Applied Mechanics and Materials Vols. 411-414 (2013) pp 2067-2073
© (2013) Trans Tech Publications, Switzerland
doi:10.4028/www.scientific.net/AMM.411-414.2067

Multi-Subexpression Programming

Longbin Chen [1, a], Pei He[2,b]
[1] School of Computer and Communication Engineering. Changsha University of Science and Technology, Changsha 410114, China
[2] Key Laboratory of High Confidence Software Technologies(Peking University), Ministry of Education, Beijing, 100871, China
122790834@qq.com, bk_he@126.com

Keywords: GEP; MEP; subexpression; average fitness value; success rate; evolutionary efficiency

Abstract. Gene Expression Programming is a new and adaptive brand evolution algorithm which is developed on the basis of genetic algorithm. In recent years, Multi-Expression Programming which is proposed in the genetic programming is a linear structure coding scheme,its main feature is a chromosome contains multiple expressions. The idea of MEP is introduced into the GEP in this paper, so a single GEP gene contains multiple solutions to solve the problem.The new algorithm analyzes each gene in the GEP to extract relational subexpressions, then fitness evaluate certain subexpressions to choose the best fitness as individual's fitness, and carry on related genetic manipulation. Finally, the improved algorithm experiment with GEP and MEP, compare their mining the same function's ability,record average fitness value and success rate. The experiment results show that the improved algorithm has better evolutionary efficiency.

1 Introduction

Gene Expression Programming(GEP) is a new type of random search and optimization algorithm that simulate natural selection and genetic evolution of the biosphere, combined with genetic algorithm (GA) and genetic programming (GP).In form, GEP and GA are branch of evolutionary computation, but the individual coding and phenotype use different design ideas. In GEP, the individual in the population is encoded into a fixed-length linear string of symbols (chromosome), then during the fitness evaluation, individual is represented as the difference in the shape and size of the expression tree. GEP combines the advantages of both genetic algorithm and genetic programming, by transforming between genotype (chromosome) and phenotype (expression tree) in the process of genetic evolution to solve the problem, this approach makes the GEP efficient in solving complex problems [1-7].

Due to traditional genetic programming using nonlinear encoding tree structure, so it's too complex and lower efficiency to deal with tree structure. To address this issue, the linear structure coding schemes have been proposed. The Romanian scholar Oltean proposed a new type of linear coding genetic programming method MEP in 2003. Compared with traditional genetic programming, the main feature of MEP is within a chromosome that contains a plurality of expressions wherein each expression may be selected to represent the chromosomes. This makes MEP chromosome can also have a good adaptability even the complexity of the target expression is unknown [8-10].

This paper combines the advantages and disadvantages of above two genetic algorithms, proposed the idea of Multi-Subexpression Programming. The feature of this algorithm is to use the GEP encoding and decoding methods, then extract subexpressions of gene, choose the best fitness of subexpression as individual's fitness. The algorithm combines the advantages that GEP encoding simple and MEP contains multiple expressions, introduces the idea of MEP to GEP in order to improve the ability of GEP to solve complex problems. At last from evolutionary efficiency aspect analyze the advantages of this algorithm compared with the traditional GEP and MEP.

2 MEP Algorithm

In MEP each gene encodes a terminal or a function whose arguments always have indices of lower values than the position of that function in the chromosome. In the selection process, the classical MEP will decode each gene in the population and assign it a fitness value according to how well it solves the problem.Usually the best solution is chosen for fitness assignment of the chromosome. Create new individuals based on encoded genes by crossover and mutation.

Standard MEP algorithm starts by creating a random population of individuals.The following steps are repeated until a problem is solved.Two parents are selected in a selection procedure.Then the parents are recombined in order to generate new offspring by crossover. Finally the offspring are considered for mutation[11-13].

The main feature of MEP is within a chromosome that contains a plurality of expressions wherein each expression may be selected to represent the chromosome, so even the complexity of the target expression is high or unknown, the chromosome can also have a good adaptability. It is precisely because of this characteristic of the MEP, we introduce it to the GEP and proposed Multi-Subexpression Programming, by improving ability of GEP to fitness evaluate the complex individual, thereby improving the evolution efficiency of algorithm. In section 3, we will describe Multi-Subexpression Programming.

3 Multi-Subexpression Programming

3.1 Gene Structure

Because the GEP encoding simple, so the proposed algorithm use the gene structure similar to GEP. The basic unit of algorithm is chromosome; the chromosome consists of one or more genes. Gene composes of linear fixed-length strings. The gene structure is shown in table 1. Each gene consists of the head and tail, the head can contain function symbol and terminal symbol, and the tail can only contain terminal symbol, encoded gene called k expression. This encoding method can ensure that the syntactically correct expression tree is constructed. The length of the head h is set in advance according to the specific issues, the length of the tail by the following formula:

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

Where n represents the function symbols need the largest number of variable. The total length of the gene is equal to the sum of the length of the head and tail.

Table 1. Gene Structure

Gene1		Gene 2		…	Gene *n*	
head	tail	head	tail	…	head	tail

For example, let $F = \{+,*,/,S\}$ be the set of function symbols where the symbol S represents the sin function, and the terminal set be $T = \{x, y\}$. From F we can know n is 2,and suppose h=7,so we can calculate t=8, the total length of the gene is h+t=15.According to above conditions generate a gene C in population,like $C = /+*xSy+yyxyxyyx$.

3.2 Fitness evaluation

Fitness evaluation is a very important step of GEP; it affects the evolutionary efficiency of GEP. Each individual in the population is assigned a fitness value according to how well it solves the problem. This paper introduces the idea of MEP to GEP by extracting subexpression, then use the test sample set to calculate the fitness value of each subexpression and choose the best fitness as individual's fitness. This makes chromosome can also have a good adaptability even the complexity of the target expression is unknown.

Before fitness evaluation we need to extract subexpression first .The algorithm adopt traversal expression tree to extract subexpression, The specific algorithm is described as follows:

First of all, according to the k expression which generated from the GEP encoding rules construct corresponding expression tree.

Secondly, for the expression tree using traversal method to extract subexpression, so the expression tree can be translated into postfix expression. Then scan the postfix expression from left to right, using stack can get all subexpressions, each subexpression need to use parentheses to distinguish.

The proposed algorithm is described in MATLAB language as follows:

```
function [sub_exp] = suffix_to_infix( suffix_exp ) %Input postfix expression
                                                   %Output Subexpression
temp=[];suffix_str = [];
index = 1;
for i=1:length(suffix_exp)                    %Scan the postfix expression from left to right
if(suffix_exp(i) == ' ')
suffix_str{index} = temp;
index = index + 1;
temp=[];
else
temp=[temp,suffix_exp(i)];
    end
end
suffix_str{index} = temp;
temp_exp = suffix_str;
sub_exp = [];
index = 1;k = 1;
while(index <= length(temp_exp))
if(findstr(temp_exp{index},two_operator)  % Determine the current operator is two_operator
flag = zeros(1,2);
j = 1;
for i = index-1:-1:1
if(~isempty(temp_exp{i}))
flag(j) = i;
j = j + 1;
if(j == 3)
break;
    end
  end
end
sub_exp{k} = ['(',temp_exp{flag(2)},temp_exp{index},temp_exp{flag(1)},')'];
temp_exp{flag(1)} = [];temp_exp{flag(2)} = [];
temp_exp{index} = sub_exp{k};
 k = k + 1;
end
if(findstr(temp_exp{index},single_operator))
                                           % Determine the current operator is single _operator
flag(1) = 0;
for i = index-1:-1:1
if(~isempty(temp_exp{i}))
flag(1) = i;
break;
```

```
    end
  end
  sub_exp{k} = [temp_exp{index},'(',temp_exp{flag(1)},')'];
  temp_exp{flag(1)} = [];
  end
  index = index + 1;
  end
  sub_exp = sub_exp';                              %Output all subexpressions
```

For example, let $F = \{+,*,/,S\}$ be the set of function symbols where the symbol S represents the sin function, and the terminal set be $T = \{x, y\}$ .n=2,h=7, generate a gene C in population, $C = +*-x/y+xyxyyxxy$. According to C can get corresponding expression tree.

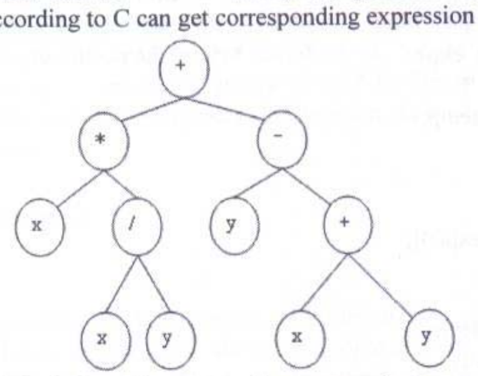

Fig.1. Gene C corresponding expression tree

After traversing the expression tree can get postfix expression $xxy/*yxy+-+$, according to this can extract all subexpressions as follows,
(x/y), $(x*(x/y))$, $(x+y)$, $(y-(x+y))$, $((x*(x/y))+(y-(x+y)))$.

Fitness evaluation is required after extracting the subexpression. The different fitness function has a different assessment effect for the gene. In this paper, fitness functions are given below that called absolute error:

$$fitness(E_i) = \sum_{j=1}^{N} \left| O_{j,i} - w_j \right| \tag{2}$$

Where $O_{j,i}$ is the value of subexpression E_i of gene on the jth sample data and Wj is the corresponding target result, N represents the number of samples. Due to the number of subexpressions extracted from each individual are different,so i is dynamic.The fitness of the gene fitness(G) is equal to the lowest fitness value of the subexpression in a single gene. As a result of the original fitness,the fitness value is lower, the gene's fitness is better.

$$fitness(G) = \min f_i(E_i) \tag{3}$$

4 Experiments and Analysis

In this section, several experiments with comparisons of Multi-Subexpression Programming and other two standard algorithms are performed on two regression problems. The two functions to be examined are $y = x^4 + x^3 + x$ and $z = x^3 + x^2 y + y$.Respectively, we will compare the performance of these three methods in average fitness value and success rate by running them on the same problems based on the same 15 samples data for 50 times.

Table 2. The experimental parameters of the two regression problems

Function Set	$F=\{+,*\}$
Terminal Set	$T=\{x\}, T=\{x,y\}$
Number of Chromosomes	20
Number of Generations	500
Crossover Probability	0.9
Mutation Probability	0.5
Gene Head Length	10

Figure 2 makes a comparison with the standard GEP,MEP and the Multi-Subexpression Programming in the average fitness value. Evolution parameters are shown in table 2.

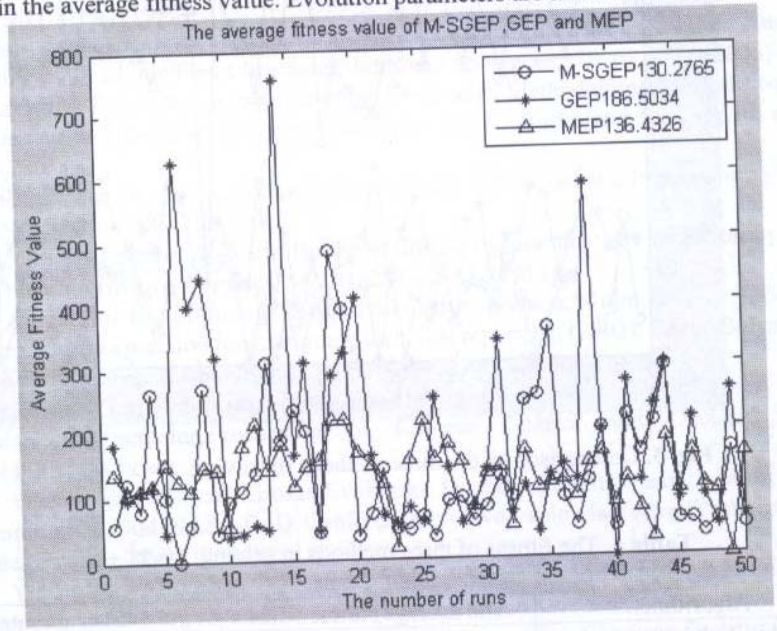

Fig. 2. Comparison of the three methods in solving $y=x^4+x^3+x$

In the fig.2, abscissa axis represents the number of evolution, and the ordinate is the average fitness value. It shows that the average fitness value of the M-SGEP algorithm is less than other two standard algorithms.

Table 3. The fitness of three methods in solving $y=x^4+x^3+x$

Algorithms	The average fitness value	Success rate
Multi-Subexpression Programming	130.2765	100%
Standard GEP	186.5034	62%
Standard MEP	136.4326	94%

The experimental results show that the average fitness value of the Multi-Subexpression Programming is less than other two standard algorithms, and it has higher success rate of mining. As a result of the original fitness, the average fitness value is lower, the individual's fitness is better. So Multi-Subexpression Programming has higher accuracy in function mining, and the evolution efficiency is higher.

In order to verify the validity of the improved algorithm, we conducted several experiments to test.The performance comparison of the other function $z=x^3+x^2y+y$ is given in figure 3, the average fitness value and success rate are shown in table 4.

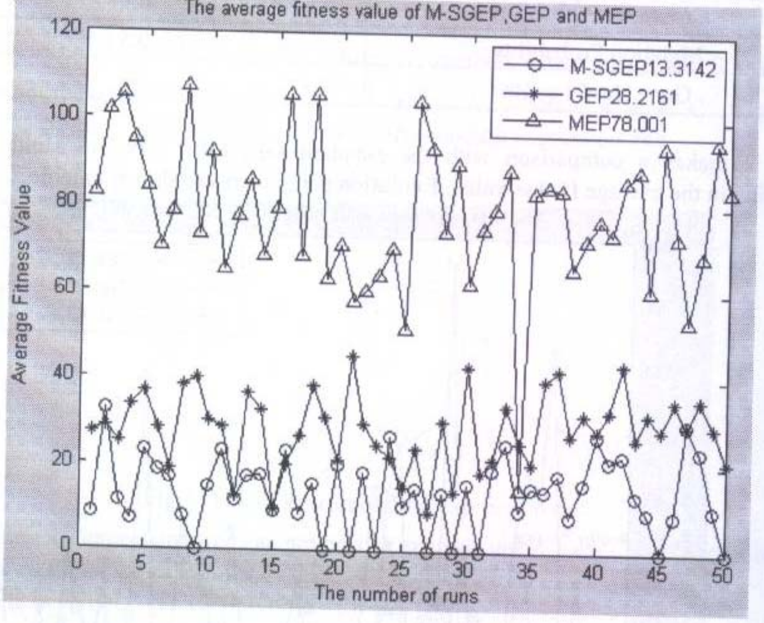

Fig. 3. Comparison of the three methods in solving $z=x^3+x^2y+y$

Table 4. The fitness of three methods in solving $z=x^3+x^2y+y$

Algorithms	The average fitness value	Success rate
Multi-Subexpression Programming	13.3142	100%
Standard GEP	28.2161	18%
Standard MEP	78.001	26%

The experimental result shows that the evolution efficiency of the Multi-Subexpression Programming is significantly higher than other two standards, and the success rate is higher.

5 Conclusion

This paper analyzes the advantages and disadvantages of standard GEP algorithm and standard MEP algorithm， and put forward a method for using the best fitness of subexpression as individual's fitness, This makes new algorithm can also have a good adaptability even the complexity of the target expression is unknown. Experimental results show that the evolution efficiency of the improved algorithm is much better than standard MEP and standard GEP.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (Grant NO. 61170199), and the Scientific Research Fund of Education Department of Hunan Province, China (Grant NO. 11A004).

References

[1]Mihai Oltean, Crina Grosan, Laura Diosan, Cristina Mihaila. Genetic Programming With Linear Representation a Survey, WSPC/INSTRUCTION FILE, (2008)

[2]Mihai O'Nell, Leonrdo Vanneschi, Steven Gustafson, Wolfgang Banzhaf. Open Issues in Genetic Programming. Genetic Programming and Evolvable Machines, 11:339-363. (2010)

[3]He Pei, Kang Lishan, Colin G. Johnson, Ying Shi. Hoare Logic-based Genetic Programming. Science China Information Sciences. 54(3): 623-637, (2011).

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

[5]Athanasios Tsakonas. A comparision of classification accuracy of four genetic programming-evolved intelligent structures. Informatin Sciences,176, 691-724,(2006)

[6]J. R. Koza and R. Poli, "Genetic programming," in Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques (E. K. Burke and G. Kendall, eds.), ch. 5, Springer, (2005).

[7]Mihai Oltean, Crina Grosan, A Comparison of Several Linear Genetic Programming Techniques. Complex Systems,14: 285-313. (2003)

[8]He Pei, Colin G. Johnson & Wang HouFeng. Modeling grammatical evolution by automaton. Science China Information Sciences, 54(12): 2544-2553. (2011)

[9]National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov M Oltean, D Dumitrescu. Multi expression programming, technical report[R], (2002), Babes-Bolyai University, Cluj-Napoca, Romania. Available from: http://www.mep.cs.ubbcluj.ro

[10]Yue hui Chen,Guangfeng Jia,Liming Xiu.Design of Flexible Neural Trees using Multi Expression Programming.Proceeding of Chinese Control and Decision Conference. 1429-1434.(2008)

[11]Mihai Oltean and Crina Grosan.Evolving Digital Circuits using Multi Expression Programming[C].2004 NASA/DoD Conference on Evolvable Hardware.Washington,DC:IEEE Computer Science, 87-94.(2004)

[12]Wang Yanan,Yang Bo,Zhao Xiuyang.Countour Registration Based on Multi-Expression Programming and the Improved ICP. IEEE (2009)

[13]Phil T. Cattani, Colin G. Johnson.ME-CGP: Multi Expression Cartesian Genetic Programming. IEEE Congress on Evolutionary Computation,1-6.(2010)