Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 40 (2007) 2211–2225
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

Designing a classifier by a layered multi-population genetic programming approach

Jung-Yi Lin[a,*], Hao-Ren Ke[b], Been-Chian Chien[c], Wei-Pang Yang[a,d]

[a]*Department of Computer Science, National Chiao Tung University, Taiwan*
[b]*Library and Institute of Information Management, National Chiao Tung University, Taiwan*
[c]*Department of Computer Science and Information Engineering, National University of Tainan, Taiwan*
[d]*Department of Information Management, National Dong Hwa University, Tawian*

Received 18 April 2006; received in revised form 12 November 2006; accepted 1 January 2007

Abstract

This paper proposes a method called layered genetic programming (LAGEP) to construct a classifier based on multi-population genetic programming (MGP). LAGEP employs layer architecture to arrange multiple populations. A layer is composed of a number of populations. The results of populations are discriminant functions. These functions transform the training set to construct a new training set. The successive layer uses the new training set to obtain better discriminant functions. Moreover, because the functions generated by each layer will be composed to a long discriminant function, which is the result of LAGEP, every layer can evolve with short individuals. For each population, we propose an adaptive mutation rate tuning method to increase the mutation rate based on fitness values and remaining generations. Several experiments are conducted with different settings of LAGEP and several real-world medical problems. Experiment results show that LAGEP achieves comparable accuracy to single population GP in much less time.

Keywords: Classification; Evolutionary computation; Multi-population genetic programming

1. Introduction

Genetic programming (GP) [1], an important evolutionary computation (EC) technique, has developed rapidly in recent years. Researchers have proposed creative ideas to improve the effectiveness and efficiency of GP, such as new fitness functions, new architectures, and new individual expressions.

Traditionally, GP works with a single population. Multi-population GP (MGP) [3,18], which employs several populations to discover optimal solutions, has been proposed and developed. Many different topologies of MGP have been proposed, such as the circle topology and the random topology. Fig. 1 shows the circle topology where circles stand for populations [3]. An important characteristic of MGP is *migration*. This means that individuals can be transmitted from one population to another. The arrows in Fig. 1 indicate the migration direction. Fernández et al. [18] performed several experiments with parallel and distributed GP (PADGP), isolated multi-population GP (IMGP), where "isolated" means that there is no migration between populations, and traditional single population GP. Their experiments show that PADGP and IMGP usually obtain better performance than traditional single population GP.

Many classifiers have been developed based on GP in recent years [2–13,19,21]. To generate classification rules, Freitas [6] proposed the tuple-set-descriptor (TSD), a logical formula to represent an individual. Kotani and Sherrah [9,13] used GP to perform feature selection before using other classification methods. Multi-category classification problems are more difficult than two-class classification problems. Kishore et al. [7] and the present authors [4] have considered such a problem as multiple two-class classification problems and then generated corresponding expressions or discriminant functions.

* Corresponding author. Tel.: +886 3 5712121x56647.
E-mail addresses: jylin@ieee.org (J.-Y. Lin), claven@lib.nctu.edu.tw (H.-R. Ke), bcchien@mail.nutn.edu.tw (B.-C. Chien), wpyang@cis.nctu.edu.tw (W.-P. Yang).

0031-3203/$30.00
doi:10.1016/j.patcog.2007.01.003

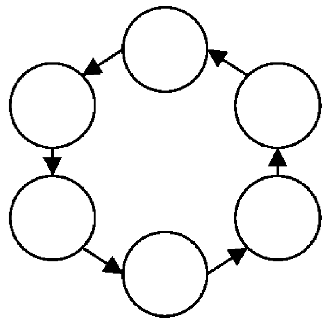

Fig. 1. An example of circle topology MGP. A circle is a population and a arrow is a migration direction.

These methods need k runs for a k-class classification problem. Muni et al. [12] proposed a novel method to solve k-class classification problems in a single run. Each individual in their work is represented by a multi-tree. Evolving one individual is equivalent to evolving k trees simultaneously. Loveard and Ciesielski [11] proposed five methods for solving multi-category classification problems including binary decomposition, static range selection, dynamic range selection, class enumeration, and evidence enumeration. Brameier and Banzhaf [3] used linear GP and MGP techniques. Individuals are represented as strings and can be transmitted between demes, i.e. subpopulations, according to their fitness value. Tsakonas [21] compares four different structures evolved by GP in several different classification problems.

Using functional expressions to represent individuals is effective in GP [4,7,10]. The tree structure is a common data structure for functional expressions. However, two problems occur when GP is employed to generate functional expressions. First, it is difficult to choose appropriate operations for a given problem because characteristics of the problem are completely unknown. If the operator set contains many operations, there is a greater possibility of discovering optimal solutions, but the searching space becomes larger and therefore may become impracticable. Fortunately, as shown in Ref. [7], GP with an operation set comprising only basic arithmetic operations, i.e. $\{+, -, \times, \div\}$, generates results comparable to that with an operation set comprising additional operations. Second, it is difficult to know the proper length of an individual because there is no prior knowledge about optimal solutions. The predefined individual length, as the length of a string-expression individual or the number of available nodes of a tree-expression individual, is usually chosen according to heuristic or empirical assumptions. The following is an example of a classification problem containing 64 dimensional data, i.e. a training instance x is represented by $x = (a_1, a_2, \ldots, a_{64})$. Suppose that an optimal solution F is known as $F = \prod_{i=1}^{64} a_i$. F can be represented as a skew binary tree with a height of 64 or a balanced binary tree with a height of seven, as shown in Fig. 2. An individual can contain at most $2^{64} - 1$ nodes if the predefined maximum depth is 64. A population containing so many large trees is highly complex and is thereby impracticable. On the other hand, if the predefined maximum depth is fixed at seven, it is very difficult to generate the ideal balanced tree. Moreover, the function F will never be obtained if the maximum depth is less than seven.

Using an acceptable and practicable individual size is a simple but dangerous way to avoid this problem. This problem has motivated us to develop this work. Since a long function can be viewed as a composition of a number of small functions, it is possible to combine a number of small GP solutions into a large one. Therefore, it is desirable to generate those small solutions with a practicable size of individuals and then use them to compose a larger solution. For example, consider the above function F and two functions $B = \prod_{i=1}^{32} a_i$ and $C = \prod_{i=33}^{64} a_i$. Clearly, F can be represented as $(B \times C)$, as shown in Fig. 3, where the tree representations of B and C have at most a height of 32 rather than 64. Functions B and C can be generated by two separate GPs and then are combined together to form F. Here we attempt to develop a method by which we can determine a proper node to combine small functions, for example, the shaded $\times$ operation in Fig. 3.

The method proposed in this paper is called layered genetic programming (LAGEP). It is a method based on MGP. LAGEP arranges populations in a layered architecture. Populations in the same layer evolve with identical training set and store the results of their best individuals into a dataset; this dataset becomes a new training set for the successive layer. After all layers have finished the evolution process, the output of the final layer is used as the result of LAGEP.

The rest of this paper is organized as follows. Section 2 describes the details of LAGEP. Section 3 presents and discusses

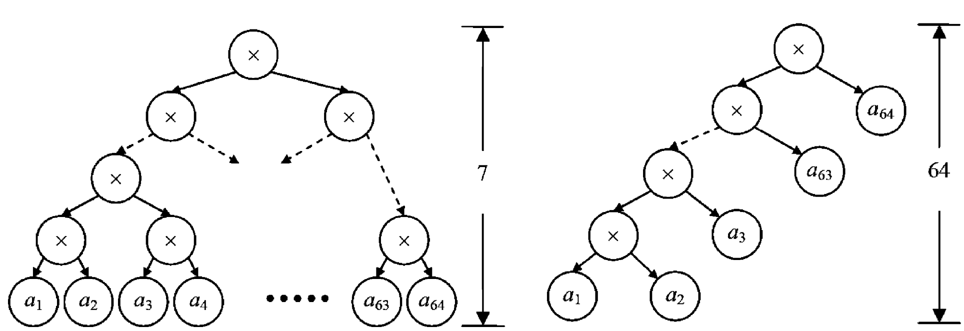

Fig. 2. Two possible representations of function $F = \sum_{i=1}^{64} a_i$. The left representation needs depth 7 but the right one needs depth 64.

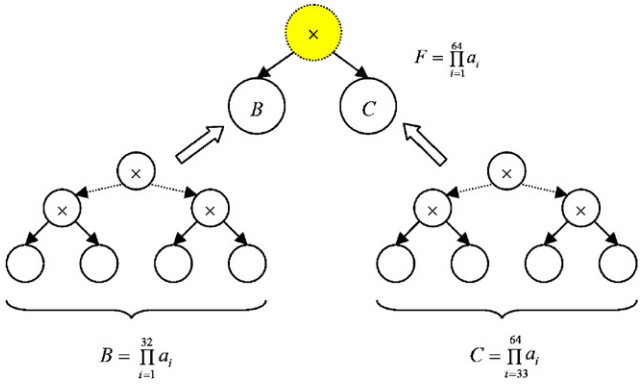

Fig. 3. Function F can be obtained from two short functions B and C. To combine B and C, we need to generate the multiplication operation as shown in the shaded circle.

the experimental results on selected classification problems. Conclusions are drawn in Section 4.

2. Proposed LAGEP method

LAGEP is based on multi-population method. In this section, we at first describe the design of each single population including a mutation weight tuning method. Then the design of LAGEP and the benefits of it are explained. The test phase and conflict problem are addressed afterward. Finally, an example demonstrates LAGEP.

2.1. Design of single population

GP is a supervised learning method. The training set is denoted by T containing m-dimensional training instances:

$$T = \{x_i | x_i = (a_{i1}, a_{i2}, \ldots, a_{ij}, \ldots, a_{im}), a_{ij} \in \mathbb{R}\}.$$

An individual is a possible solution for the given problem. In this paper, we tend to discover the optimal discriminant functions to solve the classification problem. Therefore, an individual is defined as a functional expression. An individual I is formulated by three components, *variables*, *constants*, and *operations*, which belong to the variable set S_v, the constant set S_c, and the operation set S_{op}, respectively. Variables are symbolic notations related to attributes of training instances. A variable A_i indicates ith attribute of an instance. S_c is a set of predefined constants. We define S_c as 10 floating numbers also belonging to [0, 1] because the attribute values of classification datasets used in this paper are normalized to [0, 1]. (The classification dataset will be described in Section 3.1.) S_{op} can contain logarithm operations or trigonometric functions, but we use only simple arithmetic operations because of two reasons. First, Kishore et al. [7] did experiments to show that the classification accuracy of using only simple arithmetic operations is sufficient to achieve high accuracy. Second, using simple operations is capable of reducing computation cost because individuals generated by small operations set are simple and efficient. Therefore, S_v, S_c, S_{op}, and I are defined as follows:

$$S_v = \{A_i | 1 \leqslant i \leqslant \text{the number of attributes of a training instance}\},$$

$$S_c = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\},$$

$$S_{op} = \{+, -, \times, /\},$$

$$I = (S_v, S_c, S_{op}),$$

where / in S_{op} is a protected division. When the denominator equals zero, the division will be set to 1.

The structure of individuals is the binary tree because operations are binary operations. The maximum number of available nodes of an individual is predefined and is called the individual length.

The fitness function is a function used to evaluate the fitness of every individual. When we perform the training task of a classification problem, we need to know which class is the *target class*. The target class is the class label for which we train the system to find solutions. Training instances are divided into *positive instances* if they belong to the target class and *negative instances* if they do not.

For a given training instance x, we say an individual I_j:

$$I_j \textit{ recognizes sample } x \textit{ iff } I_j(x) \geqslant 0;$$
$$I_j \textit{ repels sample } x \textit{ iff } I_j(x) < 0.$$

We try to find an individual that recognizes positive instances and repels negative instances.

An individual is capable of classifying a set of instances. We define a function Acc of a set of data S by

$$Acc(I_j, S) = \frac{\text{the number of objects of } S \text{ that are correctly classified by } I_j}{|S|}.$$

The fitness function F used in this work is made by

$$F(I_j) = Acc(I_j, T).$$

We use such a fitness function not only because an accurate discriminant function is desired but also because F will be computed many times so that should be as simple as possible.

A population P is a set of individuals and is defined by

$$P = \{I_1, I_2, \ldots, I_{|P|}\}.$$

The best individual produced by P is denoted as Λ and is derived by an evolution process of P. The *evolution process* mimics the natural selection mechanism by performing a systematic process on the population by genetic operators. Three primary genetic operators, *crossover*, *mutation*, and *reproduction*, are performed according to predefined rates R_c, R_m, and R_r, respectively. When the crossover operation is chosen, we first perform the selection process to pick two individuals from P. The crossover operator then produces two new individuals from these two individuals by randomly selecting a subtree from each individual and swapping the two selected subtrees to build two new individuals. If the mutation operator is chosen to perform, we pick one individual via the selection process. A node of the individual is randomly selected as the *mutation*

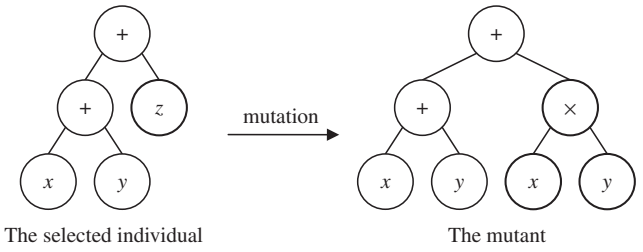

Fig. 4. An example of mutation. A mutant $(x + y + xy)$ is derived from the selected individual $(x+y+z)$. Both the mutation point of the select individual and the generated new subtree of the mutant are marked in bold circle.

point. Then the mutation operator replaces the mutation point by a randomly generated new node based on the S_v, S_c, and S_{op} to produce a mutant. While the new node is an operator, the mutation operator generates a subtree rooted at the mutation point. An example of mutation is shown in Fig. 4. The mutation operator is designed to escape from local optimum because the mutant may contain new structures that have never occurred on existing individuals. The reproduction operator keeps a selected individual alive to the next generation. It mimics the natural principle of survival of the fittest.

The evolutionary process in this paper is based on *elitism* strategy. After one generation is completed, a number of optimal individuals are reproduced directly to the next generation to keep surviving. In order to keep the diversity of the population, the number of reserved best individuals should be small. In this paper, we reserve two best individuals per generation. The remaining individuals continue performing crossover and mutation. When individual(s) are selected to perform crossover, or mutation, we insert their offspring, or its mutant, into a new population if and only if the offspring, or the mutant, has better fitness values than their parents. We use the deterministic tournament selection method to select individuals. This method at first chooses a number of individuals from the population at random and then returns the individual with the highest fitness value to be the selection result.

2.2. Adaptive mutation rate tuning method (AMRT)

As mentioned above, mutation operator is capable of generating individuals with new structures and mainly used to escape local optimum. Given a high R_m, the population tends to generate diverse individuals instead of discovering solutions from present individuals. Moreover, high R_m makes the GP system become a random search model, which is difficult to converge and generate stable results. In order to avoid such problem, R_m is usually much lower than R_c. However, when R_m is fixed at a small value, individuals may not have sufficient opportunity to mutate. As a result, the diversity of the population is limited. In particular, if some terminals are already good enough to classify samples, individuals may be stuck with such terminals. Since there is no exact guide to define R_c and R_m, we proposed a method called AMRT to raise R_m to perform the mutation operator more frequently when the generation increases.

The AMRT method at first considers the performance of individuals. In case individuals have similar fitness values, then AMRT is triggered to increase R_m. Otherwise, the population uses the initially given R_m. Moreover, because $R_m + R_c = 1$, to increase R_m implies to decrease R_c. AMRT performs every generation. At generation g the AMRT considers remaining generations to tune R_m and R_c

$$(R_m, R_c) \text{ at generation } g = \begin{cases} (R_m, R_c) & \text{if } \dfrac{f_{MAX}}{f_{AVERAGE}} > 2, \\ \left(\dfrac{\alpha}{R_c+\alpha}, 1-\dfrac{\alpha}{R_c+\alpha}\right), & \\ \quad \alpha = R_m \times \left(\dfrac{R_c}{R_m}\right)^{g/G} & \text{otherwise,} \end{cases}$$

where G is the maximum generation, f_{MAX} is the fitness value of the best individual in generation g; and $f_{AVERAGE}$ is the average fitness value of all individuals in generation g. From this formula, R_m increase smoothly and achieves 0.5 at the final generation, which means that the mutation operator and the crossover operator have the same chance to be selected.

We draw the curve of R_m. In Fig. 5 under these conditions: $G = 100$, (R_m, R_c) is initialized to (0.05, 0.95), and f_{MAX} is supposed to never larger than $2 \times f_{AVERAGE}$ during these 100 generations. This curve shows that R_m increases smoothly and ends up at 0.5.

The overfitting problem occurs when the trained solution excessively adapt to the training set. During the training phase, it is difficult to detect whether the overfitting occurs or not. Validation process can be used to avoid overfitting. Validation process uses a set of validation instances, V, to check the generalization of individuals. A good individual should derive good performance from the training set, i.e. high fitness value, and derive high classification accuracy from the validation set. When all generation complete, the best individual of each generation evaluates the classification accuracy with V. The population's best individual is the one that has maximum sum of fitness value and classification accuracy on V [21,22]. For this purpose, we define *score* of an individual I_j as

$$score(I_j) = F(I_j) + Acc(I_j, V).$$

Algorithm 1. proposed process of evolving single population:

(1) Initialize $|P|$ randomly generated individuals; define $g \leftarrow 0$; generate an empty population P'.
(2) Evaluate fitness value of individuals with the training set T.
(3) Perform reproduction on two best individuals. Insert them into population P'.
(4) Check and tune the mutation rate by AMRT.
(5) Select crossover or mutation according to R_c and R_m.
  (5.1) If the crossover is selected and $|P'| = |P| - 1$, jump to step (5.3).
  (5.2) If the crossover is selected, select two individuals as parents by tournament selection and perform the

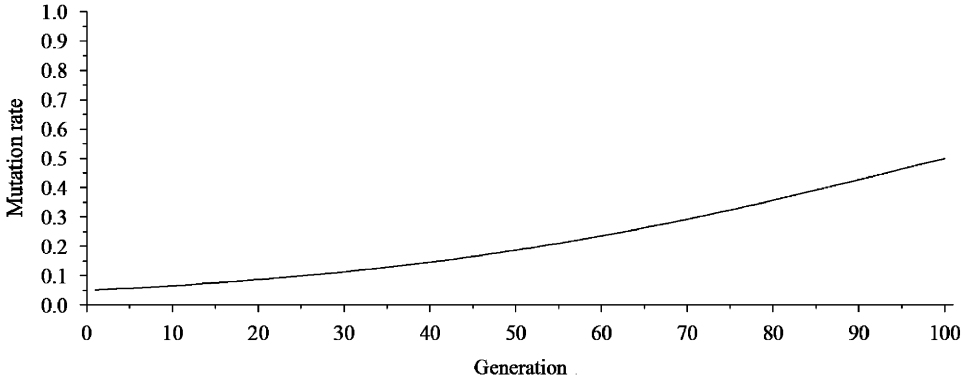

Fig. 5. The curve of W_m using AMRT given maximum generation $G = 100$.

crossover operator. Evaluate the fitness values of the offspring. Compare their fitness values with parents, and insert the best two individuals to P'.

(5.3) If mutation is selected, then perform the mutation operator on a selected individual and evaluate the fitness value for the mutant. Compare fitness values of the selected individual and the mutant, and insert the better individual to P'.

(6) Continue step (5) if $|P'| < |P|$. Otherwise, one *generation* is completed.

(7) Store the best individual of this generation Λ_g. $P \leftarrow P'$, $P' \leftarrow \emptyset$, and $g \leftarrow g + 1$.

(8) Repeat steps (2)–(6) if $g < G$.

(9) Evaluate score (Λ_i), $0 \leqslant i < G$.

(10) Output Λ_i to be the result if Λ_i has the highest score.

2.3. LAGEP layers

The LAGEP architecture is shown in Fig. 6, which provides an overview of the proposed method. LAGEP composes a number of layers. For each layer, it contains a set of populations to generate a set of best individuals. A new training set is produced by such individuals and the training set of the layer. Populations of the successive layer will use the new training set to evolve individuals. In the final layer, the results can be obtained.

LAGEP is a layered architecture model. A layer in LAGEP is a set of populations with a particular variable set and a particular training set. A layer L_i is defined by

$$L_i = (P_1, \ldots, P_i, \ldots, P_{l(i)}, T_i, S_v^i),$$

where P_i is a population, $l(i)$ is the number of populations in L_i, and S_v^i is the variable set used for all populations of L_i. A layer is a multi-population model. In this paper, we use the IMGP model [18] because IMGP is simpler than PADGP. Every population in IMGP model is independent to others. The evolution algorithm of each population does not require any change.

For layer L_i, a particular training set T_i is prepared and is used for the evolution processes of L_i's populations. Each population, as mentioned at previous section, generates an individual with best score. Since an individual is a discriminant

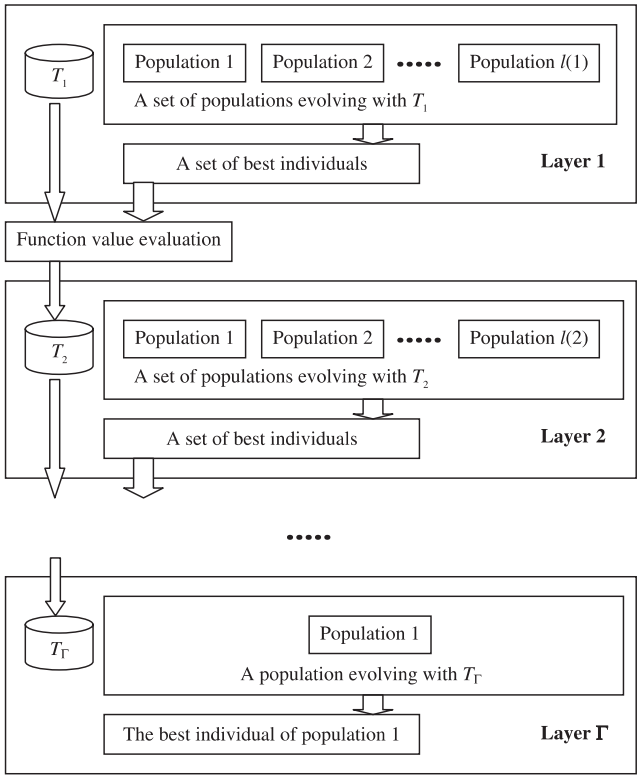

Fig. 6. The LAGEP architecture. LAGEP contains Γ layers. The ith layer has $l(i)$ populations. Those populations evolve with training set T_i.

function, after L_i's populations have performed the evolution processes, a set of discriminant functions $\{\Lambda_{i1}, \Lambda_{i2}, \ldots, \Lambda_{il(i)}\}$ is obtained. Based on these discriminant functions and training instances of T_i, a new training set T_{i+1} and a new variable set S_v^i can be constructed by

$$\begin{aligned} T_{i+1} = \{x_{(i+1)j} | x_{(i+1)j} &= (a_{(i+1)j1}, \ldots, a_{(i+1)jk}, \ldots, a_{(i+1)jl(i)}), a_{(i+1)jk} \\ &= \Lambda_{ik}(x_{ij}), x_{ij} \in T_i\}, \end{aligned}$$

$$S_v^i = \{A_{(i+1)1}, A_{(i+1)2}, \ldots, A_{(i+1)l(i)}\}.$$

An attribute of an instance $x_{(i+1)j}$ of T_{i+1} is made by a corresponding instance x_{ij} of T_i with a corresponding discriminant

function of previous layer. When T_{i+1} has been constructed, we say an *era* ends. The layer L_{i+1} is able to begin its evolution process with T_{i+1}.

The training instance $x_{(i+1)j} \in T_{i+1}$ is derived by $x_{ij} \in T$ through the set of discriminant functions, i.e. x_{ij} is transformed by the set of discriminant functions to a new space. Populations of L_i discover a set of function to separate instances of T_i. The function value of a discriminant function for an instance provides classification information. The layer architecture is capable of sending such information to next layer, L_{i+1}, to discover a set of discriminant function. Therefore, most of the positive instances will have positive attributes and most of negative instances will have negative attributes. Training instances in later layers should be easier to be classified.

According to the definitions provided so far, we define LAGEP as

$$LAGEP = \{L_i | i = 1, 2, \ldots, \Gamma\},$$

where Γ is the number of layers. The last layer is designed to have only one population. Although having more populations is possible, in this paper, we used this design to simplify the LAGEP system. The result of LAGEP is a single discriminant function and is denoted as Λ.

For a K-class classification problem LAGEP has to be trained K times for K different target classes. The K different T_Γ stands for training results of K classes. The algorithm of LAGEP evolution is shown in Algorithm 2.

Algorithm 2. LAGEP evolution:

(1) Let $T_1 \leftarrow$ given training set, $V \leftarrow$ given validation set, $i \leftarrow 1$.
(2) Perform Algorithm 1 for all populations in layer L_i, with training set T_i.
(3) Evaluate $\Lambda_{i1}, \Lambda_{i2}, \ldots, \Lambda_{il(i)}$ with all instances in T_i and store them into T_{i+1}. An *era* is completed.
(4) If $i \leqslant \Gamma$, then $i \leftarrow i + 1$. Jump to step (2).
(5) Change target class to the next class label and jump to step (1).

2.4. Advantages of layer architecture

In this section, we describe the advantages of using LAGEP. At first, we show that the result of LAGEP is a composition of small discriminant functions. The result of the last layer of LAGEP is a function Λ which can be represented by

$$\begin{aligned}
&\Lambda = \Lambda(A_{(\Gamma-1)1}, A_{(\Gamma-1)2}, \ldots, A_{(\Gamma-1)l(\Gamma-1)}),\\
&A_{(\Gamma-1)j} = \Lambda_{(\Gamma-1)j}(A_{(\Gamma-2)1}, A_{(\Gamma-2)2}, \ldots, A_{(\Gamma-2)l(\Gamma-2)}),\\
&A_{(\Gamma-2)j} = \Lambda_{(\Gamma-2)j}(A_{(\Gamma-3)1}, A_{(\Gamma-3)2}, \ldots, A_{(\Gamma-3)l(\Gamma-3)}),\\
&\vdots\\
&A_{2j} = \Lambda_{2j}(A_{11}, A_{12}, \ldots, A_{1l(1)}),\\
&A_{1j} = \Lambda_{1j}(A_1, A_2, \ldots, A_m),
\end{aligned}$$

where $(A_1, A_2, \ldots, A_m)$ is the original variable symbols of the original given training set T_1. Such expansion shows that the function Λ is a long function composed by a number of functions generated by layers.

Table 1
Prediction results of Λ_{ij}

	Belong to target class	Not belong to target class
$\Lambda_{ij} \geqslant 0$	R_1	R_2
$\Lambda_{ij} < 0$	R_3	R_4

A layer has a great probability of having higher fitness value better than its previous layer. Consider two populations $P_{(i+1)j} \in L_{(i+1)}$ and $P_{ij} \in L_i$. The classification result of Λ_{ij} with T_i is shown in Table 1. Obviously, we have

(1) $R_1 + R_2$ instances in T_{i+1} have the positive attribute A_{ij};
(2) $R_3 + R_4$ instances in T_{i+1} have the negative attribute A_{ij};
(3) R_1 positive instances have positive attribute A_{ij} and R_4 negative instances have negative attribute A_{ij}.

For any individual $I \in P_{(i+1)j}$ and $I = A_{ij}$, we have

$$F(I) = \left(\frac{R_1 + R_4}{|T|}\right) = \text{fitness value of } \Lambda_{ij}.$$

Since an individual containing only a variable is very likely to appear in a population, the fitness value of the best individual of the $P_{(i+1)j}$ is very likely to be better than the maximum fitness value of $\{\Lambda_{i1}, \Lambda_{i2}, \ldots, \Lambda_{il(i)}\}$.

2.5. The testing phase and Z-value measure

The set of test instances is denoted as TS. To predict the class of a test instance $y_i \in TS$, we substitute y_i to K LAGEPs responding to K different classes. The classification vector of y_i is defined as

$$cy_i = (r_{i1}, r_{i2}, \ldots, r_{iK}),$$
$$r_{ij} = \begin{cases} 1 & \text{if } \Lambda_{class=i}(y_i) \geqslant 0,\\ 0 & \text{otherwise.} \end{cases}$$

A problem called *conflict* occurs when $\sum_{j=1}^{K} r_{ij} > 1$, that is, conflict occurs when y_i is classified into two or more classes. This problem can be avoided by executing functions in the proper sequence, or the problem can be resolved by using additional techniques. Researchers have developed creative methods to solve the conflict problem. Kishore et al. [7] proposed a method to evaluate "strength of association" (SA) measured by each GP classification expression (GPCE). Ambiguous data are assigned to the class whose GPCE has the largest SA. They further used heuristic rules to improve accuracy. Muni et al. [12] proposed two methods to resolve the conflict problem: a heuristic rule-based scheme and a weighting scheme. Here we use a method called Z-value measure we proposed previously [4,10] because it is based on statistical theorem and accurate. Z-value measure employs means and standard deviations of all function values within the given training set to estimate Z-value for a conflict test instance. The class of such instance is

Table 2
A 2-class problem containing nine training instances in T_1

T_1	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	Class
x_1	0.20	0.10	0.10	0.10	0.20	0.10	0.20	0.10	0.10	***M***
x_2	0.20	0.10	0.10	0.10	0.20	0.10	0.30	0.10	0.10	***M***
x_3	0.50	0.10	0.10	0.10	0.20	0.10	0.20	0.10	0.10	***M***
x_4	0.50	0.40	0.60	0.80	0.40	0.10	0.80	1.00	0.10	***B***
x_5	0.50	0.30	0.30	0.10	0.20	0.10	0.20	0.10	0.10	***M***
x_6	0.20	0.30	0.10	0.10	0.30	0.10	0.10	0.10	0.10	***M***
x_7	0.30	0.50	0.70	0.80	0.80	0.90	0.70	1.00	0.70	***B***
x_8	1.00	0.50	0.60	1.00	0.60	1.00	0.70	0.70	1.00	***B***
x_9	1.00	0.90	0.80	0.70	0.60	0.40	0.70	1.00	0.30	***B***

determined to be class i if the ith function has the minimum Z-value. Furthermore, in case the test instance is not classified to any class, we also use the Z-value to find a probable class rather than directly assign it to a special class REJECT. An instance is assigned to class REJECT only when it has multiple minimum Z-value.

We introduce the main steps of Z-value measure in this section. Details of Z-value measure can be found in Ref. [4]. First, we compute vectors μ, standing for the mean, and σ^2, standing for the standard deviation, of training samples in T_Γ. Since every sample x_j in T_Γ is a scalar, μ and σ^2 are computed by

$$\mu = (\mu_1, \mu_2, \ldots, \mu_K),$$

$$\mu_i = \frac{\sum_{x_j \in T \text{ and } x_j \in \text{class } i} x_j}{\text{the number of training instances of class } i},$$

$$\sigma^2 = (\sigma_1^2, \sigma_2^2, \ldots, \sigma_K^2),$$

$$\sigma_i^2 = \frac{\sum_{x_j \in T \text{ and } x_j \in \text{class } i} (x_j - \mu_i)^2}{\text{the number of training instances of class } i}.$$

Second, for a test instance y_i, if $\sum_{j=1}^{K} r_{ij} = 1$, we do not have to execute its Z-value measure. If $\sum_{j=1}^{K} r_{ij} = 0$, we change all r_{ij} to 1. A K-dimensional vector Z_i is computed by

$$Z_i = (Z_{i1}, Z_{i2}, \ldots, Z_{iK}),$$

$$Z_{ij} = \begin{cases} \dfrac{|\Lambda_{class=j}(y_i) - \mu_j|}{\sigma_j} & \text{if } r_{ij} = 1, \\ \infty & \text{otherwise.} \end{cases}$$

We assign y_i to class j if Z_{ij} is the minimum among Z_i.

2.6. A brief example of LAGEP

In this section, we illustrate LAGEP by solving a 2-class problem, i.e. $K = 2$. Table 2 shows the training set T containing nine training instances where ***M*** stands for malignant and ***B*** stands for benign. The settings of this example are

$$\text{target class} = \boldsymbol{M}, \Gamma = 2, \quad LEN = 2^4 - 1 = 15,$$

$$LAGEP = \{L_1, L_2, L_3\},$$

$$L_1 = (P_{11}, P_{12}, P_{13}, T_1, S_v^1), \quad L_2 = (P_{21}, T_2, S_v^2)$$

Table 3
Training set T_2 generated by L_1

T_2	A_{11}	A_{12}	A_{13}	Class
x_{11}	0.9000	0.7000	1.9000	***M***
x_{12}	0.9000	0.3667	2.9000	***M***
x_{13}	0.9000	0.4000	1.9000	***M***
x_{14}	0.0000	−0.6000	−0.2000	***B***
x_{15}	0.9000	0.2000	1.9000	***M***
x_{16}	0.9000	2.7000	0.9000	***M***
x_{17}	−0.2000	0.1429	−0.3000	***B***
x_{18}	0.4286	−0.7429	0.3000	***B***
x_{19}	−0.1000	−0.9429	−0.3000	***B***

$$S_v^1 = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9\},$$

$$S_v^2 = \{A_{11}, A_{12}, A_{13}\},$$

$$S_{op} = \{+, -, \times, /\},$$

$$S_c = \{0.0, 0.1, 0.2, \ldots, 1.0\}.$$

Three best individuals generated by first layer are

$$\Lambda_{11} = (A_9/A_8) - A_6,$$

$$\Lambda_{12} = (A_5/A_7) - (A_3 + A_1),$$

$$\Lambda_{13} = (A_7/A_8) - A_8,$$

we construct T_1 according to Λ_{11}, Λ_{12}, and Λ_{13}. Training set T_1 is shown in Table 3.

Next, L_2 uses T_2 to evolve its population and generates Λ:

$$\Lambda = A_{11} + A_{12}.$$

Table 4 shows that Λ achieves perfect classification accuracy on the training set and shows the training results as well.

The solution of LAGEP for class ***M*** is

$$\Lambda = A_{11} + A_{12} = (A_9/A_8) - A_6 + (A_5/A_7) - (A_3 + A_1).$$

Here we omit the details of the training LAGEP with target class ***B*** and show its results with the training results of LAGEP with target class ***M*** in Table 5. The validation process is also omitted to simply this example.

Table 4
Training result generated by L_2

	Λ	Class
x_{11}	1.6000	***M***
x_{12}	1.2667	***M***
x_{13}	1.3000	***M***
x_{14}	−0.6000	***B***
x_{15}	1.1000	***M***
x_{16}	3.6000	***M***
x_{17}	−0.0571	***B***
x_{18}	−0.3143	***B***
x_{19}	−1.0429	***B***

Table 5
Training results responding to classes ***M*** and ***B***

	M	***B***	Class
x_{31}	1.6000	−0.6000	***M***
x_{32}	1.2667	−0.6000	***M***
x_{33}	1.3000	−0.3000	***M***
x_{34}	−0.6000	0.0000	***B***
x_{35}	1.1000	−0.1000	***M***
x_{36}	3.6000	−0.4000	***M***
x_{37}	−0.0571	0.5000	***B***
x_{38}	−0.3143	1.5000	***B***
x_{39}	−1.0429	1.2000	***B***

For two given test instances y_1 and y_2, we use the two trained LAGEPs corresponding to class ***M*** and ***B*** on y_1 and y_2 and obtain

$$\Lambda_M(y_1) = 0.3, \quad \Lambda_B(y_1) = -0.4,$$

$$\Lambda_M(y_2) = 0.3, \quad \Lambda_B(y_2) = 0.2,$$

We determine the class label of y_1 to be class ***M***. The conflict problem occurs at y_2. Through the Z-value measure [4], Z-values of y_2 corresponding to class ***M*** and ***B*** are 1.5886 and 1.0215, respectively. y_2 is classified to class ***B*** because Z_{2B} is the minimum one.

3. Experiments

In this section we describe the experiments and analyze classification results. To conduct the experiments described in this section, we developed a system based on the LAGEP Project [23] executed under an ACER VT7600GL, which is equipped with 3.0 GHz processor and 1.5 GB memory.

3.1. Experiment medical classification problems

In order to illustrate LAGEP we used six diagnostic problems selected from the PROBEN1 benchmark set of real-world problems [20], which was also used in Ref. [3]. These problems are originally from the UCI repository [17] and have been preprocessed by Ref. [20]. The values of all sets are normalized to the continuous range [0, 1]. Missing attributes are completed. Every attribute having m possible values is encoded by the 1-of-m method, i.e. using m binary attributes instead of the original attribute. Each problem prepared by PROBEN1 has been divided into three subsets: training, validation, and test. The training set contains the first half of the samples. The validation set includes the next 25% of the samples. The last 25% of the samples are test instances. Therefore, in this paper we do not re-separate each problem to new training set and test set. Furthermore, each problem in PROBEN1 has three different compositions with different distribution of instances, i.e. instances are separated to the training set, the validation set, and the test set by three different orders. This should increase confidence that classification results are not influenced by the distributions of the training set and test set. Therefore, we have 18 problems in total. We summarize these problems in Table 6.

3.2. The AMRT experiment

At first, we show the performance improved by using AMRT. Since AMRT is designed for single population, we use a population to conduct this experiment. Table 7 shows the experiment setting, which is denoted as ES1. The population size, maximum generation and the tournament size are referring to Ref. [21]. However, the crossover rate and the mutation rate in Ref. [21] are 0.35 and 0.65, respectively. Such combination of R_c and R_m is not proper. Mutation operator is not supposed to be the primary genetic operator because it is used to escape local optimum. Therefore, we consider the R_c and R_m of Ref. [12]. The crossover rate, mutation rate, and reproduction rate of Ref. [12] are 0.75, 0.15, and 0.1, respectively. In this paper, the reproduction operator is not performed by probability. We divide the reproduction rate, 0.1, equally to the crossover rate and the mutation rate. The number of available nodes of an individual used in Ref. [21] is 650 because the structure of individuals is not the binary tree structure. In this paper we use the most approximate number, 511, to be the maximum number of available nodes of an individual.

Table 6
Summary of selected problems

Problem	Classes	Number of features	Training instances	Validation instances	Test instances
Heart (HRT)	2	35	152	76	75
Horse (HRS)	3	58	182	91	91
Cancer (CAN)	2	9	350	175	174
Diabetes (DBT)	2	8	384	192	192
Gene (GEN)	3	120	1588	794	793
Thyroid (TRD)	3	21	3600	1800	1800

Table 7
Experiment settings (ES1) of AMRT experiment

Parameter	Value
Max generation G	100
Population size	2000
Individual length	511
Tournament size	6
R_c	0.8
R_m	0.2

Denoting the ES1 with AMRT and without AMRT as ES1.w and ES1.w/o, respectively, we perform them on the 18 classification problems 10 times. Table 8 shows accuracy comparisons and paired t-test between ES1.w and ES1.w/o. The three values in each cell stand for the highest accuracy, the average accuracy and the standard variation.

From Table 8, the average accuracy comparison and the paired t-test results show that AMRT is capable of improving classification performance of single population GP. ES1.w has

Table 8
Accuracy comparisons and paired t-test between ES1 with AMRT and ES1 without AMRT

Problem	ES1.w	ES1.w/o	Paired t-test	Problem	ES1.w	ES1.w/o	Paired t-test
HRT1	80.00	*81.33*		DBT1	*78.13*	76.56	
	76.80	76.67	0.4557		**72.50**	72.34	0.4667
	3.09	2.69			2.76	3.54	
HRT2	*98.67*	94.67		DBT2	*75.52*	75.00	
	93.33	92.27	0.2332		**71.25**	71.15	0.4584
	3.20	2.16			2.44	1.79	
HRT3	*88.00*	86.67		DBT3	77.60	*78.13*	
	84.40	82.93	0.0769		**75.16**	74.53	0.2594
	2.60	2.50			2.53	1.46	
HRS1	*71.43*	67.03		GEN1	89.66	89.41	
	64.51	62.53	0.1593		**85.31**	85.21	0.4787
	4.61	2.28			4.37	3.10	
HRS2	*67.03*	65.93		GEN2	*90.04*	88.65	
	61.98	60.77	0.2358		**85.98**	83.32	**0.0343**
	3.78	2.64			3.95	5.97	
HRS3	*65.93*	60.44		GEN3	*88.27*	88.15	
	61.21	55.27	**0.0010**		**84.59**	81.78	**0.0397**
	3.11	3.74			3.00	5.07	
CAN1	98.85	98.85		TRD1	97.72	97.72	
	97.70	97.70	0.5000		**97.15**	95.94	**0.0361**
	0.72	0.77			0.47	1.74	
CAN2	*96.55*	94.83		TRD2	98.28	*98.67*	
	94.89	94.08	**0.0022**		**97.33**	96.84	0.0766
	0.69	0.47			1.22	1.34	
CAN3	97.13	97.13		TRD3	98.06	*98.50*	
	96.32	96.44	0.3097		**96.95**	96.69	0.3332
	0.78	0.45			0.92	1.38	

Three values in a cell stand for highest accuracy, average accuracy, and standard deviation. The better highest accuracy is marked in italic face. The better average accuracy and significance p-value ($p < 0.05$) are marked in bold face.

Table 9
Experiment settings of traditional single population GP and LAGEP

	Settings	Γ	$l(i)$	Population size of each	Total number of individuals	Individual length
Traditional GP	ES1	1	1	2000	2000	511
LAGEP	ES2	2	2, 1	666	1998	255
LAGEP	ES3	2	3, 1	500	2000	255
LAGEP	ES4	2	4, 1	400	2000	255
LAGEP	ES5	2	5, 1	333	1998	255
LAGEP	ES6	2	6, 1	286	2002	255

Table 10
Accuracy comparisons of six experiment settings and four methods cited from [21]

Problem	ES1	ES2	ES3	ES4	ES5	ES6	G^3P D	G^3P F	G^3P A	G^3P P
HRT1	80.00	80.00	82.67	82.67	80.00	82.67	82.10	81.66	76.86	76.86
	76.80	77.47	**78.27**	76.67	76.00	77.33	78.13	77.88	76.86	74.37
	3.09	2.98	3.45	2.90	2.18	3.27	2.12	1.93	0.00	1.44
HRT2	98.67	94.67	97.33	96.00	96.00	96.00	82.10	79.04	80.35	78.17
	93.33	93.47	93.87	93.33	93.60	**94.27**	78.32	76.32	79.66	74.68
	3.20	1.72	1.91	1.66	1.86	2.81	1.81	1.89	2.15	3.73
HRT3	88.00	88.00	88.00	86.67	89.33	89.33	77.30	74.68	74.68	75.11
	84.40	**85.20**	83.87	84.13	84.40	84.80	74.24	74.24	71.40	74.10
	2.60	2.31	1.83	1.33	2.27	2.45	1.46	0.61	1.59	1.76
HRS1	71.43	73.63	68.13	71.43	68.13	73.63	71.12	61.12	71.12	63.33
	64.51	**66.59**	64.73	65.05	65.38	65.38	66.45	58.52	69.39	59.63
	4.61	3.28	1.90	3.39	2.65	4.22	2.78	4.49	2.05	3.90
HRS2	67.03	64.84	70.33	67.03	68.13	68.13	63.33	62.23	61.12	64.45
	61.98	61.98	62.31	62.31	61.65	**63.63**	57.73	59.73	56.50	61.12
	3.78	2.15	3.81	2.98	3.61	3.38	3.48	4.29	2.55	4.84
HRS3	65.93	68.13	68.13	63.74	67.03	62.64	71.11	72.23	68.89	63.33
	61.21	62.31	62.42	61.43	**63.52**	61.65	64.06	64.72	63.73	58.89
	3.11	2.69	2.73	2.95	3.18	0.96	3.04	5.00	2.53	4.44
CAN1	98.85	98.85	99.43	98.85	98.28	98.85	97.71	97.71	97.13	97.13
	97.70	97.70	**97.82**	97.76	97.70	**97.82**	96.21	95.61	94.34	95.69
	0.72	0.54	0.85	0.79	0.77	0.80	1.01	1.42	1.24	0.94
CAN2	96.55	95.40	95.98	95.98	95.98	95.40	98.28	98.28	97.13	97.71
	94.89	94.60	94.89	**94.94**	94.77	94.83	95.32	95.55	91.70	95.17
	0.69	0.48	0.92	0.59	0.74	0.47	2.18	1.23	2.16	1.19
CAN3	97.13	97.13	97.13	97.13	97.13	97.13	97.71	96.56	98.86	97.71
	96.32	96.09	96.38	**96.61**	96.32	96.03	95.61	95.10	94.72	95.58
	0.78	0.71	0.61	0.57	0.40	0.69	1.36	0.83	1.70	1.43
DBT1	78.13	75.00	77.60	76.04	75.52	76.56	73.3	78.02	77.49	76.97
	72.50	72.71	**73.91**	73.13	72.08	71.98	68.3	73.53	75.46	73.18
	2.76	2.04	2.24	1.98	1.94	2.44	3.24	3.40	1.26	2.56
DBT2	75.52	75.00	73.96	73.96	75.00	74.48	74.35	76.44	76.97	76.97
	71.25	71.46	71.46	71.88	**72.29**	72.08	68.7	75.22	74.59	72.92
	2.44	1.80	1.32	1.15	2.19	1.63	3.48	1.22	1.15	2.65
DBT3	77.60	78.65	78.65	78.65	77.08	77.60	80.11	78.01	75.92	75.92
	75.16	**75.99**	75.36	75.16	75.16	75.47	71.21	75.75	71.24	71.79
	2.53	2.72	1.32	2.13	1.04	1.91	5.11	1.64	1.84	2.16
GEN1	89.66	88.65	90.92	89.66	88.27	87.77	77.68	88.03	68.23	87.27
	85.31	85.41	**86.15**	85.85	85.36	85.49	66.97	62.88	65.26	67.50
	4.37	1.74	2.78	2.34	1.92	1.74	6.70	14.99	4.19	14.93
GEN2	90.04	89.79	93.06	91.55	90.29	91.30	70.37	85.63	68.73	79.45
	85.98	86.61	87.69	86.80	**88.25**	87.93	62.97	62.73	58.52	70.88
	3.95	2.77	2.25	3.33	1.58	3.52	4.71	11.44	4.08	12.12
GEN3	88.27	91.43	86.13	88.78	88.40	87.14	73.52	88.03	67.47	75.11
	84.59	**86.02**	84.82	84.83	85.11	84.97	67.79	62.96	62.17	74.10
	3.00	3.64	1.14	3.00	2.66	1.43	5.06	11.81	7.37	1.76
TRD1	97.72	97.89	97.67	98.00	98.17	98.44	97.11	94.72	94.56	94.45
	97.15	97.19	97.16	97.18	**97.52**	97.19	95.04	93.92	94.50	93.74
	0.47	0.28	0.49	0.55	0.55	0.66	0.81	0.57	0.80	0.69
TRD2	98.28	98.28	98.28	98.50	98.78	98.33	97.34	94.56	94.44	94.56
	97.33	97.24	97.43	97.27	**97.57**	97.54	94.27	94.05	93.92	94.05
	1.22	1.04	0.72	1.18	0.76	0.64	1.25	0.52	0.60	0.60
TRD3	98.06	98.22	97.94	98.28	98.33	98.39	98.06	94.56	94.61	95.78
	96.95	97.06	97.21	97.20	97.04	**97.64**	94.55	93.97	94.28	94.51
	0.92	0.89	0.58	0.82	1.34	0.58	1.37	0.82	0.50	0.84

Three values in a cell stand for highest accuracy, average accuracy, and standard deviation. The best average accuracies of ES1 and LAGEP settings are marked by bold face.

Table 11
Paired t-test results between ES1 and other five experiment settings

Problem	ES1 vs. ES2	ES1 vs. ES3	ES1 vs. ES4	ES1 vs. ES5	ES1 vs. ES6
HRT1	0.3004	0.1460	0.4576	0.2754	0.3392
HRT2	0.4576	0.3526	0.5000	0.3925	0.2358
HRT3	0.2125	0.3097	0.3902	0.5000	0.3817
HRS1	0.1759	0.4325	0.3819	0.2925	0.3429
HRS2	0.5000	0.4233	0.4254	0.4270	0.1762
HRS3	0.2007	0.2132	0.4427	0.0602	0.3643
CAN1	0.5000	0.3925	0.4236	0.5000	0.3820
CAN2	0.1494	0.5000	0.4236	0.3849	0.3988
CAN3	0.2955	0.4361	0.1815	0.5000	0.2201
DBT1	0.4267	0.0621	0.2608	0.3021	0.3021
DBT2	0.4076	0.4064	0.1606	0.1685	0.1966
DBT3	0.2907	0.3908	0.5000	0.5000	0.3888
GEN1	0.4772	0.2882	0.3807	0.4884	0.4549
GEN2	0.3073	0.1218	0.3299	0.0567	0.1645
GEN3	0.1757	0.4143	0.4311	0.3359	0.3668
TRD1	0.4226	0.4794	0.4426	0.0552	0.4328
TRD2	0.4445	0.4198	0.4620	0.3298	0.3254
TRD3	0.4075	0.2357	0.2526	0.4344	**0.0258**

Significant p-values ($p < 0.05$) are marked in bold face.

higher average accuracy than ES1.w/o except the CAN3 problem. Although AMRT is not proposed to enhance the solution quality, Table 8 shows that for most problems ES1.w discovers better discriminant functions. Based on these observations, all of the following experiments will use the AMRT method.

3.3. LAGEP experiment

In this paper, we use five two-layer LAGEP experiment settings to illustrate the performance of LAGEP. Table 9 shows six experiment settings including ES1. In this paper, we focus on investigating the behavior of the LAGEP architecture with different number of populations rather than with different number of layers. Table 9 shows that all settings use similar numbers of individuals. These settings are referred to Ref. [21] because we are going to compare the results with that. LAGEP is proposed to utilize short discriminant functions to obtain a longer function. We set the individual length of them to be 255. Every medical dataset is performed with each experiment setting for 10 times.

Table 10 shows accuracy comparisons of the six settings and four different GP structures cited from [21], G^3P for decision trees (G^3P D), G^3P for fuzzy rule based systems (G^3P F), G^3P for artificial neural networks (G^3P A), and G^3P for fuzzy Petri-nets (G^3P P). The values in each cell stand for the highest accuracy, the average accuracy and the standard variation. Table 11 shows the paired t-test results between ES1 and other five LAGEP settings.

3.3.1. Comparing classification accuracy between LAGEP and ES1

At first, we compare the classification accuracies between ES1 and LAGEP settings from Tables 10 and 11. ES1 has higher average accuracy than at least one LAGEP setting only in problems HRT1, HRT3, HRS2, CAN2, CAN3, DBT1, and TRD2. However, the best average accuracy of each problem is obtained by one of the five LAGEP settings. Table 11 shows that the significant p-value occurs at only problem TRD3 ($p = 0.0258$) where ES6 has higher average accuracy. If the significance level α increases to 0.1, four more significance p-values occur at ES5 with HRS3 ($p = 0.0602$), ES3 with DBT1 ($p = 0.0621$), ES5 with GEN2 ($p = 0.0567$), and ES5 with TRD1 ($p = 0.0552$). In contrast, ES1 does not significantly outperform any LAGEP setting over the 18 problems.

The classification accuracies of test set obtained by LAGEP and ES1 are similar. The reason of this situation could be explained by the inadequate generalization of LAGEP and the inconsistencies of instances of training set, validation set, and test set. Table 12 shows the average score values of ES1 and the population of the second layer of LAGEP settings. The paired t-test between ES1's score values and each LAGEP setting's score values is also conducted as shown in Table 12. LAGEP achieves significantly better score value than ES1 except for TRD problems. ES1 has higher average score value for TRD1 and TRD3 problems only, but it performs worse than all LAGEP settings in these problems. Furthermore, ES2, ES5, and ES6 achieves significantly better average score values than ES1 for CAN2 problem, but Table 10 shows that ES2, ES5, and ES6 have lower average accuracy. In summary, LAGEP has been slightly affected by the overfitting problem even though the validation process has been applied. The overfitting is not serious because we found that some LAGEP settings having highest average score values obtain highest average accuracies. For instance, for CAN1 problem, ES3 and ES6 have highest score value, 1.9627 and 1.9626, respectively, and they both achieve the highest average accuracy, 97.82. If the test set could

Table 12
Comparison of training performance of ES1 and LAGEP settings

Problem	ES1	ES2	ES3	ES4	ES5	ES6
HRT1	1.7898	1.8135	1.8280	1.8257	1.8191	1.8105
		0.0023	**0.0015**	**0.0006**	**0.0300**	**0.0275**
HRT2	1.6875	1.7454	1.7438	1.7500	1.7155	1.7158
		0.0001	**0.0005**	**0.0000**	**0.0161**	**0.0030**
HRT3	1.7477	1.7697	1.7944	1.7724	1.7727	1.7734
		0.0438	**0.0004**	**0.0183**	**0.0211**	**0.0146**
HRS1	1.6762	1.7071	1.7247	1.7255	1.7108	1.7093
		0.0011	**0.0001**	**0.0001**	**0.0086**	**0.0004**
HRS2	1.6868	1.7366	1.7385	1.7496	1.7372	1.7225
		0.0007	**0.0011**	**0.0001**	**0.0001**	**0.0028**
HRS3	1.6908	1.7275	1.7432	1.7399	1.7357	1.7443
		0.0095	**0.0004**	**0.0008**	**0.0005**	**0.0007**
CAN1	1.9537	1.9596	1.9627	1.9577	1.9610	1.9626
		0.0048	**0.0017**	**0.0324**	**0.0022**	**0.0001**
CAN2	1.9707	1.9741	1.9743	1.9734	1.9736	1.9749
		0.0127	**0.0017**	0.0532	**0.0005**	**0.0085**
CAN3	1.9560	1.9579	1.9610	1.9639	1.9649	1.9601
		0.1154	**0.0038**	**0.0001**	**0.0001**	**0.0169**
DBT1	1.5901	1.6061	1.6301	1.6206	1.6090	1.6147
		0.0446	**0.0002**	**0.0091**	**0.0081**	**0.0071**
DBT2	1.5866	1.6055	1.6206	1.6163	1.6096	1.6174
		0.0763	**0.0018**	**0.0048**	**0.0036**	**0.0043**
DBT3	1.5645	1.5956	1.5885	1.5957	1.5980	1.5820
		0.0060	**0.0109**	**0.0043**	**0.0012**	0.0556
GEN1	1.7915	1.8307	1.8262	1.8376	1.8435	1.8471
		0.0778	0.0761	**0.0299**	**0.0100**	**0.0310**
GEN2	1.7732	1.8161	1.8392	1.8090	1.8432	1.8338
		0.0089	**0.0044**	0.0959	**0.0018**	**0.0291**
GEN3	1.7786	1.8285	1.8391	1.8438	1.8406	1.8367
		0.0516	**0.0409**	**0.0176**	**0.0354**	**0.0401**
TRD1	1.9779	1.9773	1.9769	1.9786	1.9804	1.9801
		0.3867	0.3175	0.3699	0.2035	0.1924
TRD2	1.9719	1.9725	1.9746	1.9737	1.9769	1.9743
		0.4627	0.2188	0.3581	0.0642	0.2621
TRD3	1.9779	1.9773	1.9741	1.9726	1.9739	1.9752
		0.3867	0.0909	0.1151	0.1328	0.1676

For each problem, ES1 column shows its average score value. A LAGEP setting column shows the average score value of the population of the second layer and paired t-test result between it and ES1. Significant p-values ($p < 0.05$) are marked in bold face.

be more consistent with training set and validation set, the enhancement of classification accuracy of LAGEP would be considerable.

3.3.2. Comparing classification accuracy between LAGEP and four cited methods

The classification accuracies of the four cited methods are various. LAGEP has lower average accuracy than one of the four methods for problems HRS1, HRS3, CAN2, DBT1, and DBT2. However, we found that LAGEP performs amazingly better than the four cited methods for HRT2, HRT3, GEN2, and TRD1. For those problems, the lowest average accuracy of LAGEP setting is higher than the maximum highest accuracy of the four methods.

3.3.3. Comparing classification accuracy between LAGEP settings

The relation between the number of populations and the classification accuracy is not clear. ES5 achieves best average accuracy for five problems, which is the most one. ES2,

Table 13
The average score value of populations of ES2 and the paired t-test results L_iP_j indicates the jth population of the ith layer

Problem	ES2 L_1P_1	ES2 L_1P_2	ES2 L_2P_1	Paired t-test L_1P_1 vs. L_2P_1	Paired t-test L_1P_2 vs. L_2P_1
HRT1	1.7576	1.7720	1.8135	**0.0000**	**0.0000**
HRT2	1.6882	1.6737	1.7454	**0.0000**	**0.0000**
HRT3	1.7329	1.7155	1.7697	**0.0000**	**0.0000**
HRS1	1.6577	1.6423	1.7071	**0.0000**	**0.0000**
HRS2	1.6795	1.6885	1.7366	**0.0000**	**0.0000**
HRS3	1.6700	1.6723	1.7275	**0.0000**	**0.0000**
CAN1	1.9519	1.9499	1.9596	**0.0023**	**0.0000**
CAN2	1.9646	1.9664	1.9741	**0.0005**	**0.0001**
CAN3	1.9511	1.9519	1.9579	**0.0000**	**0.0004**
DBT1	1.5733	1.5754	1.6061	**0.0004**	**0.0001**
DBT2	1.5660	1.5728	1.6055	**0.0000**	**0.0000**
DBT3	1.5505	1.5434	1.5956	**0.0000**	**0.0000**
GEN1	1.7897	1.7601	1.8307	**0.0001**	**0.0000**
GEN2	1.7770	1.7585	1.8161	**0.0077**	**0.0003**
GEN3	1.7824	1.7372	1.8285	**0.0119**	**0.0001**
TRD1	1.9658	1.9694	1.9773	**0.0000**	**0.0028**
TRD2	1.9582	1.9660	1.9725	**0.0005**	**0.0034**
TRD3	1.9658	1.9694	1.9773	**0.0000**	**0.0028**

Significant p-values ($p < 0.05$) are marked in bold face.

Table 14
The average training time of six experiment settings and 18 problems in seconds

Problem	ES1	ES2	ES3	ES4	ES5	ES6
HRT1	299.12	179.61	169.50	183.20	180.49	164.36
HRT2	334.87	199.01	180.15	178.64	173.70	161.39
HRT3	299.17	200.42	179.19	166.39	170.32	168.22
HRS1	452.25	302.19	305.34	309.23	276.03	286.49
HRS2	429.87	281.04	308.62	290.77	258.99	265.44
HRS3	420.87	325.74	283.43	297.34	266.15	287.89
CAN1	313.05	238.19	235.37	226.36	239.20	239.61
CAN2	349.83	247.53	245.30	244.63	228.43	233.43
CAN3	314.57	206.97	215.00	230.19	222.49	225.88
DBT1	421.83	298.17	294.32	296.38	260.71	248.41
DBT2	344.72	270.59	265.95	282.44	277.93	251.84
DBT3	391.46	274.79	264.93	259.56	249.48	252.10
GEN1	1446.09	1232.41	1285.21	1190.22	1115.65	1157.48
GEN2	1615.26	1287.77	1262.48	1252.35	1118.27	1095.65
GEN3	1585.13	1212.39	1277.77	1157.58	1162.54	1110.59
TRD1	4410.35	2680.82	2789.47	2497.28	2612.90	2522.12
TRD2	5243.94	2774.68	2702.87	2738.57	2725.54	2649.07
TRD3	4451.99	2682.36	2613.43	2495.75	2670.19	2628.05

ES3 and ES6 achieve best average accuracy for four problems. From Table 10, it is not certain that using more populations can obtain higher classification accuracy. Considering the training performance from Table 12, both ES3 and ES5 have best average score values for five problems. ES2 have best average score value for TRD3 only. Based on these observations, the number of populations could be irrelevant to obtaining good discriminant functions.

3.3.4. The improvement of score value

The performance of second layer is supposed to be better than the first layer because the population of the second layer uses the transformed training set. We use ES2 to demonstrate the performance improvement. Table 13 illustrates the 18 problems and average score values of the three populations of ES2, where L_1P_1, L_1P_2, and L_2P implies the first population and the second population of the first layer, and the population of

the second layer, respectively. Both score values obtained by populations of the first layer are lower than the score value of the population of the second layer. Table 13 also shows the paired t-test results between L_1P_1 and L_2P_1, and L_1P_2 and L_2P_1. It is obvious that L_2P_1 is significantly better than the two populations of the first layer. This result confirms that the training performance, the accuracy of classifying the training set and the validation set, can be improved by the proposed layer architecture.

3.3.5. Comparing elapsed training time

LAGEP not only improves effectiveness but also reduces the training time. Table 14 shows the elapsed training time of six experiment settings. Obviously, LAGEP settings use much less time than single population GP does. The average elapsed training time of every LAGEP setting is at least 10% less than it is of ES1. The decrease in time is huge for TRD problems where every LAGEP setting uses at most 64% of time of ES1.

The evolutionary process of a population spends most of the time on evaluating fitness values for individuals. The individual length directly affects the required computation time. LAGEP settings and ES1 perform under the same conditions of the number of generations and the number of total individuals. LAGEP requires less training time because it uses shorter individuals. Concluding the observation from the classification accuracy and the elapsed time, LAGEP is more efficient than single population GP.

4. Conclusions and future work

In this paper, we propose a MGP method, LAGEP. LAGEP arranges a number of populations into a layer. Every layer evolves its populations to generate a set of discriminant functions. These functions transform the training set to a new training set, which is used for successive layer. The evolution process of every population is efficient because it evolves with short individuals. We also proposed a method to prevent falling into a local optimum for a long time called AMRT.

Experiment results show that LAGEP is capable of generating a high accuracy discriminant function efficiently. Moreover, an experiment comparing single population with AMRT and without AMRT shows that AMRT is effective. We found that LAGEP is affected by overfitting problem slightly, thus we show the average score values of single population GP and LAGEP settings to illustrate that LAGEP settings have significant higher training performance. We also use one of LAGEP settings to show that the training performance can be improved by the layered architecture. Although the classification accuracy is similar, the elapsed training time of LAGEP is much less than single population GP. Experiment results also show that the number of populations could be irrelevant to either training performance or classification accuracy of test set.

In this paper, LAGEP used only two layers. We intend to investigate the performance of LAGEP with more layers in future. Furthermore, we are interested in LAGEP with different population configurations that is called heterogeneous MGP model [18]. We also intend to develop further research on feature selection and feature generation based on the LAGEP architecture.

Acknowledgments

We would like to express our appreciation to the anonymous reviewers for their useful suggestions and revision. We also wish to thank Dr. Hsinchun Chen and Jiexun Li for many helpful discussions and comments.

References

[1] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, 1992.
[2] W. Banzhaf, P. Nordin, R.E. Keller, F.D. Framcone, Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Application, Morgan Kaufmann, San Francisco, CA, 1998.
[3] M. Brameier, W. Banzhaf, A comparison of linear genetic programming and neural networks in medical data mining, IEEE Trans. Evol. Comput. 5 (1) (2001) 17–26.
[4] B.C. Chien, J.Y. Lin, W.P. Yang, Learning effective classifiers with Z-value measure based on genetic programming, Pattern Recognition 37 (2004) 1957–1972.
[5] I. De Falco, A. Della Cioppa, E. Tarantino, Discovering interesting classification rules with genetic programming, Appl. Soft Comput. 23 (2002) 1–13.
[6] A. Freitas, A genetic programming framework for two data mining tasks: classification and generalized rule induction, in: Proceedings of Second Annual Conference on Genetic Programming, Stanford University, United States, July 1997, Morgan Kaufmann Press, CA, pp. 96–101.
[7] J.K. Kishore, L.M. Patnaik, V. Mani, V.K. Agrawal, Application of genetic programming for multicategory pattern classification, IEEE Trans. Evol. Comput. 4 (3) (2000) 242–258.
[8] A. Konstam, Group classification using a mix of genetic programming and genetic algorithms, in: Proceedings of the 1998 ACM Symposium of Applied computing, Atlanta, Georgia, United States, February 27–March 1, 1998, pp. 308–312.
[9] M. Kotani, S. Ozawa, M. Nakai, K. Akazawa, Emergence of feature extraction function using genetic programming, in: Proceedings of Third International conference on Knowledge-based Intelligent Information Engineering System (KES '99), Adelaide, Australia, 1999, pp. 149–152.
[10] J.Y. Lin, B.C. Chien, T.P. Hong, A function-based classifier learning scheme using genetic programming, in: Proceedings of sixth Pacific-Asia conference on Knowledge Discovery and Data Mining (PAKDD '02), Taipei, Taiwan, May 5–8, 2002, Springer, Heidelberg, Germany, pp. 92–103.
[11] T. Loveard, V. Ciesielski, Representing classification problems in genetic programming, in: Proceedings of the 2001 Congress on Evolutionary Computation, May 27–30, 2001, pp. 1070–1077.
[12] D.P. Muni, N.R. Pal, J. Das, A novel approach to design classifiers using genetic programming, IEEE Trans. Evol. Comput. 8 (2) (2004) 183–196.
[13] J. Sherrah, R.E. Bogner, A. Bouzerdoum, Automatic selection of features for classification using genetic programming, in: Proceedings of Australian and New Zealand Conference on Intelligent Information Systems, Adelaide, SA, Australia, November 18–20, 1996, pp. 284–287.
[17] C. Blake, E. Keogh, C.J. Merz, UCI Repository of machine learning databases, Irvine, University of California, Department of Information and Computer Science, 1998. [Online]. Available at ⟨http://www.ics.uci.edu/~mlearn/MLRepository.html⟩.
[18] F. Fernández, M. Tomassini, L. Vanneschi, An empirical study of multipopulation genetic programming, Genet. Programming Evolvable Mach. 4 (2003) 21–51.
[19] C.C. Bojarczuk, H.S. Lopes, A.A. Freitas, Discovering comprehensible classification rules using genetic programming: a case study in a medical domain, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99), Orlando, FL, USA, 1999, pp. 953–958.

[20] L. Prechelt, PROBEN1—a set of neural network benchmark problems and benchmarking rules, Technical Report 21/94, University of Karlsruhe, Karlsruhe, Germany, 1994.
[21] A. Tsakonas, A comparison of classification accuracy of four genetic programming-evolved intelligent structures, Inf. Sci. 176 (2006) 691–724.
[22] C. Cagne, M. Schoenauer, M. Parizeau, M. Tomassini, Genetic Programming, Validation Sets, and Parsimony Pressure, in: Proceedings of the Ninth European Conference on Genetic Programming, Budapest, Hungary, 2006, pp. 109–120.
[23] J.Y. Lin, LAGEP Project [Online]. Available at ⟨http://www.cis.nctu.edu.tw/∼gis91815/lagep/lagep.html⟩.

About the Author—JUNG-YI LIN was born in Taitung, Taiwan. He received the M.S. degree in Computer Science and Information Engineering from I-Shou University in 2002. He is currently a Ph.D. candidate in Computer Science, National Chiao Tung University, HsinChu, Taiwan. Lin is currently a visiting scholar at Artificial Intelligence Lab, Department of MIS, University of Arizona, Arizona, USA. His research interests include machine learning, data mining, and knowledge discovery.

About the Author—HAO-REN KE was born on June 29, 1967 in Taipei, Taiwan, Republic of China. He received the B.S. degree in 1989 and his Ph.D. degree in 1993, both in Computer and Information Science, from National Chiao Tung University. Now he is a professor of the Library, and Institute of Information Management, National Chiao Tung University (NCTU). He is also the associate director of the NCTU Library. His research interests include digital library, digital museum, information retrieval, web service, and data mining. He can be contacted at: claven@lib.nctu.edu.tw.

About the Author—BEEN-CHIAN CHIEN received the Ph.D. in Computer Science and Information Engineering from National Chiao Tung University in 1992. He was an associate professor of the Department of Computer Science and Information Engineering, I-Shou University, Kaohsiung, Taiwan, from 1996 to 2004. Currently, he is a professor and the head of the department of computer science and information engineering, national university of Tainan, Tainan, Taiwan. His current research activities involve machine learning, content-based image retrieval, intelligent information retrieval and data mining.

About the Author—WEI-PANG YANG was born on May 17, 1950 in Hualien, Taiwan. He received the B.S. degree in mathematics from National Taiwan Normal University in 1974, and the M.S. and Ph.D. degrees from the National Chiao Tung University in 1979 and 1984, respectively, both in Computer Engineering. He was a professor of the Department of CSIE and Department of CIS at the National Chiao Tung University, Hsinchu, Taiwan. He was a visiting scholar at the Harvard University and at the University of Washington. He was the Director of the Computer Center of National Chiao Tung University. Dr. Yang is currently the Head of the Department of Information Management and is the Dean of College of Management. His research interests include database theory and application, information retrieval, data miming, digital library, and digital museum.