Improving Multi Expression Programming Using Reuse-Based Evaluation

Wei Deng[1] and Pei He[1,2]

[1] School of Computer and Communication Engineering
Changsha University of Science and Technology
Changsha 410114, P.R. China
76995958@qq.com

[2] Key Laboratory of High Confidence Software Technologies(Peking University)
Ministry of Education, Beijing 100871, P.R. China
bk_he@126.com

Abstract. Multi expression programming is a linear genetic programming that dynamically determines its output from a series of genes of the chromosome. It works on a fixed-length individual, but gives rise to the complexity of the decoding process and fitness computations. To solve this problem, we proposed an improved algorithm that can speed up individual assessments through reuse analysis of evaluations. The experimental result shows that the present approach performs quite well on the considered problems.

Keywords: Multi Expression Programming, Linear Genetic Programming, Fitness Computations, Reuse Analysis of Evaluations.

1 Introduction

Multi Expression Programming (MEP) as a linear genetic programming approach is a Genetic Programming variant first proposed in 2002 by Oltean.M and Dumitrescu.D[1-7]. Compared to other variants of Genetic Programming, a unique feature of MEP is its ability of storing multiple solutions of problem in a single chromosome[9]. In MEP each chromosome contains a number of expressions which, called genes, consist of strings of variable length, and the number of genes per chromosome is constant. These features make it possible to greatly increase the probability of the problem solution [8-10]. So we can effectively solve complex problems. Now multi expression programming has been widely used in many fields, such as classification problems, stock market forecast, TSP, digital circuit design, and so on[11-13].

In this paper, we will present an improved algorithm for efficiently assessing the individuals of MEP. The idea lies in the fact that the fitness can be quickly computed based on encoded chromosome, and any genes inherited from parents dont be evaluated once again.

The paper is organized as follows. In section 2, the basic principle of the multi expression programming is presented. The improved algorithm in MEP is described in section 3. In section 4, we will perform several numerical experiments. Section 5 concludes the paper.

2 Multi Expression Programming

2.1 MEP Algorithm

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[11]. Then the parents are recombined in order to generate new offspring by crossover. Finally the offspring are considered for mutation.

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[9]. Create new individuals based on encoded genes by crossover and mutation. Standard MEP algorithm is described in detail as follows.

2.2 Encoding Principle

A chromosome of MEP consists of several genes. Its length is equal to the number of genes. Each gene represented by strings of a variable length is composed of function operator, terminal symbol and gene sequence number. MEP encoding rules are simple. The first symbol of the chromosome must be a terminal symbol, and function arguments of the gene must be smaller than the current genetic index.

For example, let $F = \{*, +, S\}$ be the set of function symbols where the symbol S represents the sin function, and the terminal set be $T = \{x, y\}$. Suppose a chromosome contains 6 genes, an example of chromosome using the sets F and T will take the form of Table 1.

Table 1. Gene encoding of a chromosome: the first row is gene indices, the second is the corresponding genes encoding

1	2	3	4	5	6
x	$*1, 1$	y	$*1, 3$	$+2, 3$	$S5$

2.3 Decoding Principle and Fitness Assignment

In the evaluation process, the first step is to decode genes of the chromosome. For instance, in the previous example each gene would be decoded to a simple expression. These expressions are shown in Table 2.

Table 2. Gene encoding of a chromosome: the first row is gene indices, the second is the corresponding genes encoding

1	2	3	4	5	6
x	$x*x$	y	$x*y$	$x*x+y$	$sin(x*x+y)$

In general, phenotypic decoding process can be depicted based on Table 1 as follows. Genes 1 and 3 decode a simple expression by a single terminal symbol. Genes 2, 4, 5 and 6 include function symbol and gene indices, which, say gene 2, indicates the operation of multiplication on the operands gene1 and gene 1 of the chromosome. Therefore the translation of a gene into an expression involves complicate substitutions among sub expressions and gene indices. In this paper, we proposed an improved algorithm which does not need decoding, but evaluating genes by the encoded chromosome.

In MEP algorithm, the different fitness function has a different assessment effect for the gene. One of the most popular forms of fitness functions given below is the so called absolute error[11]:

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

Where $O_{j,i}$ is the value of the expression E_i on the jth sample data and W_i corresponding target result. The fitness of the chromosome fitness(C) is equal to the lowest fitness of the genes in a single chromosome[9]. This fitness is widely used for solving regression problems.

$$fitness(C) = minf_i(E_i) \tag{2}$$

2.4 Genetic Operators

The genetic operators used in the MEP algorithm are crossover and mutation. Since genetic operators preserve the chromosome structure, all the offspring are syntactically correct expressions[1].

Crossover: Two parents are selected randomly and recombined by crossover[1]. Cross point is also randomly generated. In this paper we consider one-point recombination.

Mutation: Terminal symbol, function symbol and gene index may be used to mutate the gene. In order to prevent the chromosome structure from damaging, the first gene must be encoded to a terminal symbol. If the current gene changes into a function symbol, the function arguments must be function pointers to the previous genes.

3 Improved MEP Algorithm

In order to evaluate chromosomes, we must translate each gene into an expression, and compute function and fitness value of the decoded genes. On the one hand, translation of expression consumes tremendous time and space resources; on the other hand the evaluation process will recalculate the same gene segments when the current gene contains gene indices. As a large number of the genes contains function pointer, this way will affect the evolution efficiency. So this paper presents an improved MEP algorithm which quickly computes the value of the expression based on the encoded genes, and don't need decoding genes. Some of the genes have not been modified by crossover and mutation, but their fitness will still be recalculated in a novel generation. So this paper proposed another improved algorithm which would preserve the primary fitness value of the unmodified gene segments, thus reducing the number of repeated computation. These two improved algorithm will be applied in MEP to solve practical problems.

The improved algorithm is described in MATLAB language as follows:

```
function chro_fit = evaluate(chro, flag,x_value)
%  Input arguments: a chromosome, modification flag of genes and
   sample data
%  Output arguments: fitness value of the chromosome
%  Global variables are initialized in the begin of evolutionary
   process, but not
%  in this m file. This is the m file of the evaluation.
%  y_value and gene_fit is initialized to null, and node_num
   is initialized to 0
global y_value;
%  output values of the parents
global gene_fit;
%  fitness values of the parents
global node_num;
%  the number of calculation nodes
for i = 1:length(chro)
%  ~ flag (i) = = 1 means that the current gene has not been
   modified.
if(gene_fit &~flag(i))
%  if gene_fit is empty, it represents the first generation
   continue;
% ~flag(i) == 1 and gene_fit is not empty then continue;
end
node_num = node_num + 1;
%  the number of calculation nodes plus one
gene = chro(i);
if(length(deblank(gene))<2)
%  the current gene is a terminal symbol
    y_value(i) = x_value(argument_index)
else
%  the current gene contains function symbol
%  function symbol a binocular operator
```

```
    if(gene(1)is a binocular operator)
       y_value(i)=bin_operator(y_value((gene(function_pointer1))),
       y_value((gene(function_pointer2));
else
%  function symbol a unary operator
        y_value(i) = unary_operator
                            (y_value(gene(function_pointer1)));
    end
end
    gene_fit = sum(abs(y_value (i)C x_value(sample_index)))
%  gene fitness value
end
chro_fit = min(gene_fit)
%  chromosome fitness value
```

The parameter flag is initialized to the value 0. Its value is modified only in genetic operation. Argument flag with the value 0 indicates that the gene has not been modified by the crossover and mutation, Otherwise the flag would be set to 1. If the gene contains function pointers and is not modified, we also need to consider whether the referred expression is modified. If the expressions that are pointed to by the function pointer have been modified, the gene flag should also be set to 1. In the process of evolution the number of calculation nodes is based on the flag. A node represents a gene.

The improved algorithm directly computes a function by scanning an encoded gene. If the gene is a terminal symbol, the function value is the argument sample data. If the gene contains function symbol, the result is calculated by the gene value of the function pointer. The standard MEP algorithm requires the translation of expressions to evaluate genes. But the improved MEP algorithm performs all arithmetic based on operators, and function pointer. In addition, unmodified genes dont participate in calculation except the first generation. The two improved ideas raise evolution efficiency.

4 Experiments and Analysis

In this section, several experiments with comparisons of standard MEP and the improved algorithm are performed on two well-known regression problems. The two functions to be examined are $y^4 + y^3 + y^2 + y$ and $sin(y^4 + y^2)$, respectively. We will compare the performance of these two methods in the time complexity and the number of calculation nodes by running them on the same problems based on the same 20 sample data for 50 times.

Fig. 1 makes a comparison between the standard and the improved MEP algorithm in the time complexity and the number of calculation nodes. Evolution parameters are shown in Table 3.

In the first diagram of Fig.1, abscissa axis represents the number of evolution, and the ordinate is the running time. It shows that the average evolution time of the improved MEP algorithm is about 9 times less than the standard algorithm.

Table 3. The experimental parameters of the two regression problems

Function Set	$\{+, *, S\}$
Terminal Set	$\{y\}$
Number of Chromosomes	20
Number of Genes	10
Number of Generations	1000
Crossover Probability	0.9
Mutation Probability	0.5

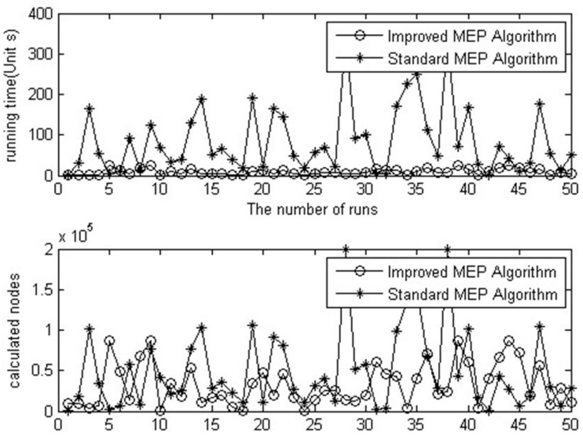

Fig. 1. Comparison of the two methods in solving $y^4 + y^3 + y^2 + y$

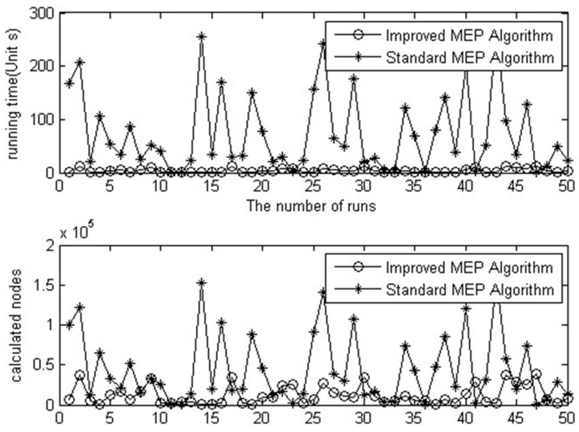

Fig. 2. Comparison of the two methods in solving $sin(y^4 + y^2)$

In the second diagram of Fig.1, abscissa axis is also the number of evolution, but the ordinate represents the calculated number of genes in the assessment process. The experimental results show that the number of calculation nodes by the improved algorithm was less than the traditional approach during the evaluation phase. The number of calculation nodes is based on the third part of the mentioned flag to decide whether to evaluate the gene.

In order to verify the validity of the improved algorithm, we conducted several experiments to test. The performance comparison of the other function $sin(y^4 + y^2)$ is given in the below Fig. 2.

The experimental result shows that the evolution efficiency of the improved algorithm is significantly higher than the standard. Moreover owing to that unmodified genes neednt be recalculated, the number of genes calculated decreases in the evolutionary process.

5 Conclusion

This paper analyzes the evaluation deficiency of standard MEP algorithmand put forward a method for sharing expression values among the same gene segment. Thus it becomes unnecessary to translate genes to phenotypic expressions in the evaluation stage. Furthermore, regarding evaluations of chromosomes, any genes inherited from parents neednt be recalculated. The two improved ideals greatly reduced the program running time and space complexity. Experimental results show that the evolution efficiency of the improved algorithm is much better than standard MEP.

Acknowledgment. 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. Oltean, M., Grosan, C., Diosan, L., Mihaila, C.: Genetic Programming With Linear Representation a Survey. WSPC/INSTRUCTION FILE (2008)
2. O'Nell, M., Vanneschi, L., Gustafson, S., Banzhaf, W.: Open Issues in Genetic Programming. Genetic Programming and Evolvable Machines 11, 339–363 (2010)
3. He, P., Kang, L., Fu, M.: Formality Based Genetic Programming. In: IEEE Congress on Evolutionary Computation, Hong Kong, pp. 4080–4087 (2008)
4. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press (1992)
5. Tsakonas, A.: A comparision of classification accuracy of four genetic programming-evolved intelligent structures. Informatin Sciences 176, 691–724 (2006)
6. Koza, J.R., Poli, R.: Genetic programming. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, ch. 5. Springer (2005)

7. Oltean, M., Grosan, C.: A Comparison of Several Linear Genetic Programming Techniques. Complex Systems 14, 285–313 (2003)
8. He, P., Johnson, C.G., Wang, H.: 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`; Oltean, M., Dumitrescu, D.: Multi expression programming, technical report, UBB-01-2002, Babes-Bolyai University, Cluj-Napoca, Romania, `http://www.mep.cs.ubbcluj.ro`
10. Chen, Y.H., Jia, G., Xiu, L.: Design of Flexible Neural Trees using Multi Expression Programming. In: Proceeding of Chinese Control and Decision Conference, vol. 1, pp. 1429–1434 (2008)
11. Oltean, M., Grosan, C.: Evolving Digital Circuits using Multi Expression Programming. In: 2004 NASA/DoD Conference on Evolvable Hardware, pp. 87–94. IEEE Computer Science, Washington (2004)
12. Wang, Y., Yang, B., Zhao, X.: Countour Registration Based on Multi-Expression Programming and the Improved ICP. IEEE (2009)
13. Cattani, P.T., Johnson, C.G.: ME-CGP: Multi Expression Cartesian Genetic Programming. In: IEEE Congress on Evolutionary Computation, pp. 1–6 (2010)