Chapter 49
Multi-Expression Based Gene Expression Programming

Wei Deng, Pei He and Zhi Huang

Abstract Among the variants of GP, GEP stands out for its simplicity of encoding method and MEP catches our attention for its multi-expression capability. In this paper, a novel GP variant-MGEP (Multi-expression based Gene Expression Programming) is proposed to combine these two approaches. The new method preserves the GEP structure, however unlike the traditional GEP, its genes, like those of MEP, can be disassembled into many expressions. Therefore in MGEP, the traditional GEP gene can contain multiple solutions for a problem. The experimental result shows the MGEP is more effective than the traditional GEP and MEP in solving problems.

Keywords Genetic programming · Gene expression programming · Multi expression programming

49.1 Introduction

Genetic programming is a new branch of the evolutionary algorithm, early GP is organized with structures of tree-based, and operated directly on the tree, which leads to the low efficiency of function mining. To solve this problem, linear genetic encoding and graphic genetic encoding methods have been proposed, such as linear genetic programming (LGP), multi expression programming (MEP) and

W. Deng (✉) · P. He · Z. Huang
School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, China
e-mail: 76995958@qq.com

P. He
Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China

Z. Sun and Z. Deng (eds.), *Proceedings of 2013 Chinese Intelligent Automation Conference*, Lecture Notes in Electrical Engineering 256,
DOI: 10.1007/978-3-642-38466-0_49,

gene expression programming (GEP). These algorithms nowadays have been applied in a wide range of real-word problems.

Genetic Expression Programming (GEP) algorithm is a new member of the genetic family, which is widely used in knowledge discovery. GEP integrating the GA and GP advantages uses simple encoding method to solve complex problems. GEP still exist some problems that evolutionary generation is too large or unable to obtain optimal results. MEP (Multi Expression Programming) is a GP variant. A unique MEP feature is its ability of encoding multiple solutions for a problem [1]. But the complexity of the MEP decoding process is higher than the other linear genetic programming. In this paper, we propose a novel algorithm MGEP which combines GEP algorithm with MEP (Multi Expression Programming). MGEP employs multi expression features used in GEP. The improved algorithm can achieve higher efficiency of function mining.

This paper realized a new algorithm and related genetic operator, and analyzed the advantages of MGEP algorithm in the usage of expression space compared with the traditional GEP. The algorithm achieved good results for solving specific problems. The Experimental results show that when compared with the standard GEP, the average evolution generation of MGEP is 1.1–5.3 % of the traditional GEP and 7–18 % of MEP.

The paper is organized as follows. The backgrounds of the GEP and the MEP are presented in Sect. 49.2. The MGEP algorithm is described in Sect. 49.3. In Sect. 49.4, we will perform several numerical experiments to compare the MGEP, the GEP and the MEP algorithm. Section 49.5 concludes the paper.

49.2 Background

49.2.1 MEP and GEP

GEP and MEP are automated method for creating computer programs. GEP as a linear chromosome is composed of genes containing terminal and function symbols. Search operators are operated on a GEP gene to produce new individuals, such as crossover, mutation and transposition. MEP chromosomes are composed of the number of the genes which are substrings of variable length. A unique MEP feature is the ability of choosing the best gene provide the output for the chromosome. This is different from other GP techniques which employed a fixed gene for output. In MEP the offspring obtained by crossover and mutation. These two techniques of the GP variant have been applied in various fields.

GEP Representation: A GEP gene is composed of head and tail. The head contains function or terminal symbols, but the tail can only contain terminal symbols. For each problem, the length of the head (denoted h) is chosen by the user, and the length of the tail (denoted t) is evaluated by formula 49.1, where n is the maximum number of the function arguments in the function set.

$$t = h * (n - 1) + 1 \tag{49.1}$$

Let us consider a set of function symbols $F = \{+, *, /, S\}$ and a set of terminal symbols $T = \{a, b\}$, where the symbol S stands for the sin operator. In this case $n = 2$. If we choose $h = 7$, then we get $t = 8$, so the length of the gene is $7 + 8 = 15$. Such a gene is given below: $C = +/*aSb + bbababab$. GEP is encoded by a fixed length string called K-expression, and the individual phenotype is an expression tree (ET). A K-expression can be translated into an expression tree (ET) by breadth-first parsing. The ET based on the string C is created in Fig. 49.1. The expression encoded by the gene C is: $E_1 = a/\sin(b) + b * (b + a)$. The last five elements of this gene are not used. Usually, a GEP gene is not entirely used for phenotypic transcription.

MEP Representation: A MEP gene encodes a terminal or a function symbol. If a gene contains a function symbol, its arguments must point to gene indices of lower values than the position of the function itself. And the first symbol of the chromosome must be a terminal symbol.

Consider the set of functions: $F = \{*, +, S\}$, and the set of terminals $T = \{a, b\}$. The numbers on the left stand for gene index and the right is the encoded gene. An example of a chromosome C using the sets F and T is given in Fig. 49.2a. Translation of the MEP chromosome into expressions is done top-down. A terminal symbol decodes a simple expression. A function symbol translates into a complex expression by connecting the operands with the gene indices which would be substituted for the decoded expression. The chromosome in the previous example would be decoded into expressions in Fig. 49.2b.

49.2.2 MEP and GEP Related Literature

GEP as a new genetic algorithm shows its superior performance in many fields, especially the outstanding performance in the area of data mining. Nowadays, more and more researches are focusing on GEP all over the world, mainly including the basic principle of GEP and its application.

Some foreign literature reviewed genetic programming [2–5]. We are reviewing the main GP variant with linear representation in paper [2], which provides a

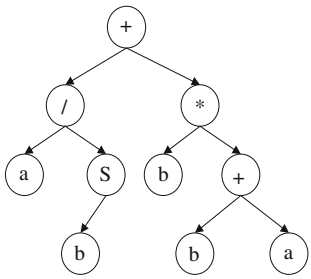

Fig. 49.1 An expression tree encoding the function $a/\sin(b) + b * (b + a)$

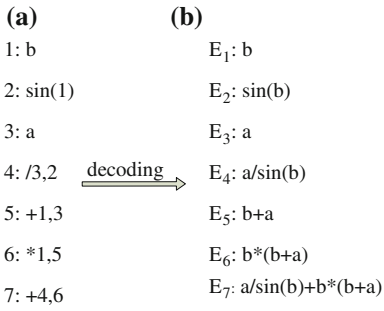

Fig. 49.2 MEP encoding and decoding way of the gene C **a**. **b**

complete description for each method. In paper [3] the author outlines some of the challenges and open issues that face researchers and practitioners of GP. The first article of GEP was published by Portugal Candida Ferreira in 2001. Then she wrote many papers and published GEP monographs [6–9]. In paper [9] a hybrid evolutionary technique is proposed for data mining tasks, which combines the Clone Selection Principle with GEP.

MEP has a flexible encoding method and the ability to provide abundant representations. MEP was reviewed systematically in literature [9–11]. All aspects of the MEP are summarized in paper [9]. MEP is widely used in many fields [12, 13]. In paper [12] authors proposed a way of structuring a CGP algorithm to make use of the multiple phenotypes which are implicitly encoded in a genome string.

49.2.3 MEP and GEP Weakness

Generally,the more genes in a GEP chromosome,the higher sucess rate of GEP. However, if the genes are over a certain value, the success rate will decrease. This situation occurs in that GEP uses a complex chromosome to encode a less complex expression. If the target expression is short and the head length is large, the majority of individual is unused. This problem usually arises in evolutionary process when GEP employs chromosome with a fixed length.

There are problems where the complexity of the MEP decoding process is higher than the complexity of the GE, GEP, and LGP decoding processes. This situation usually arises when the set of training data is not a priori known [2].

Consider a GEP gene, as long as we read a K expression from the different head symbols, we can build more expression trees. Owing to the breadth and depth of the same symbols in other expression tree are different, thus the corresponding expressions are not only decomposed by standard GEP expression. Based on this idea, we proposed a novel technique MGEP which combines the advantages of the GEP with the MEP features. The new algorithm would build up more expression trees for a GEP gene. And compared with traditional MEP, MGEP has a diversity of the gene. The MGEP with these two characteristics would overcome the weakness of MEP and GEP.

49.3 MGEP Model

49.3.1 MGEP Initialization

MGEP initialization is the same as the GEP. MGEP chromosomes are initialized randomly, but they must obey two rules that a gene contains two parts of the symbols and the tail of the gene must be composed of terminal symbols.

49.3.2 MGEP Representation

The evolutionary process of the MGEP algorithm is similar to the traditional genetic algorithm, namely: initial population → selection → crossover → variation → new population → selection. MGEP encoding way is identical with the standard GEP. But the novel algorithm would decode into multiple sub-expressions by a K-expression. MGEP algorithm as the standard GEP algorithm starts by creating a random population of individuals. To realize our aim, we start by reading the K-expression from the first head character to build the first expression tree, which is totally the same with the GEP. Then, we continue reading the K-expression from the second head character to establish another expression tree. Repeat reading the head characters until the end of the head. Don't consider the single node of the sub-expression tree which called the dead gene.

A MGEP gene contains multiple solutions for a problem. Compared with the MEP algorithm, MGEP have more diverse individuals in the population. Because several MEP genes are disjoined or combined by the sub-expressions, but MGEP gene can decode into diversity of genes. Consider the referred gene C in the second section. We utilize the MGEP algorithm to build multiple expression trees based on the gene $C = +/*aSb+bbababab$. The decomposition of the first sub-expression tree is shown in Fig. 49.1, and more sub-expression trees by reading K-expression from the second head character beginning to the last are shown in Fig. 49.3.

As shown in Figs. 49.1 and 49.3, the gene C decomposed into five sub-expressions. The number of sub expressions is less than the head length of the gene, because the MGEP algorithm doesn't build an expression tree for a single node. Sub-expressions in Fig. 49.3 are not obtained only by separating the expression 49.1, it produced many novel expressions. The reason is that the character of the depth and breadth in the new sub expression trees are different. The tail length of the gene is larger than the required length, so the sub-expression is valid.

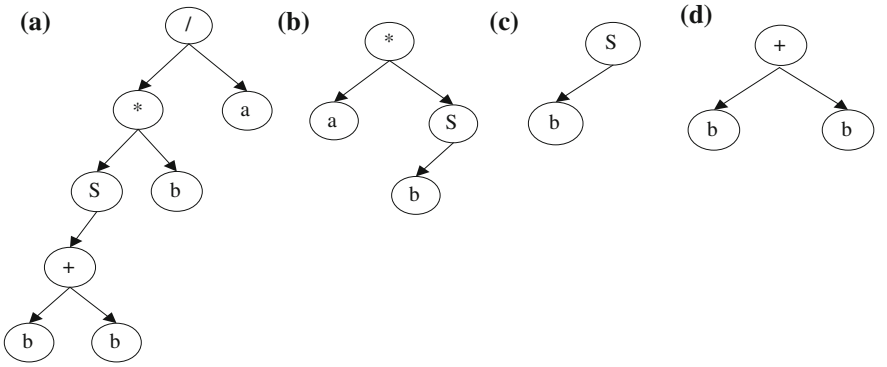

Fig. 49.3 A K-expression translates into five sub-expressions. **a** $E_2 = [b * \sin(b + b)]/a$. **b** $E_3 = a * \sin(b)$. **c** $E_4 = \sin(b)$. **d** $E_5 = b + b$

49.3.3 MGEP Fitness Assignment

The number of the sub-expressions of a gene is less than the head length of the gene. The MGEP gene fitness is defined by the best sub-expression. This fitness assignment idea is identical with the MEP algorithm. For instance, if we solve a symbolic regression problem, the fitness of each sub-expression E_i may be computed using the formula:

$$f(E_i) = \sum_{j=1}^{N} |O_{j,i} - w_j| \tag{49.2}$$

where $O_{j,i}$ is the value of the sub-expression E_i of the gene on the jth sample data and W_j is the corresponding target result. The fitness of the gene $f(G)$ is equal to the lowest fitness of the sub-expression in a single gene.

$$f(G) = \min f(E_i) \tag{49.3}$$

49.3.4 MGEP Genetic Operators

The MGEP technique uses the same search operators as the GEP, such as crossover, mutation and transposition to obtain new individuals. In crossover, two parent genes are randomly chosen and paired to exchange some material between them. Mutation can occur anywhere in the gene, but the structural organization of the gene should be preserved in this genetic operator. In MGEP, there are also three kinds of recombination: one-point, two-point and gene recombination. The genetic operation of MGEP is also operated on K-expression. So this novel technology does not damage the structure of the classical GEP.

49.3.5 MGEP Algorithm

In this section, we introduced the MGEP algorithm. The new algorithm only improved GEP decoding way and fitness assignment of a gene. A K-expression is decoded into several expression trees. So a gene of the MGEP contains multiple solutions for a problem. Combined with MEP idea the best solution is chosen for the MGEP gene. The MGEP algorithm is described in MATLAB language as follows:

```
function gene_fit = Fitness_Assignment(k_exp,T,h,t)
% Input arguments: a K-expression, terminal symbols, head length, tail length
% Output arguments: fitness value of the gene
% Create_ET is a another function which established an expression tree based on
% a sub-expression
% The Traverse function is to traverse an expression tree
% The Evaluate function is to assess a gene
for i = 1:h
   sub_k = []; sub_ET = []; sub_expression = [];   %Initialize variables
   if(~isempty(findstr(k_expression(i),T))   % Death gene is not taken into account
         sub_k = k_expression(i:h+t);              % Decomposition of a K-expression
         sub_ET = Create_ET(sub_k);                % Create an sub-expression tree
         sub_expression = Traverse(sub_ET);     % Traverse the sub expression tree
         sub_fit(i) = Evaluate(sub_expression);    % Evaluate the sub expression
     end
end
gene_fit = min(sub_fit);
```

49.4 Experiments and Analysis

We conducted two experiments, and compared the MGEP evolutionary performance with the MEP and the GEP program. Three genetic algorithms were set with the same evolution parameters. The length of a MEP chromosome was set to be equal to the head length of a GEP gene. The number of MEP genes was more than the MGEP's. So we can compare their performance in the same environment. The main parameters are given in Table 49.1.

Experiment: Two symbol regression problems, as the following formula:

$$y = x^4 + x^3 + x \tag{49.4}$$

$$z = x^3 + x^2 y + y \tag{49.5}$$

Based on the fourth problem, the terminal symbols would be set T = {x}, and the other function is set to T = {x, y}. In this two problems, we choose h = 10. For the sake of fairness, the length of a MEP chromosome takes the value 10. So

Table 49.1 Two symbolic regression problems of experimental parameters

Function set	Number of chromosomes	Number of training data	Maximum generation	Crossover probability	Mutation probability	Accuracy
{+ ,*}	50	15	500	0.9	0.5	0.000001

MGEP and MEP algorithm includes approximately the same number of genes. Other parameters are given in Table 49.1. For two problems, each algorithm ran 50 times, and the comparison performance of three kinds of algorithm is shown in Figs. 49.4 and 49.5 respectively, the specific result in the following Tables 49.2 and 49.3.

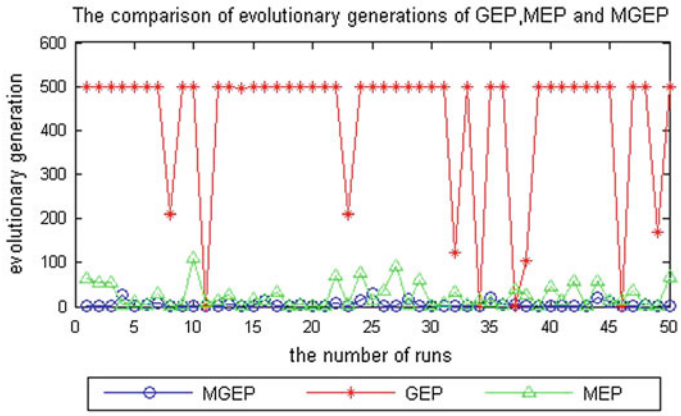

Fig. 49.4 The evolutionary generation of the expression $y = x^4 + x^3 + x$

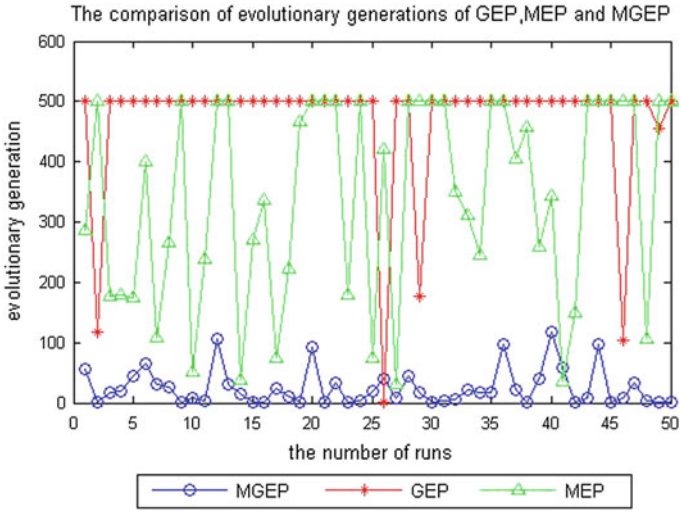

Fig. 49.5 The evolutionary generation of the expression $z = x^3 + x^2y + y$

Table 49.2 Comparison evolutionary results of the three kinds of algorithm for the problem $y = x^4 + x^3 + x$

Algorithm	Average generation	The longest generation	The shortest generation	Probability of success (%)
MGEP	4.68	28	1	100
Standard GEP	426.46	497	1	20
Standard MEP	25.64	110	1	100

Table 49.3 Comparison evolutionary results of the three kinds of algorithm for the problem $z = x^3 + x^2y + y$

Algorithm	Average generation	The longest generation	The shortest generation	Probability of success (%)
MGEP	25.46	117	1	100
Standard GEP	467.06	455	1	10
Standard MEP	343.12	466	31	58

The results of Experiment 1 showed that, in function mining with one variable, MGEP algorithm was obviously better than the standard GEP and MEP algorithm. The average number of generation to success of the MGEP is only 1.1 % of the GEP and 18 % of the MEP, and the success probability of function mining was significantly higher than the standard GEP and MEP algorithm.

In function mining with two arguments, the average number of generation to success of MGEP was about only 5.3 % of the GEP and 7 % of the MEP. These two experimental results showed the evolution efficiency of the MGEP algorithm was much higher than the standard GEP and MEP algorithm. Because the novel algorithm has the MEP unique feature which can store multiple solutions for a problem in a single gene.

49.5 Conclusion

This paper proposed a new genetic algorithm MGEP. And we introduced the fitness assignment and genetic operators for the MGEP. The algorithm integrates GEP and MEP idea which contains several sub-expressions. Experiments showed that when compared with the traditional GEP and the MEP, MGEP algorithm significantly improved the evolutionary performance.

Acknowledgments This work was supported by the National Natural Science Foundation of China (Grant No. 61170199), Hunan Provincial Innovation Foundation for Postgraduate (CX2012B367), Guangxi Key Laboratory of Trusted Software (Guilin University of Electronic Technology), and the Scientific Research Fund of Education Department of Hunan Province, China (Grant No. 11A004).

References

1. Oltean M, Grosan C (2004) Evolving digital circuits using multi- expression programming. In: Proceedings of NASA/DoD conference on evolvable hardware. IEEE Press, pp 87–94
2. Oltean M, Grosan C, Diosan L, Mihaila C (2008) Genetic programming with linear representation a survey, WSPC/INSTRUCTION FILE
3. O'Nell M, Vanneschi L, Gustafson S, Banzhaf W (2010) Open issues in genetic programming. Genet Program Evolvable Mach 11:339–363
4. Oltean M, Grosan C (2003) A comparison of several linear genetic programming techniques. Complex Syst 14:285–313
5. He P, Kang L, Johnson CG, Ying S (2011) Hoare logic-based genetic programming. Sci Ch Inf Sci 54(3):623–637
6. Ferreira C (2002) Gene expression programming, 1st edn. Angra do Heroismo, Portugal
7. Ferreira C (2004) Gene expression programming and the evolution of computer programs. In: de Castro LN, Von Zuben FJ (eds) Recent developments in biologically inspired computing, vol 5. Idea Group Publishing, New York, pp 82–103
8. He P, Johnson CG, Wang HF (2011) Modeling grammatical evolution by automaton. Sci Ch Inf Sci 54(12):2544–2553
9. Karakasis VK, Stafylopatis A (2006) Data mining based on gene expression programming and clonal selection. IEEE congress on evolutionary computation, Vancouver, Canada, pp 514–521
10. Tsakonas A (2006) A comparision of classification accuracy of four genetic programming-evolved intelligent structures. Inf Sci 176:691–724
11. Cattani PT, Johnson CG (2010) ME-CGP: multi expression cartesian genetic programming. IEEE Congr Evolut Comput 2010:1–6
12. Yanan W, Bo Y, Zhao X (2009) Countour registration based on multi-expression programming and the improved ICP. IEEE, 2009
13. Chen Y, Jia G, Xiu L (2008) Design of flexible neural trees using multi expression programming. In: Proceedings of Chinese control and decision conference, vol 1, pp 1429–1434