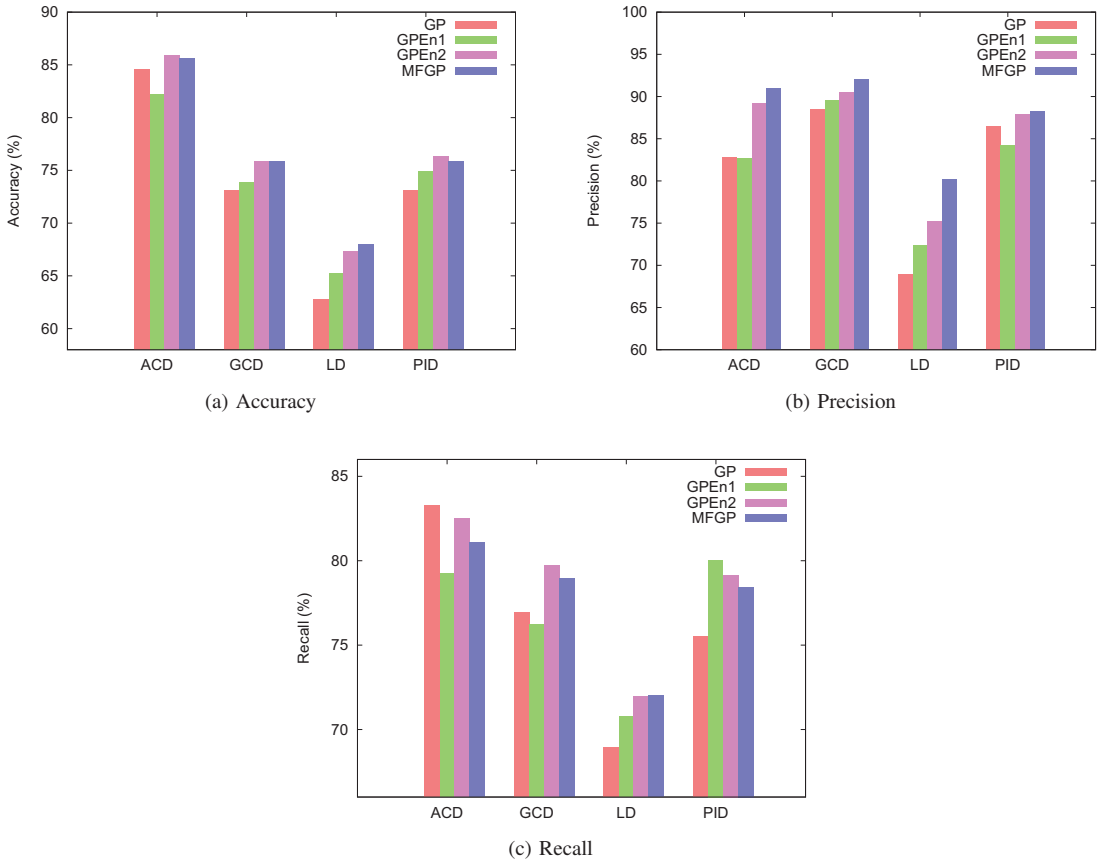
This work was supported by the Ministry of Education of Taiwan.

References

[1] M. Singh, P. Singh, and H. Singh, "Decision tree classifier for human protein function prediction," in *Pro-*

Table IV
ACCURACY, PRECISION, RECALL, AND REQUIRED NUMBER OF FITNESS EVALUATIONS FOR GP, GPEN1, GPEN2, AND MFGP ON THE FOUR DATASETS. BOLDFACE INDICATES THE BEST RESULTS AMONG THE FOUR TEST ALGORITHM.

Dataset	Algorithm	Accuracy	Precision	Recall	#Evaluations ($\times 10^3$)
ACD	GP	84.54	82.82	**83.29**	502
	GPEn1	82.19	82.68	79.22	502
	GPEn2	**85.86**	89.16	82.51	1506
	MFGP	85.59	**91.01**	81.06	506
GCD	GP	73.13	88.52	76.94	502
	GPEn1	73.09	89.56	76.22	502
	GPEn2	**75.87**	90.47	**79.73**	1506
	MFGP	75.86	**92.07**	78.93	506
LD	GP	62.79	68.96	68.92	502
	GPEn1	65.21	72.42	70.79	502
	GPEn2	67.35	75.21	71.96	1506
	MFGP	**68.00**	**80.24**	**72.01**	506
PID	GP	73.12	86.42	75.53	502
	GPEn1	74.95	84.26	**80.04**	502
	GPEn2	**76.38**	87.90	79.15	1506
	MFGP	75.87	**88.28**	78.44	506



(a) Accuracy

(b) Precision

(c) Recall

Figure 7. Comparison of accuracy, precision, and recall obtained from GP, GPEn1, GPEn2, and MFGP on ACD, GCD, LD, and PID datasets

ceedings of the International Conference on Advanced Computing and Communications, 2006, pp. 564–568.

[2] B. Nair, N. Dharini, and V. Mohandas, "A stock market trend prediction system using a hybrid decision tree-neuro-fuzzy system," in *Proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing*, 2010, pp. 381–385.

[3] N. Prasad, P. Kumar, and M. Naidu, "An approach to prediction of precipitation using gini index in sliq decision tree," in *Proceedings of the 4th International Conference on Intelligent Systems Modelling Simulation*, 2013, pp. 56–60.

[4] D. AL-Dlaeen and A. Alashqur, "Using decision tree classification to assist in the prediction of alzheimer's disease," in *Proceedings of the 6th International Conference on Computer Science and Information Technology*, 2014, pp. 122–126.

[5] A. Geetha and G. Nasira, "Data mining for meteorological applications: Decision trees for modeling rainfall prediction," in *Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research*, 2014, pp. 1–4.

[6] M. Kadhem and A. Zeki, "Prediction of urinary system disease diagnosis: A comparative study of three decision tree algorithms," in *Proceedings of the International Conference on Computer Assisted System in Health*, 2014, pp. 58–61.

[7] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[8] G. Folino, C. Pizzuti, and G. Spezzano, "A cellular genetic programming approach to classification." in *Proceedings of the Genetic and Evolutionary Computation Conference*, 1999, pp. 1015–1020.

[9] ——, "Parallel genetic programming for decision tree induction," in *Proceedings of the 13th International Conference on Tools with Artificial Intelligence*, 2001, pp. 129–135.

[10] ——, "Improving induction decision trees with parallel genetic programming," in *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, 2002, pp. 181–187.

[11] T. Khoshgoftaar, N. Seliya, and Y. Liu, "Genetic programming-based decision trees for software quality classification," in *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, 2003, pp. 374–383.

[12] A. Shali, M. Kangavari, and B. Bina, "Using genetic programming for the induction of oblique decision trees," in *Proceedings of the Sixth International Conference on Machine Learning and Applications*, 2007, pp. 38–43.

[13] M. Keijzer and V. Babovic, "Genetic programming, ensemble methods and the bias/variance tradeoff — introductory investigations," in *Proceedings of the European Conference on Genetic Programming*, 2000, pp. 76–90.

[14] Z. Fan, Y. Zuo, D. Jiang, and X. Cai, "Prediction of acute hypotensive episodes using random forest based on genetic programming," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2015, pp. 688–694.

[15] M. Brameier and W. Banzhaf, "Evolving teams of predictors with linear genetic programming," *Genetic Programming and Evolvable Machines*, vol. 2, pp. 381–407, 2001.

[16] Y. Zhang and S. Bhattacharyya, "Genetic programming in classifying large-scale data: An ensemble method," *Information Sciences*, vol. 163, pp. 85–101, 2004.

[17] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2015.

[18] K.-H. Liu, M. Tong, S.-T. Xie, and V. T. Y. Ng, "Genetic programming based ensemble system for microarray data classification," *Computational and Mathematical Methods in Medicine*, 2015.

[19] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml