MERGE: A Novel Evolutionary Algorithm based on Multi Expression Gene Programming

Shucheng Dai[1], Changjie Tang[1], Mingfang Zhu[1,2],Yu Chen[1],
Peng Chen[1], Shaojie Qiao[1],Chuan Li[1]
[1]*School of Computer Science, Sichuan University, Chengdu, China*
[2]*Dept. of Computer Sci. and Technol, Shaanxi Univ. of Technol Hanzhong, China*
{*daishucheng,tangchangjie,zhumingfang*}*@cs.scu.edu.cn*

Abstract

Gene Expression Programming (GEP) is a new member in genetic computing. The traditional GEP lacks the power to handle very complex function mining problems due to its limited express capability. To solve the problem, this paper presents a new evolutionary algorithm named Multi ExpRession GEne programming (MERGE). The main contributions include: (a) Provides a novel hierarchical gene encoding and decoding model; (b) Proposes a chromosome architecture that allows of a genome with multiple candidate expressions; (c) Implements MERGE algorithm and gene fitness evaluation algorithm. (d) Gives extensive experiments to show that MERGE outperforms the traditional GEP. Furthermore, When mining complex functions, the success rate of MERGE is 3-5 times of GEP, the average number of generation of successful evolution is 87% higher than GEP , and the average minimum generation of successful evolution of MERGE is reduced to 0.4% of GEP.

1. Introduction

Gene Expression Programming (GEP) was proposed by Ferreira C. in 2000[3], which is derived from Genetic Algorithm(GA) [4, 9] and Genetic Programming(GP)[7]. GEP is widely applied in data mining area such as function finding, classification, association rule, time series prediction, parameter optimization, and digital circuit design etc. In GEP, Genotype (Chromosome) and Phenotype (Expression Tree (ET)) are separated from each orther. Without prior knowledge, GEP automatically evolves over training data and discovers knowledge such as mathematical formulae.

This work was supported by the National Science Foundation of China under Grant No. 60773169 and the 11th Five Years Key Programs for Sci. &Tech. Development of China under grant No. 2006BAI05A01.

Function finding is a typical application of GEP. Traditional GEP always treats multiple genes in a chromosome as one entity. When mining complex functions, multiple genes encode a single expression formula by link functions such as plus, minus, etc.

GEP lacks the power for mining complex functions due to following limitations: (a) independence of gene expression restricts express capability of chromosome and slows down the convergence speed, and (b) one chromosome encodes just one function expression. They restrict search space of GEP, and result in oversized evolution generations.

MEP(Multi Expression Programming) is proposed recently [10, 11]. However, (a)Little redundancy exists in MEP genes, and the population diversity is low, (b)Gene fitness calculation and decoding process are time-consuming for complex function finding, and (c)Evolutions may fall in trap of Local Optimal Solution(LOS).

This study proposes a novel genetic algorithm based on Multi ExpRession GEne programming(*MERGE*). The contributions include: (a) Each gene is evaluated against its selection environment independently, and a chromosome encodes multiple solutions to problems because each gene encodes a solution. (b) Fitness of a chromosome is the best fitness among those of all genes it contains. (c) Propose a new model for hierarchical gene encoding and decoding. After genetic operations, one gene appears zero or more times in its posterior genes. (d) Implement *MERGE* algorithm including genetic operations, encoding and decoding of genes and fitness evaluation of chromosome and gene. (e) Give extensive experimental results and compare *MERGE* with traditional uni-genic GEP and multi-genic GEP. The experimental simulation shows that *MERGE* is an effective and efficient evolutionary algorithm.

The paper is organized as follows. Section 2 describes related works. Section 3 gives several definitions and implementation of *MERGE* algorithm. Section 4 presents fitness evaluation algorithms. Section 5 gives experimental results

978-0-7695-3304-9/08 $25.00 © 2008 IEEE
DOI 10.1109/ICNC.2008.164

and performance analysis. Section 6 concludes the paper.

2. Related work

Function mining is an important task in knowledge discovery. In literature[1, 2], GEP was introduced to function discoveries and performed quite well in many practical situations. Li et al. proposed P-GEP which is a GEP variant[8]. They adopted a new scheme based on prefix notation to construct a hierarchy composition of the solution to complex function finding problems with better performance than traditional GEP. Huang et al.[5, 6] proposed MEM based on GEP to mine complex functions with multiple domains. Ferreira proposed a Boolean function discovering method based on GEP[2]. Peng et al. presented M-GEP which constructs a multi-layer chromosome beyond GEP [12, 13]. Wu et al. solved the problem of recursive function discoveries using GEP[14].

In the above methods, one chromosome is translated into just one expression tree(ET) which can only express an algebraic function or formula. Although each gene encodes an ET and multi-genes chromosome encodes multiple ETs, these ETs have to be linked by a predetermined linking function to obtain one unique expression. Oltean et al. proposed Multi Expression Programming (MEP)[10, 11] where one chromosome encodes multiple expressions. MEP is inspired by computer programming language like C or Pascal, and solved multiple expressions problem with regard to chromosome. Meanwhile, it introduced another problem that a gene in chromosome is always a triple or a terminal symbol, which contains little redundant information that is critical for successful evolutions

3. Definition and *MERGE* Algorithm

The algorithm *MERGE* proposed in this paper is based on multi expression gene programming where each chromosome encode multiple candidate mathematical expressions. Moreover, *MERGE* adopts a different individual fitness evaluation model, encoding and decoding model and genetic operators.

The following example motivates the new model.

Example 1. Let C2 be a 2-genes chromosome in *MERGE*, function set *F* be {+,-,*,/} and terminal set *T* be {a, b}, head length be 5 and tail length be 6 by Equation (1)). Then the two sub-ETs and their corresponding mathematical functions are given in Fig. 1.

Fig. 1(a) describes a chromosome with two genes which locate at position 0 and 1 respectively (note that the position information is not a part of the genotype). Fig. 1(b) gives two ETs encoded in gene 0 and 1, and their mathematical

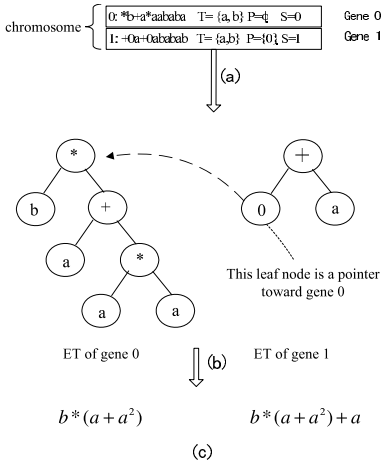

Figure 1. Individual representation in *MERGE*: 2-genes chromosome, two expression trees and corresponding mathematical formulae

function expressions are shown in Fig. 1 (c). Note that the genotype of gene 1 is '+1a+1abababab', and its phenotype, the right ET in Fig. 1(b), is an ET with one pointer toward the ET of gene 0. Therefore, you can obtain the complete ET of gene 1 if you replace gene pointer leaf node 0 with the ET of gene 0. We just use a pointer here to simplify this complexity.

Definition 1 (Gene)**.** *A gene G is a 6-tuple (Q, F, T, S, Fitness, P), Where:*

- *Q is genotype,*
- *F is function set,*
- *T is basic terminal set,*
- *S is gene position where the gene is located in its chromosome,*
- *Fitness is the score of gene according to designated fitness evaluation function,*
- *P is gene pointer terminal set* $\{i|\ i \in [0, S-1]$ *and* $i \in N\}$*, where a pointer, e.g. i points to the gene located at i.*

Definition 1 gives each gene a unique position *S*. If a gene is located at *i*, it is called *i*-th gene or gene *i*. Fig. 1(a) shows two genes i.e. gene 0 and 1 which locate at position 0 and 1 respectively.

There are two terminal sets, i.e. the basic terminal set *T* and the gene pointer terminal set *P*. *P* is very important for hierarchical gene representation in *MERGE*, whereas GEP gene has only basic terminal set *T*.

The gene in *MERGE* also contains two parts: head and tail, and the length of the gene satisfis Equation (1). The head consists of symbols from head set (*HS*, *HS*=*F*∪*T*∪ *P*), and the tail consists of symbols from tail set (*TS*, *TS*=*T*∪

P). After the structure of chromosome is set, *P* of each gene is determined. Obviously, *S* and *P* of one gene are different from those of other genes in a specific chromosome. By Definition 1, *P* of gene 0 is $\emptyset$ and *P* of gene 1 is $\{0\}$, which is illustrated in Fig. 1(a). For example, "+0a+0ababab *T*= {a, b} *P*= $\{0\}$ *S*=1" represents a gene where *Q* is "+0a+0ababab" which is located at position 1.

$$t = h(n-1) + 1 \tag{1}$$

where *t* is the length of tail, *h* is the head length and *n* is the max arity of all functions in function set *F*.

Definition 2 (Chromosome)**.** *A chromosome C is a 3-tuple (U, K, CFitness), where:*

- *U is the set of all genes in the chromosome.*
- *K is the number of genes the chromosome contains;*
- *CFitness is the fitness of this chromosome, which is the best fitness of all its genes and can be calculated through the following Equation (2):*

$$CFitness = Max\{Fitness_j \mid Fitness_j \, is \, the \, fitness \, of \, gene \, j, where \, 0 \leq j \leq K-1.\} \tag{2}$$

The length of a chromosome can be obtained by Equation (3):

$$LC = K[hn + 1] \tag{3}$$

A chromosome in *MERGE* encodes multiple mathematical formulae and all of them are candidate solutions to a problem, which are shown in Fig. 1(c). This is greatly different from that in the traditional GEP, where a chromosome can have just one formula in spite of the number of genes it contains.

Definition 3 (Allele)**.** *Let G1, G2 be two genes from two different chromosomes in the same population, the position S in Definition 1 of Gi be S_{Gi}, If $S_{G1} = S_{G2}$, then G1, G2 are Alleles. If $S_{G1} < S_{G2}$, then G1 is lower position gene of G2 or G2 is higher position gene of G1.*

Lemma 1. *Let G1, G2 be two genes from two different chromosomes in the same population. Let TS_{G1}, TS_{G2} be their union of basic terminal set T and gene pointer set P respectively. Then G1, G2 are alleles, if and only if TS_{G1}= TS_{G2}*

Proof. Let TS_{G1}, TS_{G2} are tail sets of G1 and G2 respectively.If G1, G2 are alleles, they locate at the same position in its chromosomes respectively, i.e. $S_{G1} = S_{G2}$. Thus, they have the same P with each other by Definition 1. Moreover, they share the same T because they belong to two chromosomes in the same population. So they have the same union set of *T* and *P*, i.e. $TS_{G1} = TS_{G2}$. On the other hand, if $TS_{G1} = TS_{G2}$, then G1 and G2 have same *P* because their *T*s are same. Since G1 and G2 come from two genomes in the same population, the same *P* means the same *S* by Definition 1, thus, they are alleles. □

The following Example 2 illustrates the stronger express capability of *MERGE* gene.

Example 2. Let the target function be a^{2^n}, the function set be {+, -, *, /}. To evolve the target function, GEP chromosome requires at least $2^{n+1} - 1$ symbols i.e. 2^n terminal symbols 'a' and $2^n - 1$ function symbols '*'. However, *MERGE* only requires 3n symbols. For example, chromosomes from GEP and *MERGE* that encode a^8 are given below:

GEP: C=*******aaaaaaaa
MERGE: 0:*a, a
1:*0, 0
2:*1, 1

The most important thing is that a GEP chromosome expresses just one function a^8, whereas, a *MERGE* chromosome can express multiple formulae i.e. a^8, a^4, and a^2. Therefore, for the same chromosome length, the express capability of *MERGE* chromosome greatly exceeds that of GEP and *MERGE* has much stronger search ability.

The key ideas of *MERGE* algorithm are: (a) encoding more powerful genes through lower position genes, (b) each gene is evaluated respectively and participates in selection operation, and (c) the best gene is chosen to represent its chromosome. The evolution process is illustrated in Algorithm 1.

Algorithm 1 *MERGE* Evolution Process

1: **INPUT:** configuration and training data
2: **OUTPUT:** one individual with best fitness
3: load training data and initial configuration
4: randomly create an initial population
5: **for** i = 0 to $Max\_Generation$ **do**
6: decode each chromosome into multiple ETs
7: calculate each chromosome's fitness which can be obtained through calculating its genes' fitness.
8: select individuals and generate next generation
9: apply genetic operations sequentially on the new generation
10: randomly extract h symbols from HS to generate gene head
11: randomly extract t symbols from TS to generate gene tail, where t is calculated by Equation (1)
12: **end for**
13: **return** an individual with best fitness

The algorithm-*MERGE* is similar to the algorithms of GA[4, 9], GP[7], and GEP[3] in evolution process; it repeats the process of selection—genetic operation—fitness evaluation—new population until the stop condition is satisfied. *MERGE* adopts a different policy to evaluate the fitness of an individual. In *MERGE*, each chromosome contains multiple genes where each gene can be decoded into

an independent ET. Thus we first calculate the fitness of each gene in the chromosome, and then set the individual's fitness as the maximum of its genes' fitness. By this way, the best gene encoding the best solution to a problem can be find out and kept during evolution.

4. Fitness Evaluation of Individual

MERGE calculates all genes' fitness of each chromosome in population, and then sets the individual's fitness as the max fitness of its genes. In *MERGE*, we use relative error evaluation Equation (4)[2] to evaluate the fitness of a gene.

$$F_e(E_i) = \sum_{j=1}^{\mathcal{T}} \left(R - \left| \frac{P_{ij} - T_j}{T_j} \cdot 100 \right| \right) \quad (4)$$

In Equation (4), *R* is selection range, P_{ij} is the model value of *i*-th gene in the chromosome over the *j*-th training data, T_j is the target value of the *j*-th training data, and $F_e(E_i)$ is the fitness value of the *i*-th gene, and $\mathcal{T}$ is the number of training data.

After calculating each gene's fitness, we set the fitness of the chromosome as the maximum of these genes' fitness. Equation (5) is used to calculate an individual fitness.

$$F(C) = Max\{F_e(E_i)\}, 0 \leq i \leq K-1 \quad (5)$$

C is chromosome representing an individual, and $F(C)$ is the fitness of the chromosome over all training data, and K is the number of genes.

The Algorithm 2 is used to calculate individual's fitness and its genes' fitness.

Algorithm 2 *MERGE*_GIFitness

```
1: INPUT: Training Set and Individual C
2: OUTPUT: Individual's Fitness
3: load training data and initialize two arrays: modelV and
   fitnessofChromo.
4: decode individual C into multipule ETs
5: for all gene G in C do
6:    for all data D in training data do
7:       calculate the model value of G over D
8:       store the model value into array modelV
9:    end for
10:   calculate gene G's fitness by Equation (4) and store
      it to array fitnessofChromo
11: end for
12: return the maximum fitness from fitnessofChromo
```

5. Experiment and Performance Analysis

MERGE is implemented based on Java SDK 1.5, Eclipse 3.2 and Windows XP.

Experiment 1. Problem of function finding. The function is as follows:

$$f(a) = a^6 - 2 * a^4 + a^2 \quad (6)$$

Randomly generate 20 sample data according to Equation (6) in the range of [0, 3]. Parameters are listed in Table 1. Run each algorithm 100 times.

Table 1. Parameter settings*

Parameters	values
Function Set	+,-,*/
Basic Terminal Set	{a }
Number of Generations	5000
Population Size	100
Selection Range	1000
Head Length	7
Number of Genes	4
Length of Chromosome	60
IR,IS,RIS	0.1
Length of Insertion Sequence	{1,2,3}
One-point Recombination Rate	0.3
Two-point Recombination Rate	0.3
Gene Recombination Rate	0.1
Precision	0.01

The experimental results are given in Table 2.

Table 2. Performance comparison*

	Suc	Avg	Max	Min
MERGE	100%	387	1008	3
Single-Gene GEP	84%	1290	4812	24
Multi-Genes GEP	22%	2909	4840	743

*"Suc" means the success rate. "Avg" means the average number of generation of success evolution. "Max" means the max number of generation of success evolution and "Min" is the min number of generation of success evolution.

Table 2 shows that *MERGE* outperforms singe-gene GEP and multi-genes GEP on function mining and its success rate is 100% which is 19% and 355% higher than single-gene GEP and multi-genes GEP respectively. The average number of generation of success evolutions is 70% and 87% higher than singe-gene GEP and multi-genes GEP respectively, and the minimum number of generation of success evolutions is as much 12.5% and 0.4% as singe-gene GEP and multi-genes GEP.

Experiment 2. Test the influence of head length on success rate.

The training data are generated in the range [0, 20] by target function $f(x) = a^4+a^3+a^2+a$. Run the experiment 100 times, and the results are shown in Fig. 2.

The evolution parameters are partially listed in Table 3, and other parameters were same to those in Table 1.

Table 3. Parameter settings

Parameters	values
Number of Generations	50
Population Size	30
Selection Range	100
Number of Genes	6

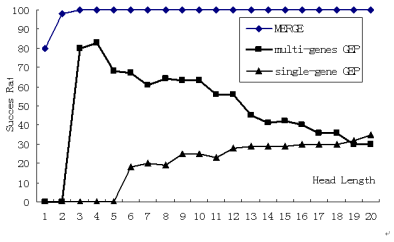

Figure 2. Variation of success rate with head length in *MERGE*, single-gene GEP and multi-genes GEP

The experiment shows:

(a) *MERGE* required less head length when the parameters setting and target function were same. Furthermore, *MERGE* reached the success rate of 83% even though the head length was 1 or 2. However, GEP will fail.

(b) *MERGE* can easily arrive at the success rate of 100% when head length was 3 and it never went down after that. Although multi-genes GEP could achieve at the maximum success rate of 83% when head length was 4, it would go down when the head length was longer.

(c) Single-gene GEP never succeeded in evolution when head length was less than 5, but when the head length was longer; the success rate also went up steadily.

6. Conclusions

We propose a novel genetic evolutionary algorithm - *MERGE* featured with multi expression chromosome. The main contributions include: (a)Presents a novel hierarchical gene representation model including encoding and decoding model.(b)Puts forward a novel chromosome encoding scheme that allows of a genome with multiple candidate expressions.(c)Implements algorithms including *MERGE*, gene encoding, gene decoding, and gene fitness evaluation algorithms.(d)Gives extensive experimental results to show that *MERGE* outperforms single-gene GEP and multi-genes GEP.

References

[1] C. Ferreira. Gene expression programming: A new adaptive algorithm for solving problems. *Complex Systems*, 13(2):87–129, 2001.

[2] C. Ferreira. Discovery of the boolean functions to the best density-classification rules using gene expression programming. In E. Lutton, J. A. Foster, J. Miller, C. Ryan, and A. G. B. Tettamanzi, editors, *Proceedings of the 4th European Conference on Genetic Programming, EuroGP 2002*, volume 2278 of Lecture Notes in Computer Science, pages 51–60. Springer-Verlag,Berlin, Germany, 2002.

[3] C. Ferreira. *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*. Springer-Verlag, Berlin, Germany, 2 edition, 2006.

[4] D. E. Goldberg. *Genetic Algorithms in search, Optimization and Machine learning*. Addison-Wesley, 1989.

[5] X. D. Huang and C. J. Tang. A gene expression programming based function discovery method. *Computer Science*, 30:278–282, 2003.

[6] X. D. Huang, C. J. Tang, Z. Li, D. H. Pu, and Y. Liao. Mining functions relationship based on gene expression programming. *Journal of Software*, 15(suppl.):96 105, 2004.

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

[8] X. Li, C. Zhou, W. M. Xiao, and P. C. Nelson. Prefix gene expression programming. In *Late Breaking Paper at Genetic and Evolutionary Computation Conference, GECCO-2005*, Washington, D.C., USA, 2005.

[9] M. Mitchell. *an Introduction to Genetic Algorithms*. MIT Press, 1996.

[10] M. Oltean and C. Grosan. Evolving evolutionary algorithms using multi expression programming. In B. W, editor, *the 7th European Conference on Artificial Life*, volume 2801, pages 651–658, Dortmund, 2003. LNAI , Springer-Verlag, Berlin.

[11] M. Oltean and C. Grosan. Evolving digital circuits using multi expression programming. In R. Zebulum, D. Gwaltney, G. Horbny, D. Keymeulen, J. Lohn, and A. Stoica, editors, *NASA/DoD Conference on Evolvable Hardware*, pages 87–90, Seatle, 2004. IEEE Press, NJ.

[12] J. Peng, C. J. Tang, C. Li, and J. J. Hu. M-gep: A new evolution algorithm based on multi-layer chromosomes gene expression programming. *Chinese Journal of Computer*, 28(9):1459–1466, 2005.

[13] J. Peng, C. J. Tang, C. Li, and J. J. Hu. A new evolutionary algorithm based on chromosome hierarchy. *NETWORK International Journal of Computers and Applications*, 30(2):1–9, 2008.

[14] J. Wu, C. J. Tang, Y. Jiang, S. Y. Ye, L. Duan, and T. Y. Li. Mining recursive functions based on gene expression programming", based on gene expression programming. *Journal of Sichuan University (Natural Science Edition)*, 39(5):127–131, 2005.