An Adaptive Multi-phenotype GEP Algorithm

Qu Li, Chao Chen, Weihong Wang, Cheng Ren
College of Computer Science and Technology
Zhejiang University of Technology
Hangzhou, China

Siliang Xia
School of Software
Harbin Institute of Technology at Weihai
Weihai,China

***Abstract*—Gene Expression Programming (GEP) is a GA and GP based evolutionary computation method. Researchers have improved the efficiency and effectiveness GEP by optimizing gene encoding scheme and inventing new genetic operators. An Adaptive Multi-phenotype GEP (AM-GEP) is proposed in this paper in order to solve the problem of excessive evolutionary generations and massive time consuming. In the evolutionary process of AM-GEP, individuals change the number of genes adaptively, and the convergent rate is greatly improved with a new gene combination mechanism. Experiments showed that running time is reduced significantly, the expression space of chromosomes is extended, and the quality of solutions is improved with AM-GEP.**

Keywords- Gene Expression Programming, Gene Combination, Superior Gene

I. INTRODUCTION

Gene expression programming (GEP) is a novel evolutionary algorithm derived from Genetic algorithm (GA)[1] and Genetic Programming (GP) [2]. It uses fixed-length strings of symbols organized as linear polygenic chromosomes to encode solutions (parse trees) and thus clearly separates genotype from phenotype. GEP has been used in various fields, such as symbolic regression, function finding, classification rules mining. It has been widely applied to scientific and technical problems for modeling and prediction [3,4].

The coding method of GEP is a topic center of GEP research since its invention in 2001. Many researchers have proposed different coding schemes by analyzing GEP's shortcomings in many aspects. S. Lopes[5] proposed an GEP coding method which can vary the length of gene according to problem's difficulty. P. Jing[6] proposed an multilayer chromosomes with hierarchical structure which is able to call sub-structures layer by layer. D. Lei[7] and S. Dai[8] also proposed two different coding scheme in order to utilize code reuse of GEP genes. Elena Bautu[9] proposed an adaptive GEP (AdaGEP) which automatically adjusts the required number of genes according to problems. To a certain extent, it helps to determine the optimum number of genes and evolves better solutions than basic GEP with fewer symbols in shorter period of time. However, AdaGEP ignores different future change of the number of genes when meeting different problems and thus reduces the probability of which superior genes are selected, slowing down convergent speed. Z. Wu[10] also proposed a new GEP algorithm based on multi-phenotype chromosomes.

This paper proposed an Adaptive Multi-phenotype GEP (AM-GEP) to overcome the weaknesses of AdaGEP. Experimental results indicate that AM-GEP outperforms AdaGEP with less computational effort. The next sections of the paper outline the main features of the AM-GEP algorithm. In section II, we briefly give the background of GEP and AdaGEP. Section III presents the new AM-GEP algorithm and its improvement in GEP chromosomes and explains the "gene combination" strategy. Section IV presents the experimental results of AM-GEP against basic GEP and AdaGEP. In section V, we draw some conclusions and point out further research directions on AM-GEP.

II. BACKGROUND

A. GEP Algorithm

The genes of gene expression programming are composed of a head and a tail. The head contains symbols that represent both functions and terminals, whereas the tail contains only terminals [1]. For each problem, the length of the tail t is a function of the length of the head h and the number of operators within the function set, n:

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

Consider a gene in which the functions set is F = {+, -, *, /} and the terminal set is T = {a, b}. In this case n = 2 (because add, minus, multiply, divide are all binary operators.); let h = 3, then t = 3* (2 - 1) + 1 = 4; thus, the length of the gene is 7. For example, one gene is shown below:

```
0123456
*a+abba
```

It codes for the following Expression Tree (ET):

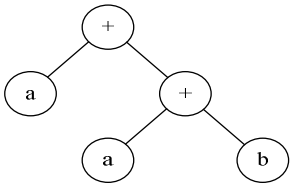

Figure 1. An Expression Tree

By parsing the expression tree from top to bottom, from left to right, a correct string can be derived. GEP chromosomes can also be composed of more than one gene. Each gene is coded as a sub-ET and these sub-ETs form a more complex ET with linking functions. The chromosomes in GEP are subjected to

978-1-4577-2133-5/10/$26.00 ©2012 IEEE

initialization, evaluation, selection, recombination and mutation operators, and the best solution is derived after generations of evolution, as other evolutionary computation techniques do. Usually five components are specified when using GEP to solve a practical problem: the function set, the terminal set that includes problem-specific variables and pre-selected constants, fitness function, control parameters, and stop condition [2].

B. AdaGEP Algorithm

The number of genes in GEP greatly affects the efficiency and accuracy of the algorithm. If the number of genes is too small, a good solution may never be found for complex problems because gene sequences lack the ability of expression; if it is too large, good solutions can be encoded by chromosomes, but extra genes constructing ETs usually extend the solution space, increasing time complexity for searching best solutions. In AdaGEP[9], each GEP chromosome is improved with a sequence named "genemap". Genemap is composed of binary bits containing either 0 or 1. Each bit in the genemap corresponds to a gene of the chromosome and those bits in genemap determine whether a gene related to should be decoded. If a genemap bit is set as 1, the corresponding gene will be decoded. Otherwise, the corresponding gene will be ignored in the decoding process, the gene is thus "deactivated" [9]. For example, if the genemap is 011, the first gene is deactivated and the AdaGEP chromosome is as follows:

```
Gene Map      0           1           1
Chromosome 012345678  012345678 012345678
          +aa+aaaaa  *aa-aaaaa /aa+aaaaa
```

It encodes for the following ET:

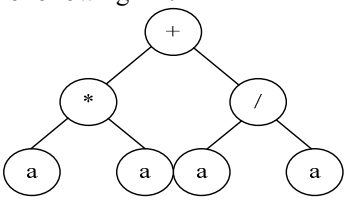

Figure 2. An Expression Tree for AdaGEP

III. THE AM-GEP ALGORITHM

A. Encoding Method

Although the problem of parameter setting of gene number is solved by AdaGEP by introducing "genemap", initializing the genemap in AdaGEP randomly is not a good approach because it ignores difference in changing tendency of gene numbers under various circumstances and thus causes the decreased likelihood of expression of superior genes. Thus the convergent rate is decreased and the algorithm may become less efficient than basic GEP.

Consider the chromosome containing two genes, where the length of a gene is 7 and linking function is '+':

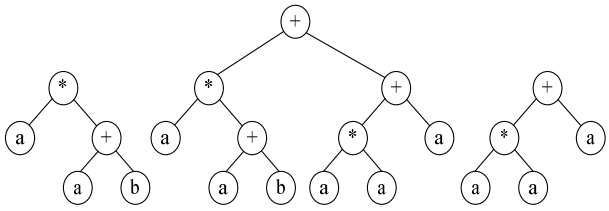

Figure 3. Two Expression Trees Linked by "+"

TABLE I. COMPARISON OF THE NUMBER OF ETS IN AM-GEP AND ADAGEP

Gene Number	ET in GEP	ET in AM-GEP
1	1	1
n	1	2^n - 1

In AM-GEP, a chromosome contains 2^n-1 ETs, but in AdaGEP, a chromosome contains 1 ET. The comparison of the number of ETs in AM-GEP and AdaGEP is shown in Table I.

As every ET has its own express value (EV), there must be a best ET among all of the 2^n-1 ETs according to EV. In AM-GEP, a gene of multiple phenotypes has the potential to be decoded as various expression trees. Therefore, there must be a best phenotype among all phenotypes in terms of fitness. Thus, we set the value of genemap according to the best phenotype. This technique is defined as "Gene Combination". By means of gene combination, we initialize the genemap in purpose rather than randomly. We define the gene in best phenotype as superior gene. In the initialization of genemap table, superior genes are set as active genes. Moreover, Gene Combination is applied in AM-GEP every 100 generations. The algorithm is shown as follows:

Algorithm：Gene Combination
Input：**G**ene Number: N, Population Size: M, Population
Output：Gene-Map and Chromosomes
BEGIN
1）Create the initial population of individuals
2）**for** i =1 to M **do**
3） F[M] : = 0.0 ， MaxFitness : = 0.0 ， index := 0
4） **for** j = 1... 2^n -1 **do**
5） Set F[j] to its corresponding EV
6） **if** MaxFitness < F[j] **then**
7） index := j
8） MaxFitness := F[j]
9） **end if**
11） **end for**
10） Set Gene Map to its corresponding BET
11）**end for**
END

B. Fitness Function

The discovery of solutions to a problem is greatly determined by the way the fitness function is designed: the goal must be clearly and correctly defined in order to make the population evolves towards that direction. Ferreira presents two

kinds of fitness function in her paper [1]. The following fitness function is chosen in AM-GEP (3.2).

$$F = \sum_{i=1}^{N} \left(M - \left| \frac{v_i - v_i'}{v_i} \right| \right). \qquad (2)$$

Where N is the number of fitness cases, M is a constant value, v_i is the target value for the fitness case i and v_i' is the returned value by chromosome for fitness case i.

C. Mutation, Transposition and Recombination

The three genetic operators are important to GEP algorithm because they are necessary conditions for gene diversity. As for AM-GEP, a different evolution strategy is applied. In AM-GEP, genes are divided into two groups: superior genes and inferior genes. Mutation is only performed in evolution of genemap in order to guarantee superior genes are more likely to be selected[1].

IV. Experiment Results and Analysis

We conducted several experiments to compare AM-GEP with basic GEP and AdaGEP on several function finding problems. All the following experiments are conducted on the following platform: CPU: Intel(R) Core(TM) Duo P7450, Memory: 2.0GB, OS: Windows XP, IDE: Eclipse 3.6.1. All algorithms are implemented in Java. We used the same parameters and fitness function for three benchmark problems.

A. Experiment 1

The first experiment based on a fairly simple one variable function.

$$y = x^3 + x^2 + x + \text{Î} \quad . \square \ (3)$$

The training set consists of 10 data records. Independent variable x is generated within the scope of [-10, 10]. M is a constant value of 100. The fitness function is shown in Equation (2). Other parameters are shown in Table II.

TABLE II. Parameters of GEP, AdaGep and Am-GEP

Parameters	GEP	AdaGEP	AM-GEP
Population size	100	100	100
Train set size	10	10	10
Function set	+-*/	+-*/	+-*/
Linking function	+	+	+
Head length	10	10	10
Total length	21	21	21
Mutation rate	0.02	0.02	0.02
IS transposition rate	0.01	0.01	0.01
RIS transposition rate	0.01	0.01	0.01
Gene transposition rate	0.02	0.02	0.02
Single-Point recombination rate	0.05	0.05	0.05
Two-Point recombination rate	0.05	0.05	0.05
Gene recombination rate	0.05	0.05	0.05
precision	0.00001	0.00001	0.00001
Performance	GEP	AdaGEP	AM-GEP
Success Rate	93%	94%	96%
Average Evolutionary Generations	10.88	11.65	6.36

The algorithms are executed 100 times independently. Results are shown in Table III. From Table III, we can see that the average success rate of AM-GEP is 96%, while the average success rate of AdaGEP and basic GEP are 93% and 94%. This shows that AM-GEP outperform AdaGEP and basic GEP in terms of success rate. The average evolutionary generation of AM-GEP is only 118.87, for AdaGEP, it is 235.30 and as for basic GEP, it's 334.97. It means that AM-GEP and AdaGEP extends the search space of GEP by sacrificing the evolutionary speed. Thus, the average search time of both AM-GEP and AdaGEP is much longer than basic GEP.

The result of Experiment is given in From Table IV. From Table IV, we can see that in AM-GEP, appropriate solutions are found 72 times within 10 generations, but for AdaGEP, the number is only 42 times, even less than the basic GEP.

TABLE III. Results of Experiment 1

Performance	GEP	AdaGEP	AM-GEP
Success Rate	93%	94%	96%
Average Evolutionary Generations	10.88	11.65	6.36
Max Evolutionary Generations	102	71	22
Min Evolutionary Generations	0	0	0
Average Search Time(sec)	3.24	4.95	5.84
Average Running Generations per Sec.	334.97	235.30	118.87

TABLE IV. Analysis of Evolutionary Generations of Experiment 1

Generation	GEP	AdaGEP	AM-GEP
[0, 10]	65	42	72
(10, 100]	33	58	28
[100, ∞)	2	0	0

According to this experiment, we come to the conclusion that AM-GEP outperforms basic GEP and AdaGEP in terms of Evolutionary generations. Gene Combination method improves not only the convergent rate of AM-GEP, but also increases its computation effort, which leads to the average search time to almost the same as AdaGEP. To better examine the advantage of AM-GEP, another experiment is conducted.

B. Experiment 2

This experiment is based on a 5-order one variable function. Consider the following function:

$$y = x^5 - 2x^3 + \tilde{x} \qquad . \square (4)$$

Three algorithms are executed 100 times separately. The results of this experiment are shown in Table V.

TABLE V. Results of Experiment 2

Performance	GEP	AdaGEP	AM-GEP
Success Rate	53%	53%	59%
Average Evolutionary Generations	53.43	46.67	26.11
Max Evolutionary Generations	1089	291	87
Min Evolutionary Generations	4	7	0
Average Search Time(sec)	26.70	21.38	14.12
Ave. Running Generations per Sec.	276.10	287.14	186.46

TABLE VI. ANALYSIS OF EVOLUTIONARY GENERATIONS OF EXPERIMENT 2

Generation	GEP	AdaGEP	AM-GEP
[0, 50]	85	68	92
(50, 100]	6	28	8
(100 ,500]	4	4	0
(500, ∞)	5	0	0

From the result, we can see that the average Evolutionary generation of AM-GEP(14.12) is only 55.9% of that in AdaGEP(21.38), and almost 48.8% of basic GEP(26.70). The average search time in the AM-GEP is only half as that of the basic GEP. From this experiment, it is quite obvious that AM-GEP outperforms basic GEP and AdaGEP.

C. Experiment 3

This experiment is based on a 6-order one variable function. Consider the following function:

$$y = x^6 - 2x^4 + x^2 \qquad (5)$$

Three algorithms are executed 100 times separately. The results of this experiment are shown in Table VII.

TABLE VII. ANALYSIS OF EVOLUTIONARY GENERATIONS

Performance	*GEP*	*AdaGEP*	*AM-GEP*
Success Rate	27%	35%	35%
Average Evolutionary Generations	130.87	95.24	43.88
Max Evolutionary Generations	1270	690	262
Min Evolutionary Generations	8	8	0
Average Search Time(sec)	82.00	34.82	26.01
Average Running Generations per Second	193.72	282.97	204.78

From the result, we can see that the average evolutionary generations in AM-GEP is only 43.88.While for AdaGEP, it is 95.24. And for basic GEP, the number is 130.87. The average search time in the AM-GEP is less than 1/3 of basic GEP. The success rate in AM-GEP and AdaGEP is also higher than basic GEP. In summary, as problems become more complex, the performance of AM-GEP becomes better with higher success rate and less computational effort.

V. CONCLUSION

This paper proposed an improved AdaGEP algorithm called AM-GEP. It took the advantage of the Gene Combination, not only improved the convergent rate, leading to the significant decrease in the running times, but also extended the expression space of chromosomes. Further research includes optimizing Gene Map Evolution, mechanism of the gene combination, and effect of under different parameter settings. It would be also interesting to apply AM-GEP in other fields like time series prediction and text classification.

REFERENCES

[1] David E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley Longman Publishing Co., Inc,1989.

[2] Koza J.R, "Genetic Programming: On the programming of Computers by means of Natural Selection", Cambridge: MIT Press, 1992.

[3] C. Ferreira, "Gene Expression Programming: "A New Adaptive Algorithm for Solving Problems", Complex Systems, vol. 13, No.2, pp. 87-129, 2001.

[4] C. Ferreira, "Gene Expression Programming: Marhematical Modeling by an Articial Intelligence(2nd ed)". Berlin Heidelberg: Springer-Verlag, 2006.

[5] Heitor S. Lopes, Wagner R. Weinert, "EGIPSYS: An Enhanced Gene Expression Programming Approach for Symbolic Regression Problems", Int. J. Appl. Math. Comput. Sci., Vol. 14, No. 3, pp. 375–384, 2004.

[6] Peng Jing, Tang Changjie, Li Chuan, et al. M-GEP: A new evolution algorithm based on multilayer chromosomes Gene Expression Programming. Chinese Jouanl of Computer, 2005, 28 (9),pp.1459 - 1466.

[7] D. Lei, T. Changjie, L. Yintian, Z. Jie, W.Jian, "Design and Implementation of ORF Filter in Gene Expression Programming", Journal of Sichuan University., vol. 39, No. 6, pp. 102-106, Nov 2007.

[8] S. Dai, C. J Tang, M. Zhu, Y. Chen, P. Chen, S. Qiao, C. Li, "MERGE: A Novel Evolutionary Algorithm Based on Multi Expression Gene Programming", ICNC '08. Fourth International Conference on Natural Computation, 2008., vol. 1, pp. 320 – 324, Oct 2008.

[9] E. Bautu, A. Bautu, H. Luchian, "AdaGEP - An Adaptive Gene Expression Programming Algorithm", Proceedings of the Ninth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pp.403-406, 2007.

[10] Zenghong Wu, Min Yao, "A New GEP Algorithm Based on Multi-phenotype Chromosomes", Second International Workshop on Computer Science and Engineering, WCSE '09., pp. 204 – 209, 2009