Investigation of Constant Creation Techniques in the Context of Gene Expression Programming

Xin Li[1], Chi Zhou[2], Peter C. Nelson[1], Thomas M. Tirpak[2]

[1] Artificial Intelligence Laboratory, Department of Computer Science,
University of Illinois at Chicago, Chicago, IL 60607, USA
{xli1, nelson}@cs.uic.edu
[2] Physical Realization Research Center of Motorola Labs,
Schaumburg, IL 60196, USA
{Chi.Zhou, T.Tirpak}@motorola.com

Abstract. Gene Expression Programming (GEP) is a new technique of Genetic Programming (GP) that implements a linear genotype representation. It uses fixed-length chromosomes to represent expression trees of different shapes and sizes, which results in unconstrained search of the genome space while still ensuring validity of the program's output. However, GEP has some difficulty in discovering suitable function structures because the genetic operators are more disruptive than traditional tree-based GP. One possible remedy is to specifically assist the algorithm in discovering useful numeric constants. In this paper, the effectiveness of several constant creation techniques for GEP has been investigated through two symbolic regression benchmark problems. Our experimental results show that constant creation methods applied to the whole population for selected generations perform better than methods that are applied only to the best individuals. The proposed tune-up process for the entire population can significantly improve the average fitness of the best solutions.

1 Introduction

First introduced by Candida Ferreira in 2001, Genetic Expression Programming (GEP) [4] is a new technique for the creation of computer programs. In GEP, computer programs are represented as linear character strings of fixed length (called *chromosomes*) which, in the subsequent fitness evaluation, can be expressed as expression trees (ETs) of different sizes and shapes. The search space is separated from the solution space, which results in unconstrained search of the genome space while still ensuring validity of the program's output. Due to its linear fixed-length genotype representation, genetic manipulation becomes much easier. Thus, compared with traditional GP, the evolution of GEP gains more flexibility and power in exploring the entire search space. GEP methods have performed well for solving a large variety of problems, including symbolic regression, optimization, time series analysis, classification, logic synthesis and cellular automata, etc. [2]. Zhou, et al. [5, 6] applied a different version of GEP and achieved significantly better results on multi-category pattern classification problems, compared with traditional machine learning methods and GP classifiers. Instead of the original head-tail method [2], their GEP implementation used a chromosome validation algorithm to dynamically determine the feasibility of any individual generated, which results in no inherent restrictions in the types of genetic operators applied to the GEP chromosomes, and all genes are treated equally during the evolution. The work presented in this paper is based on this revised version of GEP.

Despite its flexible representation and efficient evolutionary process, GEP still has difficulty discovering suitable function structures, because the genetic operators are more disruptive than traditional tree-based GP, and a good evolved function structure is very likely to be destroyed in the subsequent generations. Different tentative approaches have been suggested, including multi-genetic chromosomes, special genetic operators, and constant creation methods [2]. Our attention was drawn to constant creation methods due to their simplicity and the potential benefits. It is assumed that local search effort for finding better combinations of numeric constants on top of an ordinary GEP process would help improve the fitness value of the final best solution. In this paper, we propose five constant creation methods for GEP and have tested them on two typical symbolic regression problems. All of these methods are variants of two basic constant creation methods for a single chromosome, namely creep mutation and random mutation. Experimental results have demonstrated that basic constant creation methods performed on the whole population for selected generations are preferred than those performed only for the best individuals, and that tune-up processes applied to the whole population can achieve meaningful improvement in the average fitness value of the best solutions.

The next section of this paper gives an overview of related work. Section 3 explains the constant creation methods that we investigated. The experiment design and setup are described in section 4. Section 5 summarizes the experimental results and gives a qualitative analysis of the applicability of the proposed constant creation methods for GEP. Section 6 presents some conclusions and ideas for future work.

2 Related Work

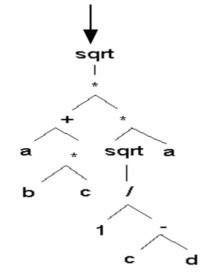
2.1 A Brief Overview of Gene Expression Programming (GEP)

As is the case with GP, when using GEP to solve a problem, generally five components, i.e., the function set, terminal set (including problem-specific variables and pre-selected constants), fitness function, control parameters, and stop condition need to be specified. Each chromosome in GEP is composed of a fixed length of character strings, which can be any element (called *gene*) from the function set or the terminal set. Using the function set {+, -, *, /, sqrt} and the terminal set {a, b, c, d, 1}, Fig. 1 gives an example GEP chromosome of length fifteen. This is referred to as *Karva notation*, or *K-expression* [4]. A K-expression can be mapped into an ET following a width-first procedure and be further written in a mathematical form as shown in Fig.

1. The conversion of an ET into a K-expression can be accomplished by recording the nodes from left to right in each layer of the ET in a top-down fashion.

Chromosome: sqrt.*.+.*.a.*.sqrt.a.b.c./.1.-.c.d .

ET:



Mathematical form:

$$\longrightarrow \sqrt{(a+bc)*a\sqrt{\frac{1}{c-d}}}$$

Fig. 1. An example of GEP chromosome, the corresponding expression tree and the mathematical form. The character "." is used to separate individual genes in a chromosome

A chromosome is valid only when it can map into a legal ET within its length limit. Therefore all of the chromosomes randomly generated or reproduced by genetic operators are subject to a validity test procedure, in order to prevent illegal expressions from being introduced into the population [6].

The GEP algorithm begins with the random generation of linear fixed-length chromosomes for the initial population. Then the chromosomes are represented as ETs, evaluated based on a pre-defined fitness function, and selected by fitness to reproduce with modification. The individuals of this new generation are, in their turn, subjected to the same developmental process until a pre-specified number of generations are completed, or a solution has been found. In GEP, the selection procedures are often determined by roulette-wheel sampling with elitism [12] based on individuals' fitness, which guarantees the survival and cloning of the best individual to the next generation. Variation in the population is introduced by applying one or more genetic operators, i.e., crossover, mutation and rotation [2], to selected chromosomes, which usually drastically reshape the corresponding ETs.

2.2 Constant Creation

Research and discussion on constant creation issues have continued for some time in GP research circles, as it is well known that GP has difficulty discovering useful numeric constants for the terminal nodes of s-expression trees [1, 3]. This is one of the major obstacles that stand in the way of achieving greater efficiency for complex GP applications. A detailed analysis of the density and diversity of constants over generations in GP was given by Ryan and Keijzer in [8]. They also explored the applicability of improving the search performance of GP through small changes, by introducing two simple constant mutation techniques, namely creep mutation, a stepwise mutation that only permits small changes, and uniform/random mutation that chooses a new random value uniformly from some specified range. Uniform mutation is reported to have considerably better performance. Some other enhancements to the constant creation procedure in GP can be categorized as local search algorithms. Several researchers have tried to combine hill climbing [1, 2], simulated annealing [1, 11], local gradient search [3] and other stochastic techniques to GP to facilitate finding useful constants for evolving solutions or optimizing extra parameters. Although meaningful improvements have been achieved, these methods are somewhat complicated to implement compared with simple mutation techniques. Furthermore, an overly constrained local search method would possibly reduce the power of the "free-style" search inherent in the evolutionary algorithms. A novel view of constant creation by a digit concatenation approach is presented in [9] for Grammatical Evolution (GE). Most recently, a new concept of linear scaling is introduced in [10] to help the GP system concentrate on constructing an expression that has the desired shape. However, this method is suitable for finding significant constants for evolved expressions that are approximately linearly related to their corresponding target values, but it is generally ineffective for identifying other candidates with good function shapes.

Since the invention of GEP, constant creation techniques have received attention in the research literature. Ferreira introduced two approaches for symbolic regression in the original GEP [7]. One approach does not include any constants in the terminal set and relies on the spontaneous emergence of necessary constants through the evolutionary process of GEP. The other approach involves the ability to explicitly manipulate random constants by adding a random constant domain *Dc* at the end of chromosome. Experiments have shown that the first approach is more efficient in terms of both accuracy of the evolved models and computational time for solving problems.

3 Constant Creation Methods for GEP

The way we handle numeric constants in GEP is as follows: several constant symbols are selected into the terminal set at the beginning of a GEP run. These constant symbols will evolve with function symbols and other terminals to produce candidate solutions. This mechanism makes it possible to obtain desirable set of constants by evolving substructures composed of only functions and constant terminals. This search for desirable constants is concurrent with the search for desirable function structures. Therefore, the problem of finding useful constants in GP now also applies to GEP. In order to fit some constant coefficients the chromosome has to keep structurally changing, even though a function structure similar to the optimal solution may have been previously discovered. Our investigation of constant creation methods in GEP was performed with the following assumptions:

- The proposed methods for constant creation should be as simple as possible, so that they will not dominate the fundamental evolutionary process of GEP, which should play the leading role in finding an optimal solution. Therefore, we have chosen two basic constant creation methods for a single chromosome, namely creep mutation and random mutation, as the basis for our proposed methods.
- Local search should be biased towards optimality, which means that mutation proceeds only when it actually improves the fitness value of the chromosome.
- Different variations of the basic constant creation methods should be examined

as independent approaches, since the manner in which these basic methods are applied will result in difference in the exploration of the search space.

3.1 Definition of Basic Constant Creation Methods

The two basic constant creation methods we have employed for a single chromosome are similar to those adopted in [8] and have been described in the literature as creep mutation and random mutation. However, in our approach, we only preserve mutations which actually contribute to an improvement in the fitness of the chromosome against the application of mutations to all constants present in the chromosome as used in [8]. We first define a *single-point constant mutation* in GEP as follows:

> For the initial configuration of GEP, a list of constants is selected and sorted into the terminal set as seeds to produce any other desirable constants during evolution. A *single-point constant mutation* changes a single constant gene in a chromosome to another constant gene. If the new constant is randomly selected from the constant gene list, it is a *random mutation*; if the new one is restricted to be selected from the neighboring constants of the current one, it is a *creep mutation*.

Then we define a *GEP constant (creep or random) mutation operation* as below:

> For every constant gene in a chromosome, perform a single-point constant mutation (creep or random) in a greedy manner, i.e., only if the fitness of the new chromosome is improved would the mutation actually proceed with the new constant symbol substituted for the old one.

These two mutation methods for a single chromosome form the foundation of our various constant creation methods for GEP.

3.2 Constant Creation Methods for GEP

After reviewing the existing literature, we have proposed five constant creation methods (CCMs) to investigate, which are detailed as follows:

(1) Creep mutation on best individuals (CM_BST): Apply constant creep mutation to the fittest individual of each generation.

(2) Random mutation on best individuals (RM_BST): Apply constant random mutation to the fittest individual of each generation.

(3) Creep mutation for first α% generations (CM_FST): For the first α% generations, at the end of the GEP run for each generation, all individuals in the population undergo constant creep mutation.

(4) Random mutation for first α% generations (RM_FST): Same as CM_FST except that constant random mutation is used in place of creep mutation.

(5) Random mutation for generations at intervals (RM_INTV): Starting from the first generation, for generations at a certain interval *g*, all individuals in the population undergo constant random mutation at the end of the GEP run for each generation. Here the interval value *g* is chosen such that the generations subject to random mutation count α% of the total amount of generations.

For the first two methods, CM_BST and RM_BST, only the best individual of each generation is considered for tuning up the constants. This is based on the well-known practice for population-based evolutionary processes, where only the best individual is of real interest because it serves as the final solution to the problem. This is guaranteed by selection with elitism in reproduction of the population. Furthermore, if certain new constants achieve better fitness for the best individual, they should be useful components for an optimal solution. And the extra computation is limited since only one chromosome in the population applies CCM. For the CM_FST and RM_FST methods, constant mutation is performed for every individual in the population but only for the first few generations. This reflects the temporal qualities of constant mutation in GP, as examined in [8]. Namely, there is a dramatic drop-off in the number of constant mutations that contribute to the best-of-run individual as the GP procedure progresses. We conjecture a similar property for GEP. Moreover, as convergence is desired for an evolutionary process, the fluctuation of constants would be less useful or even harmful in later generations. The RM_INTV method was constructed for comparison with RM_FST to test our hypothesis that constant mutations are more beneficial at early stages of a GEP run. In the last three methods, the parameter α% is problem dependent and is usually set to a small value to avoid too much extra computation.

4 Experiments

4.1 Problem Statement

In order to test the applicability of the proposed constant creation methods, we have selected two symbolic regression problems as the test cases, both of which have been studied by other researchers in published literatures on constant creation issue in GP or GEP. The first equation (1) is a simple polynomial with coefficients of real numbers [1]. Since real numbers belong to an unlimited set, we are never able to preselect appropriate ones to participate in the evolutionary process, and instead rely on the evolutionary process itself to discover such complex constants. The purpose of conducting this experiment is to find out the performance of potential methods in helping GEP compose real numeric values. A set of twenty-one fitness cases equally spaced along the x axis from -10 to 10 are chosen for this polynomial.

$$y = x^3 - 0.3x^2 - 0.4x - 0.6 \quad . \tag{1}$$

$$y = 4.251a^2 + \ln(a^2) + 7.243e^a \quad . \tag{2}$$

The second equation (2) is a “V” shaped function which not only has real coefficients with higher precision but also exhibits complex functionality and structure. Thus, the major challenge here for GEP is to obtain a good approximation to the shape of the target function. The fitness cases for this “V” shaped function problem are the same as used in [7], namely, a set of twenty random fitness cases chosen from the interval [-1, 1] of the variable a.

4.2 Experiment Setup

We compared the performance of different CCMs under the same experiment setup. For the GEP control parameters, we used 100 for the GEP chromosome length, 500 for the population size, 1000 for the maximum number of generations, 0.7 for the crossover probability and 0.02 for the mutation probability. Though more generations usually provide greater chances to evolve a fitter solution, here we deliberately chose modestly sized GEP control parameters since we want to examine the evolutionary trend of each method under investigation, rather than obtain a perfect solution. In addition, the roulette-wheel selection with elitism is utilized as the selection method based on the fitness function calculated by (3), where $\mathrm{fitness}_i$ indicates the fitness function for the ith individual in the population, minR is the best (or minimum) residual error obtained so far and $\mathrm{ResError}_i$ is the individual's residual error. This normalizes the fitness values within the interval [0, 1], and $\mathrm{ResError}_i=0$ gives the best, where $\mathrm{fitness}_i=1$. Note this is the fitness function used for selection, and the fitness of a chromosome is measured by its residual error whose value is better when smaller.

$$fitness_i = \min R / (\min R + \mathrm{Re}\, sError_i) \ . \tag{3}$$

The terminal set includes the constants {1, 2, 3, 5, 7} and input attributes (either the variables x or a). However, due to our a priori knowledge about these two benchmark problems, different function sets were used: {+, -, *, /} for the polynomial problem and {+, -, *, /, log, exp, power, sin, cos} for the "V" shaped function problem, where *log* represents the natural logarithm, *exp* represents e^x, *power(x, y)* represents x^y, *sin* represent sine function and *cos* represents the cosine function. Furthermore, in experiments for GEP with CCMs requiring α% parameter, we set the value as 1%.

Our methods were incorporated into Java software for GEP, and experiments were conducted on a computer with Intel Pentium 4 CPU 1.70GHz and 512MB RAM using Microsoft Windows 2000.

5 Experimental Results and Discussion

Taking the stochastic behavior of the GEP process into account, each experiment was repeated for thirty independent runs, and the results were averaged. Since we utilized a greedy approach to perform constant mutation, i.e., keeping the change only if it improves the fitness, we want to first investigate the actual usage of the constant mutation for each strategy in the experiments we conducted. For those experiments in which constant mutations were actually performed, it is necessary to compare them with applications of GEP without any constant creation strategy (referred to as *plain GEP*) and examine whether the final solution's fitness was improved.

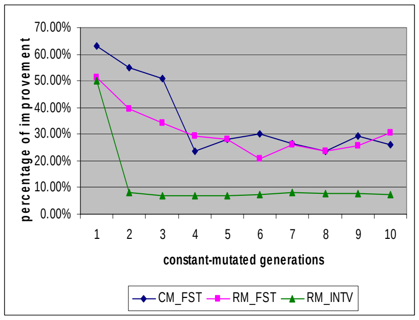
5.1 Finding 1: Not All Strategies Work

In Table 1 we summarize the percentage of chromosomes whose fitness (i.e., the residual error) has been improved (i.e., minimized) through constant mutation.

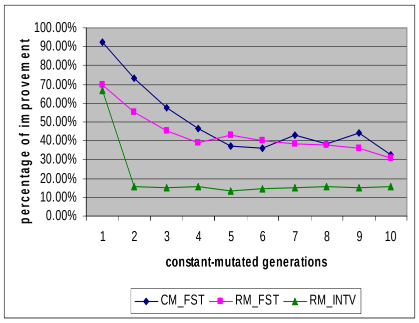
Table 1. Percentage of improved constant-mutated chromosomes. Percentage is defined as the ratio of chromosomes with fitness improved by constant mutation over the total number of constant-mutated chromosomes. Results were averaged over thirty independent runs

Name of CCM	Percentage of improved constant-mutated chromosomes	
	Polynomial	"V" Shaped function
CM_BST	0.0%	0.0%
RM_BST	0.0%	0.0%
CM_FST	35.6%	50.2%
RM_FST	31.0%	43.5%
RM_INTV	16.6%	25.7%

As shown in Table 1, GEP with constant mutation (either creep or random) applied to the best individual, never actually improved the best individual of the generation in our experiments. Consequently, except for an additional routine for checking the possibility of improving the best individuals by constant mutation, these two versions of GEP are essentially the same as plain GEP. This strongly indicates that the best individual evolved by plain GEP is already good enough, and that it is not likely to be improved by creep or random constant mutation operation. This observation also fits our assumption about GEP with CCM in section 3, namely that the GEP evolutionary process always plays the leading role in finding an optimal solution. Since the determination of whether or not to use the constant mutation method requires extra computation, but it is not likely that the GEP solution will be improved, we conclude that GEP with CM_BST or RM_BST is not a beneficial enhancement to plain GEP.



(a) The polynomial problem



(b) The "V" shaped function problem

Fig. 2. Percentage of improved individuals at constant-mutated generations. Percentage is defined as the ratio of individuals with fitness improved by constant mutation over the whole population; generations are those at which constant creation strategy is applied to the whole population. Results were averaged over thirty independent runs

In contrast, when constant mutation is applied to the whole population, as is the case in the remaining three versions of GEP with CCM, it typically improves the fitness of many of the chromosomes in each generation. In particular, GEP with CM_FST or RM_FST has a larger portion of chromosomes improved via constant mutation than GEP with RM_INTV. The percentage of individuals (out of the whole population) whose fitness is improved (via constant mutation) in a given generation is plotted in Fig. 2, according to the number of generations subject to CCM.

The curves in Fig. 2 highlight a characteristic of the fitness of average individuals in the population: the percentage of individuals that could be improved by constant mutation is prone to decrease over the generations. A qualitative analysis for this phenomenon is that in the beginning, the population is composed of dramatically diverse individuals and their corresponding functions have structures and constant coefficients combined differently from the optimal solution. Thus, constant mutation results in fitness improvement for large amounts of individuals. However as the GEP evolutionary process moves on, the population gradually settles down to be composed of sub-optimal coalitions of function structures and constants found along the way. Therefore in later generations, direct recombination of the genes in chromosomes by GEP algorithm itself is more significant in changing the fitness value than small modifications like constant mutations.

5.2 Finding 2: Some Strategies Exhibit Better Performance

The performance of GEP was examined with the constant creation strategies of CM_FST, RM_FST and RM_INTV. The experimental results are shown in Table 2, where *best residual* refers to the best residual error among all of the final best individuals from the thirty runs; *average of best residuals* is an average of all thirty final best residual errors along with an approximate 95% confidence interval; *average running time* is the running time of each GEP process averaged over thirty runs and measured in seconds; and *average tree size* refers to the average corresponding expression tree size (i.e., the number of nodes in the tree) of the best individuals over thirty runs. The following observations can be made from Table 2:

(1) The entries for *best residual* show that no single approach exhibits a predominant advantage for both problems in finding a best solution out of thirty runs.
(2) The numerical ranges for *average of best residuals* give evidence of better performance of GEP with CCMs than plain GEP.
(3) GEP with RM_FST and GEP with RM_INTV slightly outperform GEP with CM_FST with respect to *average of best residuals*.
(4) The comparable expression tree sizes suggest all approaches produce solutions of equivalent functional complexity.
(5) GEP with CCMs takes more running time than plain GEP does. We will discuss it separately in section 5.3.

To verify the second observation, we have conducted one-tailed Student's t-tests with an unequal variance assumption to determine the significance level that versions of GEP with CCM outperform plain GEP in terms of the average best residual error. Using this method, a good approach should produce a low best residual error (i.e., a high best fitness) in general. The t-test was carried out between plain GEP and each of GEP with CCMs, and the results demonstrate that their performances have significant difference for both testing problems. More specifically, three versions of GEP with CCM outperformed plain GEP with greater than 95% significance in almost all of the conducted experiments. The only exception was the test between GEP with

CM_FST and plain GEP for the “V” shaped problem, where the former only outperformed the later with around 91% significance. However, these results are strong enough for us to conclude that GEP with CCMs over the population has achieved nontrivial improvement in the fitness of the average best solution compared with plain GEP.

Table 2. Performance comparison of GEP with CCMs. GEP with CM_FST, RM_FST and RM_INTV are compared against plain GEP. Results were averaged over thirty independent runs and each entry for *average of best residuals* is an approximate 95% confidence interval

Problem	Statistics	Plain GEP	GEP with CM_FST	GEP with RM_FST	GEP with RM_INTV
Poly-nomial	Best residual	0.382	0.419	0.268	0.157
	Average of best residuals	1.261±0.213	1.007±0.116	0.992±0.178	0.966±0.167
	Average running time (s)	56	79	80	81
	Average tree size	32.3	36.3	36.9	37.3
“V” shaped	Best residual	1.065	1.013	1.110	1.038
	Average of best residuals	2.045±0.145	1.914±0.121	1.865±0.149	1.863±0.127
	Average running time (s)	62	96	90	98
	Average tree size	25.8	27.1	27.2	28.4

The third observation draws our attention to the difference among performances of these three versions of GEP with CCM in our experiments. We again conducted one-tailed Student’s t-tests with an unequal variance assumption for each pair of them. However, the test results reveal that even the most significant difference in their performance, which comes between GEP with CM_FST and GEP with RM_INTV for the “V” shaped problem, is just around 72%. Therefore, we do not have enough evidence to claim that one GEP with CCM outperforms the other in terms of the fitness of the average best solution. The indication here is that by tuning up the fitness of the whole population via constant creation methods, the performance of GEP can be effectively boosted. The individuals who are able to gain fitness improvement purely via constant mutation tend to have fitter function structure components. Meanwhile the improved fitness values of these individuals after constant mutation make them more likely to survive through multiple generations, when compared with the rest of the population. Search direction is consequently biased to these fitter function structures, which is essential for the evolution to converge to an optimal solution. It appears that specific manner in which the constant mutation methods are applied to tune the whole population, i.e., when they are applied (in the beginning or at intervals) and how they are applied (as creep or random mutation), do not have a large effect on the fitness of the final solution.

5.3 Finding 3: All Strategies Require Extra Computation

As shown in the entries for *average running time* in Table 2, our constant creation strategies require some extra computational resources. However, as noted from the experiments in [3], the use of local learning creates a bias in the structure of solutions, namely it prefers structures that are more readily adaptable by local learning. Therefore, the fitness landscape [13] is altered due to the fitness improvement of the best individual or large portion of other individuals in the population. Consequently, it is not fair to directly compare the efficiency of GEP with or without CCMs. This is particularly true when those variants of GEP that require more computation actually tend to find better solutions with higher fitness values on average. We are not able to precisely estimate how many more generations (and thus computational resources) are needed in order to make plain GEP produce an equally competitive solution as those gained by GEP with CCMs. Since the percentage of generations subject to constant mutation is adjustable, which would help minimize additional overhead of GEP with CCMs, whether or not to pick up these enhanced versions of GEP for solving a symbolic regression problem is better left to the actual preference between fitness improvement and computational cost considerations with respect to the nature of the problem at hand.

6 Conclusions and Future Work

This paper has explored a way to improve the performance of the GEP algorithm by implementing constant creation methods. The experimental results reported in this paper show that the GEP algorithm possesses a very strong capability to find or compose the most suitable combination of constants and function structures. As a result, the best individual of the generation usually exhibits the true best evaluation score and can seldom be further improved by simple local search/tuning of its constants. However, constant creation methods applied to the whole population can significantly improve the fitness of average individuals in the population, especially in early generations. On average, via this constant tune-up process, higher fitness scores have been achieved for the final best solutions with only a modest increase in the computational effort of the GEP algorithm.

In future research, we plan to further examine the proposed GEP with constant creation methods for larger scale regression problems. Furthermore, the current constant creation methods under investigation either apply the basic constant mutation techniques to the best individual of the generation or to the whole population. Experiments have shown two extreme results, where the former one seldom obtains improvement in the fitness of the best individual while the latter shows notable benefits. We infer that there possibly exists a more appropriate intermediate setup point between these two choices. For example, it may be possible to extend the constant creation methods to an elite group, which contains a set of individuals with relatively high fitness values, as compared to the remaining individuals in the whole population for a given generation. This would save some computation but still yield the advantage of improving overall fitness of final best solutions via constant mutation methods.

Acknowledgement

We appreciate the *Physical Realization Research Center of Motorola Labs* for providing funding for this research work. We would also thank Weimin Xiao for his insightful discussion on related topics.

References

1. Matthew Evett, Thomas Fernandez: Numeric Mutation Improves the Discovery of Numeric Constants in Genetic Programmin. Proceedings of the Third Annual Genetic Programming Conference. Madison, Wisconsin (1998) 66-71
2. Candida Ferreira: Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence. Angra do Heroismo, Portugal (2002)
3. Alexander Topchy, William F. Punch: Faster Genetic Programming based on Local Gradient Search of Numeric Leaf Values. Proceedings of the Genetic and Evolutionary Computation Conference. San Francisco, California (2001) 155-162
4. Candida Ferreira: Gene Expression Programming: a New Adaptive Algorithm for Solving Problems. Complex Systems, Vol. 13, No. 2 (2001) 87-129
5. Chi Zhou, Weimin Xiao, Peter C. Nelson, Thomas M. Tirpak: Evolving Accurate and Compact Classification Rules with Gene Expression Programming. IEEE Transactions on Evolutionary Computation, Vol. 7, No. 6 (2003) 519-531
6. Chi Zhou, Peter C. Nelson, Weimin Xiao, Thomas M. Tirpak: Discovery of Classification Rules by Using Gene Expression Programming. Proceedings of the International Conference on Artificial Intelligence. Las Vegas, Nevada (2002) 1355-1361
7. Candida Ferreira: Function Finding and the Creation of Numerical Constants in Gene Expression Programming. The Seventh Online World Conference on Soft Computing in Industrial Applications (2002)
8. Conor Ryan, Maarten Keijzer: An Analysis of Diversity of Constants of Genetic Programming. European Conference on Genetic Programming. Lecture Notes in Computer Science, Vol. 2610. Springer-Verlag, Berlin Heidelberg New York (2003) 404-413
9. Michael O'Neill, Ian Dempsey, Anthony Brabazon, Conor Ryan: Analysis of a Digit Concatenation Approach to Constant Creation. European Conference on Genetic Programming. Lecture Notes in Computer Science, Vol. 2610. Springer-Verlag, Berlin Heidelberg New York (2003) 173-182
10. Maarten Keijzer: Improving Symbolic Regression with Interval Arithmetic and Linear Scaling. European Conference on Genetic Programming. Lecture Notes in Computer Science, Vol. 2610. Springer-Verlag, Berlin Heidelberg New York (2003) 70-82
11. Anna I. Esparcia-Alcazar, Ken Sharman: Learning Schemes for Genetic Programming. Proceedings of Late Breaking Papers at the Annual Genetic Programming Conference. Stanford University, California (1997) 57-65
12. David E. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Pub. Co. (1989)
13. Melanie Mitchell: An Introduction to Genetic Algorithms (Complex Adaptive Systems). MIT Press (1996)