The Strategies to Improve Performance of Function Mining By Gene Expression Programming[1]
—Genetic Modifying, Overlapped Gene, Backtracking and Adaptive Mutation

Changjie Tang Lei Duan Jing Peng Huan Zhang and Yixiao Zhong

School of Computer Science, Sichuan University, Chengdu 610065, P. R. China

E-mail: {tangchangjie, duanlei}@cs.scu.edu.cn

Abstract This paper introduces the technologies to improve the performance of function mining by Gene Expression Programming (GEP) developed in Sichuan University last year. The main results include: (a) Genetic Modifying Algorithm (Trans-gene). By injection gene segment into genome, it guides the evolutional direction and speeds up knowledge discovery process. (b) Overlapped gene expression. Borrowing the idea of overlap gene expression from biological study, it applies overlapped gene expression, saves space for gene expression. (c) Backtrack-able GEP. Enlightened by atavism in biology, it proposes backtrack-able GEP algorithms, designing Geometric Proportion Increased Checkpoint Sequence and Accelerated Increased Checkpoint Sequence to restrict the backtrack process. (d) Adaptive Mutation. The mutation rate for each individual can vary in evolution according to the value of fitness. Experiments show that these techniques boost the performance of GEP by one or two magnitudes, respectively.

Keyword Template Knowledge discover, Gene Expression Programming, Function Mining

1. Introduction

Gene Expression Programming (GEP) [1] is a new member in the family of genetic computing. It combines advantages in both Genetic Algorithms (GAs) and Genetic Programming (GP). In GEP, candidate solutions are called chromosomes and represented as linear strings with fixed-length, and can be easily expressed as expression trees (ETs). The GEP chromosome and its coding style are designed so perfectly that chromosomes always alive under various genetic operations, hence always results in a valid expression tree. Based on the genetic operators and the separation of genotype and phenotype, GEP is endowed with more flexibility and power of exploring the entire solution space compared with traditional GAs and GP [2]. GEP offers great potentiality to solve complex modeling and optimization problems and it has been used to solve a large variety of problems efficiently, including symbolic regression, function finding, classification, time series analysis, logic synthesis and cellular automata, etc. [3, 4, 5].

This paper gives a survey to the new techniques for GEP developed in Sichuan University in 2005. It focuses on the key idea of the strategies to improve the performance in discovering function by Gene Expression Programming, i.e. trans-gene, overlapped expression and backtracking evolution and adaptive mutation.

2. The Basic Concepts and Terminologies

The main process of GEP is similar to its predecessors, GAs and GP. The essential difference is: in GAs the individuals are symbolic strings of fixed length; in GP the individuals are non-linear entities of different sizes and shapes; and in GEP the individuals are also non-linear entities of different sizes and shapes, but these complex entities are encoded as simple strings of fixed length. In GEP, the expression trees consisting of the genetic information encoded in the chromosomes. The chromosome consists of a linear, symbolic string of fixed length composed of one or more genes. GEP genes are composed of a head and a tail. The head contains symbols that represent both functions and terminals, whereas the tail contains only terminals. For each problem, the length of the head h is chosen, whereas the length of the tail t is a function of h and the number of arguments of the function with more arguments n, and is evaluated by the equation: $t = h(n - 1) + 1$. Consider a gene for which the set of functions $F = \{+, -, *, /\}$. In this case the maximum arity of F is 2, then $n = 2$.

Through parsing the expression tree from left to right and from top to bottom, the valid part of GEP genes can be got. Thanks to the structural organization of GEP genes, any modification made in the chromosome, no

[1] This work was supported by the National Science Foundation of China under Grant No.60473071, the National Research Foundation for the Doctoral Program by the Chinese Ministry of Education under Grant No.20020610007.

matter how profound, always results in a valid expression tree. So all programs evolved by GEP are syntactically correct. Based on the principle of natural selection and "survival for the fittest", GEP operates iteratively evolving a population of chromosomes, encoding candidate solutions, through genetic operators, such as selection, crossover, and mutation, to find an optimum solution.

Other than C. Ferreira's researches [1,2,3], several studies based on GEP have been performed, such as mining predicate association rule by GEP [5], predicting time series based on GEP [6], and mining functions from data set containing noise [7].

GEP algorithm begins with the random generation of the chromosomes of the initial population. Then the chromosomes are expressed and the fitness of each individual is evaluated. The individuals are then according to fitness to reproduce with modification, leaving progeny with new traits. The individuals of this new generation are, in their turn, subjected to the same developmental process: expression of the genomes, confrontation of the selection environment, and reproduction with modification. If a solution of satisfied quality is found, or a predetermined number of generations is reached, the evolution stops and the best-so-far solution is returned.

According to both fitness and the selection method, individuals are selected to reproduce with modification, creating the necessary genetic diversity allowing for adaptation in the long run. In nature, several modifications, like mutation, deletion, and insertion, are introduced during the replication of the genomes. In basic GEP algorithm, the genetic operators perform in an orderly fashion, starting with replication and continuing with mutation, transposition, and recombination.

The mutation operator aims to introduce random modifications into a given chromosome. Of the operators with intrinsic modification power, mutation is the most efficient [1]. With mutation, populations of individuals adapt very efficiently, allowing the evolution of good solutions to virtually all problems. In GEP, there are three transposition operators: insertion sequence (IS), root IS (RIS) and gene transposition. The transposable elements of GEP are fragments of the genome that can be activated and jump to another place in the chromosome. Furthermore, in GEP there are three kinds of recombination: one-point recombination, two-point recombination and gene recombination. In all types of recombination, two chromosomes are randomly chosen and paired to exchange some material between them, resulting in the formation of two new individuals. The implementations of mutation, transposition and recombination are detailed in [1, 2].

3. The GEP with Genetic Modifying

3.1. The genetic modifying in Bio-engineering

The genetic modified technique in Bio-engineering aims at creating new species or fastening evolution by injection the classified or modified gene into genome of organism. Once trans-gene is integrated, they will be entailed upon offspring and produce corresponding biological functions. Biologists classify the required genes, clone them, and inject them into target species. The key points are:

- Separate target gene or gene segments.
- Reorganize DNA ectogenically (out of the body). Put foreign DNA segment into receiver body.
- Filter out the "good" DNA and clone.
- Clone target gene to acceptor body and expressed gene to get the desired property.

3.2. The basic idea behind Genetic Modifying

Enlightened by the modern genetic modified technique in bio-engineering, we proposed Genetic Modifying and Gene Injection algorithms to control evolution direction. It combines *Nature selection* and *Human selection*. It gets excellent species in relative short evolution procedures. However, we name our genetic modifying algorithm as *Trans-gene* to differentiate it from the technique used in bio-engineering. For special problem, the selection of good genes and injection time is based on the evaluation of fitness and under guidance of heuristic rules.

In the original GEP invented by C. Ferreira, the evolution procedure is loose-controlled. Once the evolution begins, the whole population is wild. Users passively wait for the results produced by evolutions after specified generations. The "good" gene accumulated by many generations may be destroyed in one "bad" mutation. The key ideas of *Trans-gene* are: (a) classify the "good" genes, and store them in gene library; (b) inject "good" genes into proper individuals at proper evolutional step to quicken the process of evolution.

For example, let *Attn* = $(1-x+x^2/2-x^3/6)$, $(\approx e^{-x})$. It is an attenuation gene consisting of 20 basic symbols. It is easily destroyed by a "bad" mutation. In our model, *Attn* is in stored in Gene-lab as an "atomic". It may be injected into object when attenuation property is apperceived to speed up the evolutionary process.

3.3. The key steps in GEP with Trans-gene

The kernel technique of GEP-with Trans-gene is the injection of foreign gene segments; GEP algorithm can not predict the target property. Foreign gene is dependent on the evolutional process. The experience shows that, after evolving enough generations, the "good" structures appear in genes of excellent individuals. These genes can be stored in a buffer for reproduction. To keep these good structures, we proposed algorithms to separate and decompose gene in [8], the key steps are as the name of the following algorithms:

- Algorithm Get_sigle_Gene_from_chromosome (chrom); output single_gene.
- Algorithm Get_sement_from_chromosome (single _gene); by deletion some factor and terminal symbol in single_gene, it decompose single_gene as some Gene_sengment.
- Algorithm Single_Gene_Evolution(); Based on previous steps, The chromosome with good genes is selected and dispatched to separated population for independent evolution to develop excellent gene.
- Algorithm Gene_Segment_Evolution(); Extract gene segments from genes as individual dispatched to separated population for independent evolution.
- Algorithm GGSE to Filter Gene for GSE and FGSGE to Filter Gene for SGE...Filter good chromosome to prepare evolution.
- Algorithm GEP-trans-gene ();
- Each segment got from previous evolution is evaluated, selected and sent to independent evolution.

3.4. The experiment on GEP with Trans-gene

Two experiments with different parameters are done on GEP and Trans-GEP (TGEP) for 10 times respectively. The detail results can be seen in [8]. The object function is the famous Schaffer function f_6 as following, where $-100 \le x_i \le 100$ (i = 1, 2)

$$f_6(x_1, x_2) = 0.5 + \frac{\sin^2 \sqrt{x_1^2 + x_2^2} - 0.5}{\left[1.0 + 0.001\left(x_1^2 + x_2^2\right)\right]^2}$$

There are 1000 test records are synthesized by f_6 formula with random x_1 and x_2. To simulate the real mining environment, no human intervene are given. In 10 times random initialized experiments, four results of TGEP are better than the best result of GEP. The average fitness of TGEP is 0.11 higher than the one of GEP. Fig. 1 gives the comparison of TGEP and traditional GEP.

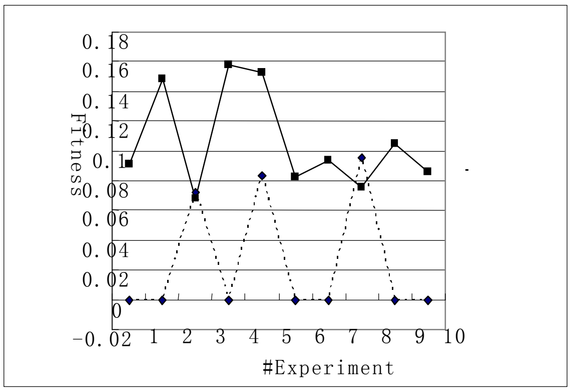

Fig. 1. The evolution of GEP & TGEP for function f_6

4. Overlapped Gene Expression Life-form

4.1. The features of overlapped gene expression

Compared with other evolutionary algorithms, the Evolutionary Algorithm based on Overlapped Gene Expression (EAOGE) has the following advantages:

- An individual consists of several genes. Gene segments can be overlapped under certain conditions.
- EAOGE is efficiency in space, since the segments are overlapped.
- There is no need to restrict the content of a gene or a chromosome. Both GP and GEP have to restrict the formats of gene in some ways such as the type and the length of gene head and tail in GEP. Experiments show that under same condition, the velocity of EAOGE is 2.8 to 9.7 times of GEP.

The capability of discovering higher-degree polynomial function is high. Compared with GEP, EAOGE greatly increases the success rate in polynomial function mining.

4.2. Definitions and encoding methods

Different from existing GEP, our EAOGE code does not have head or tail concepts, and any position in the gene can include the elements of *F* and *T*. Hence EAOGE code has the simplicity as GA. On the other hand, like GEP algorithm, EAOGE can be translated into a unique corresponding expression according gene coding. The translating process is as follows:

- Scan each element of gene in order.
- If the current symbol belongs to *T*, then let it be a leaf-node in ET
- If the current symbol belongs to *F*, it is a non-leaf node in ET, the number of its sub-trees equals the number of the function parameters. Let the element which is the directly succeeding of current symbol

be the first root node of the sub-trees, the secondary element be the root node of secondary sub-trees, and the rest may be deduced by analogy. It meets the end of a gene. The first element in *T* is a sub-tree root-node.

In the viewpoint of code structure, EAOGE possesses the simplicity of GA. It does not need to restrict the elements in the gene; on the other hand, EAOGE can also form expression tree to complete the mapping from genotype to phenotype. Thus it makes a good foundation to solve complicated polynomial function mining.

4.3. The research results on Overlapped Gene Expression

Algorithm EAOGE simulates Nature Selection over biome. It implements genetic operation, such as mutation, transition recombination, etc; evolve populations, selects excellent individual as solution to given problem. We gave a series algorithms and theorems in [9]. By the limitation of paper space, here give some important results:

(a) Space theorem for Multi-Genes

Assume the number of parameters in the operator set is 2, *m* is the length of chromosome and *k* is the number of genes. Then the maximal expression space of multi-gene individual *I* in EAOGE algorithm, $MAX_m(D_I)$, satisfies:

$$MAX_m(D_I) = k \times \frac{2}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{\frac{m}{k}+2} - \left(\frac{1-\sqrt{5}}{2}\right)^{\frac{m}{k}+2}\right) - 1$$

(b) The Theorem on existence of equivalent gene-type

Assume $H(x_1, x_2, \ldots, x_n) = x_1^{p_1}x_2^{p_2}\cdots x_n^{p_n}$, where x_i is a variant, $P_1, \ldots, P_n$ is non-0 integer. Then there is a genotype *E* of EAOGE algorithm, such that the expression of *E* equals

$$f(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{k} x_1^{p_{i1}} x_2^{p_{i2}} \cdots x_n^{p_{in}}$$

We gave four experiments in [9]. Here simply introduce the first results. Consider discovery of function with two arguments. We synthesize test data of 20 records by formula $Z=X^5+3*X*Y$, X and Y are in [-3, 3], M=10000, Take gene length are 9~23 for EAORG and GEP. We run EAORG and GEP 100 times. The average number of generation, Max number of generation and minimum number of generation are shown in Fig.2. The time consumed is shown in Fig 3. The extended experiments show that the speed for FAOGE is 2.8~9.7 times faster than GEP. In the problem to discover function containing high rank polynomials, WAOGE is much better than GEP. The details can be seen in [9].

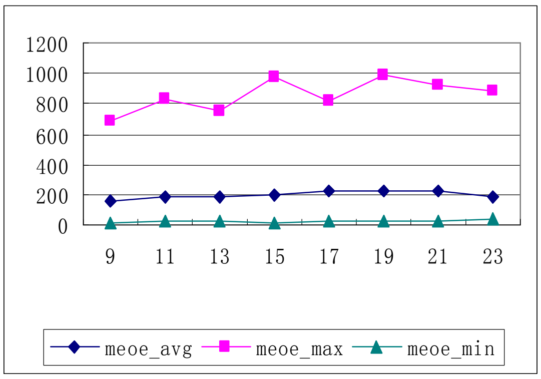

Fig. 2. Comparison for different Gene Length in EAOGE

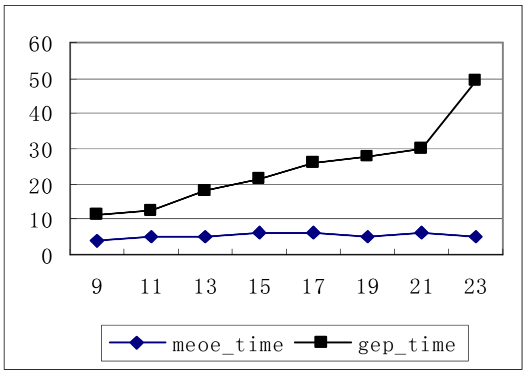

Fig. 3. Comparison of the time cost by EAOGE and GEP

5. The Backtracked GEP

5.1. Inspiration from atavism

When GEP evolutional process reaches specific generations, the average fitness is high enough, the diversity of population is small, the evolution may fall into the trap of local peak and losses the chance to get global optimization [8].This is so called prematurity like that in life-form world. The atavism in life-form gives solution inspiration. In the view point of modern genetics, the reasons of atavism are: (a) some lost gene of ancestor re-combined by crossbreed or mutation. (b) some gene of ancestor is inactive by stop-protein. The stop-protein is broken off by some causation and the gene actives again. This shows that the evolution process is reversible. To solve prematurity problem, we proposes *Backtracked GEP*, It gives traditional GEP a chance to modify evolution direction by backslide.

5.2. The key points in Backtracked GEP

The evolutional steps are along with the generation number. A Backtracked Checkpoint Sequence (BCS) and a stack are maintained by an algorithm.

- Check the maximal fitness at pre-specified check point; compare it with the fitness at previous checkpoint in stack. If the later fitness is higher, then the evolution is valid, push the current population in stack.
- Otherwise, the evolution got in wrong direction. Quits by poping stack top, re-start evolution form previous population

Since GEP searches in random style, in the sense of probability, re-evolution does not repeat previous evolution steps, hence daps from prematurety.

5.3. New concepts in Backtracked GEP

(1) Backtracked Checkpoint Sequence (BCS). It is a pre-specified generation number to check whether a backtracked step should be considered. Each BCS is with corresponding fitness and stack node. In practice we observed that the constringency speed is nonlinear. In the earlier stage, the diversity in population is high, hence constringency speed is high. At the later stage, the diversity in population is low, hence constringency speed is low.

(2) Geometric progressing Backtracked Checkpoint Sequence (GPBCS) and accelerative increase backtracked checkpoint Sequence (AIBCS).

(3) Degeneration factor α, a positive number, to control backtrack. When backtrack from $g_{i+1}$to g_i, it makes new population at g_i as $\alpha * P_{gi} + (1 - \alpha) * P_{gi+1}$.

(4) Scalable backtrack. Especially, when Degeneration facto α = 1, the evolution is backtracked GEP. If α = 0.5 it is called semi-backtracked GEP, if α = 0 it degenerates to traditional GEP

5.4. The experiments for Backtracked GEP

Two experiments are given in [9]. The Data for the first Experiment is as that in [11]. C. Ferreira used it to verify mining capability of GEP in the five dimensional spaces. We used the target function produce 50 data as evolution environment. Since the function is rather complex and dimension is not low, we used *Geometric Progressing Backtracked Checkpoint Sequence* (GPBCS). The fitness threshold for success is 0.8. The comparison of traditional GEP and Backtracked GEP is shown in Table 1.

Table 1. The success rate comparison (I)

	traditional	GPBCS
number of records	100	100
success rate	40%	90%

The data for decode experiment is borrowed from [11]. It is relatively simple. We used *accelerative increase backtracked checkpoint sequence* (AIBCS). The fitness threshold for success was 0.84. The comparison of traditional GEP and Backtracked GEP is shown in Table 2.

Table 2. The success rate comparison (II)

	traditional	AIBCS
number of records	100	100
success rate	20%	100%

The experiments show that, with same evolution generation, backtracked GEP can avoid prematurity and gets global optimization much easier than traditional GEP. Fig. 4 gives results in the second experiment. The x-coordinate is data number, y-coordinate is values. Fig.4 shows that new algorithm get more accurate results (The dashed line is traditional GEP).

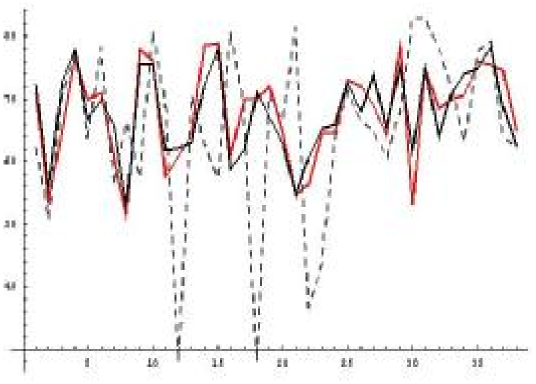

Fig. 4. The accuracy comparison of GEP and Backtracked GEP

6. Strategies for Population Diversities

As stated in Section 3, the population diversity is very important for the quality of results. The initial population needs to have as many different individuals as possible in order to better explore the search space in the further evolution [2]. The original GEP generates the initial population randomly. It is simple but does not pay attention to the diversity of the generated chromosomes.

To overcome the limitation, we propose strategies for diversifying the initial population and adjusting the mutation rate dynamically. The key ideas are as follows:

- Developing an algorithm to extract the open reading frame (ORF) of a gene without parsing the corresponding ET.
- Using the *r*-continuous-bits matching rule to evaluate the similarity between chromosomes.

– Proposing a novel adaptive mutation rate strategy for each chromosome in the evolution. Different chromosomes may be assigned different mutation rates according to their fitness values.

6.1. Improve initial population diversity

As mentioned above, the ORF is the valid part of a gene and can be got by parsing ET from left to right and from top to bottom. Moreover, a gene is the genotype of a GEP individual. Thus, in our study we evaluate population diversity from the aspect of genotype instead of phenotype.

The main reasons for us to adopt this method are on the following: (1) all the genetic operators are conducted on the genotypes in GEP; (2) the genotypes are strings with fixed length. It is easy to implement matching algorithm on strings; and (3) the phenotypes are tree structures. It is difficult to evaluate the similarity between two phenotypes. The case will become more difficult when taking the commutative operators into account.

It is common for GEP genes to have noncoding regions downstream from the termination point. However, it is unreasonable to consider these noncoding regions when evaluating the similarities among genes, since they do not interfere with the product of expression. Thus, we measure the difference among different genes based on their ORFs rather than the entire genotypes.

Although the conversion from an ET into an ORF can be accomplished by recording the nodes from left to right in each layer of the ET in a top-down fashion to form the string, it is a time-consuming process. To deal with this problem, we develop an algorithm to extract the ORF of a gene without parsing the corresponding ET. The main idea of the algorithm is based on the following facts: (1) the start site is always the first position of a gene in GEP; (2) a gene is mapped into an ET according to a wide-first procedure, and (3) a branch of the ET stops growing when the last node in this branch is a terminal.

Observation 1. For each function in the gene, there are as many symbols as there are arguments to that function downstream from it.

Based on Observation 1, we can extract the ORF from a gene without mapping genotype to phenotype. Let the variable *length* be the number of symbols at least belongs to the ORF downstream from the scanning symbol. The process begins with reading the first symbol in the gene and assigning its arity to *length*. If the *length* does not equal to 0, the process continues with reading the next symbol. Since the process has read another symbol, the value of *length* should minus 1. If the second symbol is also a function, we add its arity to *length*, else do nothing. The process is repeated until the value of length equals to 0. Figure 5 shows an instance of extracting the ORF from "+a*babab".

+a*-babab									
	step	0	1	2	3	4	5	6	7
	length	1	2	1	2	3	2	1	0
	input symbol		+	a	*	-	b	a	b

Fig. 5. An example of extracting an ORF

6.2. The strategy of adaptive mutation

Although there are several genetic operators to create the necessary genetic diversification that allows evolution in the long run, mutation is the single most efficient genetic operator to modify individuals in GEP [10, 12]. Mutations can occur anywhere in the chromosome. However, a particularity of this operator is that some integrity rules must be obeyed to avoid syntactically invalid individuals. In the head of a gene, both terminals and functions are permitted (except for the first position, where only functions are allowed); in the tails terminals can only change into terminals.

If a function is mutated into a terminal or vice versa, or a function of one argument is mutated into a function of more arguments or vice versa, the ET is modified drastically [1]. Therefore, the value of mutation rate is important for evolving the optimal solution in a running. Even the fitness of individuals (candidate solutions) varies a lot; each of them has the same probability to survive. In other words, in original GEP, the modification probability for every individual is equal and no individual has more viability than the others.

However, the common sense tells that *the fitter the being is, the higher survival probability it has*. Thus, the individual with higher fitness value should be assigned lower mutation rate. In this section, we discuss an adaptive mutation rate strategy, in which the fitness of each individual is considered. Given individual *I*, let p_m denote the mutation rate of *I*. That is,

$$p_m = (1 - fitVal \,/\, fitMax)*(p_m\_max - p_m\_min)+p_m\_min$$

where *fitVal* denotes the fitness value of *I*, *fitMax* denotes the fitness value when individual is the best solution, p_m_max is the maximum mutation rate for *I* and p_m_min is the minimum mutation rate for *I*. Both p_m_max and p_m_min are assigned by user before running. And they satisfy: $1.0 \geq p_m\_max > p_m\_min \geq 0.0$.

From above equation, the mutation rate for each individual can vary from p_m_max to p_m_min in evolution according to the value of fitness. The fitter the individual is, the lower the mutation rate is. Thus, the opportunity for losing individuals with higher fitness in population decreases, while individuals with lower fitness are more likely to undergo mutation operator, which can modify individuals drastically. If p_m_max equals p_m_min, the mutation rate remains invariable as in original GEP.

Theoretically the thought of adaptive mutation rate can be applied to other genetic operators in the same way. However, we do not apply this strategy to transposition and recombination. The reasons are stated as follows:

a) As stated in [10], although other genetic operators can be and are regularly used in GEP both for practical and theoretical reasons, mutation has a tremendous creative power and, indeed, this operator alone is more than sufficient to evolve solutions to virtually all problems.

b) In original GEP algorithm, individuals are selected according to their fitness by the well-known roulette-wheel selection with elitism and modified by genetic operators, which are performed in an orderly fashion, starting with replication and continuing with mutation, transposition, and recombination. Thus, while calculating mutation rates, fitness can be got from previous step. However, if we apply the thought of adaptive mutation to other operators, fitness of individuals should be evaluated once more. As individuals have been modified by other genetic operators implemented previously.

c) C. Ferreira studied the transforming power of mutation, transposition, and recombination in [10]. She pointed out that the finger-shaped plot observed for mutation, is very different from plots obtained both for transposition and recombination. Thereby the adaptive strategy is not suit for transposition and recombination.

Based on above reasons, we just apply the adaptive strategy to mutation operator instead of all genetic operators.

6.3. Performance evaluation

We made a similar experiment as C. Ferreira did in [10]. The test function, $y = a^4 + a^3 + a^2 + a$, was relatively simple, as it can be exactly solved using relatively small populations and relatively short evolutionary times. A set of 10 random fitness cases chosen from the interval [-10, 10] was used. The training data set remained the same for each running in order to minimize any evolution difference caused by training data.

To demonstrate the effectiveness of the strategy of initial population diversity, we implemented the original GEP algorithm (O-GEP) as well as the GEP algorithm with the new strategy (I-GEP). Due to the stochastic nature of GEP, the success rate (p_s) and the number of generations necessary to find the best solution (gen_{best}) of each algorithm were evaluated 100 independent runs and the average values were reported. The similarity threshold (t_s) was assigned 7 in this problem. The results of this experiment are shown in Table 3.

Table 3. Results of O-GEP and I-GEP for 100 runs of the synthetic problem

	O-GEP	I-GEP
p_s	96%	99%
gen_{best}	21.8	16.6

Next, we carried out another performance compare between the GEP algorithm with the strategy of adaptive mutation rate and the original GEP algorithm. For the strategy of adaptive mutation rate, p_m_max and p_m_min were assigned as 0.1 and 0.044 respectively. The original GEP algorithm was evolved in the case of mutation rate equals 0.1 and 0.044 respectively in this experiment. In order to observe the effects of the strategy fairly, only mutation operator of all genetic operators was used in the algorithms. So we denote them as A-GEP' and O-GEP' respectively. In addition, we added the strategy of initial population diversity to A-GEP' so as to observe the effect of the proposed strategies implemented simultaneously. This algorithm is denoted as A&I-GEP'. The results of them are shown in Table 4.

Table 4. Results of O-GEP', I-GEP' and A&I-GEP' for 100 runs of the synthetic problem

	O-GEP'$_{Rm=0.044}$	O-GEP'$_{Rm=0.1}$	I-GEP'	A&I-GEP'
p_s	92%	96%	97%	97%
gen_{best}	34.6	29.7	24.8	23.1

As shown in Table 3 and Table 4, the performance of GEP algorithm can be enhanced remarkably when using our proposed strategies.

7. Conclusion

GEP is a powerful function mining tool with simple coding and wide application area. In the past year, we developed some strategies to make traditional GEP more powerful, i.e. (a) Genetic Modifying Algorithm in GEP (Trans-gene). By injection gene segment, it guides the evolution direction, controls knowledge discover process.

(b) Overlapped gene expression. It borrows the idea of overlap gene expression from biological study, introduces overlapped gene expression, save space for gene expression. (c) Backtracked GEP. It is enlightened from atavism in biology. We propose the backtracking GEP algorithms, designed Geometric Proportion Increased Checkpoint Sequence and Accelerated Increased Checkpoint Sequence to restrict the backtracking process. (d) Population diversity strategy and adaptive mutation rate strategy for improving the efficiency of GEP in this paper. (e) Extensive experiments show that these strategies respectively boost the performance of GEP by one or two magnitudes.

For future work, we plan to apply our strategies to real-life applications. Therefore, more work such as determining proper parameters for GEP and our methods, will be considered.

References

[1] C. Ferreira. Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. Complex Systems, 2001, 13(2): 87-12

[2] C. Ferreira: Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence. Angra do Heroismo, Portugal, 2002.

[3] C. Ferreira: Function finding and a creation of numerical constants in gene expression programming, In: Advances in Soft Computing, Engineering Design and Manufacturing. Berlin: Springer-Verlag (2003) 257–266.

[4] Chi Zhou, Weimin Xiao, Thomas M. Tirpak and Peter C. Nelson: Evolution Accurate and Compact Classification Rules with Gene Expression Programming. IEEE Transactions on Evolutionary Computation. Vol. 7, 6 (2003) 519–531.

[5] Zuo J, Tang CJ and Zhang TQ: Mining predicate association rule by gene expression programming. In: Proc of the 3rd Int'l Conf for Web Information Age 2002(WAIM02). LNCS 2419. Berlin: Springer-Verlag (2002) 92–103

[6] Zuo J, Tang CJ, Li C, et al: Time Series Prediction based on Gene Expression Programming. In: Proc of the 5th Int'l Conf for Web Information Age 2004 (WAIM04). LNCS 3129. Berlin: Springer-Verlag (2004) 55–64

[7] Duan L, Tang CJ, Zuo J, et al: An Anti-noise Method for Function Mining Based on GEP. Chinese Journal of Computer Research and Development (2004) 41(10): 1684–1689

[8] Jing Peng, Chang-jie Tang, Jing Zhang, Chang-an Yuan: Evolutionary Algorithm Based on Overlapped Gene Expression, In: Proc of CNC2005. LNCS 3612. Berlin: Springer-Verlag (2005) 194–204

[9] Zhong YX, Tang CJ, Chen Y, Duan L and Wei DG: Improve KDD Efficiency of Gene Expression Programming by Backtracking Strategy, Chinese Science Paper Online. http://www.paper.edu.cn No. 200505-193

[10] C. Ferreira: Mutation, Transposition, and recombination: An Analysis of the Evolutionary Dynamics. 4th Int'l Workshop on Frontiers in Evolutionary Algorithms, Research Triangle Park, North Carolina, USA (2002) 614–617

[11] http://www.amstat.org/publications/jse/jse_data_archive.htm

[12] Heitor S. Lopes and Wagner R. Weinert: EGIPSYS: An Enhanced Gene Expression Programming Approach for Symbolic Regression Problems. International Journal of Applied Mathematics and Computer Science. Vol. 14, 3(2004): 375–384.