Application of Gene Expression Programming to Real Parameter Optimization

Kaikuo Xu\textsuperscript{1}, Changjie Tang\textsuperscript{1}, Rong Tang\textsuperscript{1,2}, Yintian Liu\textsuperscript{2}, Jie Zuo\textsuperscript{1}, Jun Zhu\textsuperscript{3}

\textsuperscript{1}School of Computer Science, SIChuan University, ChengDu, 610065, China
\{xuikaikuo, tangchangjie\}@cs.scu.edu.cn
\textsuperscript{2}College of Computer, Chengdu University of Information Technology, ChengDu, 610225, China
\textsuperscript{3}National Center for Birth Defects Monitoring, ChengDu, 610041, China

Abstract

Gene Expression Programming (GEP) is a new evolutionary algorithm that implements genome/phoneme representations. Despite its powerful global search ability and wide application in symbolic regression, little work has been done to apply it to real parameter optimization. A real parameter optimization method named Uniform-Constant based GEP (UC-GEP) is proposed in this paper. The main work and contributions include: (1) Compares UC-GEP with Meta-Constant based GEP (MC-GEP), Meta-Uniform-Constant based GEP (MUC-GEP), and Floating Point Genetic Algorithm (FP-GA) on optimizing seven benchmark functions, respectively. Experiment results show that GEP methods outperform FP-GA on five of the seven functions and UC-GEP reaches the global optimum on all seven functions. (2) Compares UC-GEP with both MC-GEP and MUC-GEP on optimizing Rastrigin and Griewangk with various dimensions. Experiment results also show that UC-GEP is the best among these three algorithms.

1. Introduction

This paper primarily focuses on the following optimization problem:

Minimize $f(x_1, x_2, \ldots, x_n)$ \quad ($l_i \leq x_i \leq u_i \quad i = 1, 2, \ldots, n$ and $l_i, u_i \in R$)

Real parameter optimization is a typical task for global optimization. Stochastic sampling methods have been used to real parameter optimization. Comparing with traditional search methods, Stochastic sampling methods is good at optimizing multimodal, discontinuous and non-differentiable functions. Evolutionary algorithm is an important branch of stochastic sampling methods. Genetic Algorithms (GAs) is a representative of evolutionary algorithms and have been applied to real parameter optimization and obtained good results [1, 2, 6, 7, 10]. However, it is relatively easy for GAs to be attracted to local optimum. Gene Expression Programming (GEP), proposed by Ferreira [3] in 2001, has already been applied in many fields, such as regression, classification, and neural network. It has been proved to be a powerful global search tool [3,4,5]. However, little work has been done to apply it to real parameter optimization. The paper proposes a powerful optimization method Uniform-Constant based GEP (UC-GEP) and it is compared with a GA method: Floating Point Genetic Algorithm (FP-GA) and another two GEP methods: Meta-Constant based GEP (MC-GEP), and Meta Uniform Constant based GEP (MUC-GEP).

The rest of the paper is organized as follows: Section 2 gives a detailed description of GEP and GA methods, and analyzes the characteristics of each structure. Section 3 compares various real parameter optimization methods, and Section 4 concludes the paper with some remarks.

2. Methodology

2.1. Floating Point Genetic Algorithm (FP-GA)

In FP-GA, each chromosome vector is coded as a vector of floating point numbers, with the same length as the solution vector. Each element is later initialized within a given range $[L, U]$ and the operators are carefully designed to preserve such requirements.

The mutation operator is adopted from [6]: if $(u_1, \ldots, u_n)$ is a chromosome and the element $u_k$ is selected for the mutation, the child is a vector $(u_1, \ldots, u_k, \ldots, u_n)$, where

$$u'_k = \begin{cases} u_k - (u_k, \max(u_k - M, L)) & \text{if } r = 0 \\ u_k + (u_k, \min(u_k + M, U)) & \text{if } r = 1 \end{cases}$$

Here, $r$ is a uniformly distributed Boolean value, and $M$ is the maximum mutation size. $M$ is the maximum mutation size. (LB, UB) is a randomly generated value from the domain [LB, UB] with a uniform distribution. The crossover operator is a two-point crossover similar to the one in binary coded GA. For FP-GA, all problems are solved with the single elitist strategy [7], stochastic universal sampling [8], a scaling window of 5 generations.
and a population size of 20.

Fig 1: Decoding from genotype (single gene) to a formula

2.2. GEP methods

Gene Expression Programming (GEP) introduced by Ferreira [3] is a new revolutionary member in the family of evolutionary computing. Unlike GP, simple GEP uses linear strings as its genes, and uses only one gene (single gene chromosome) as its chromosome (genotype) and expression tree as its protein (phenotype). Genotype is decoded into phenotype to express the genetic information. Then phenotype is turned into a formula for real application. For example, GEP uses a very smart method to decode chromosome (+0-1*2345) to a real number within a given range. In FP-GA, the mutated point is kept within range through certain techniques. Section 2.1 shows an example. The literatures [9,10] propose other methods for implementation. However, there are no suitable techniques for GEP yet. Thus a mapping from real number to a given region [L, U] is proposed in our work. The detailed process is described in Algorithm 1.

Algorithm 1: (Map a real number to [L,U])

Input: a real number r
Output: a real number r’ in [L,U]
1 if r ≤ L
2 temp = (r - L) mod (U - L);
3 r’ = U - temp;
4 else if r > U
5 temp = (r - U) mod (U - L);
6 r’ = L + temp;
7 else r’ = r;
8 return r’;

The organizations of GEP methods are fundamentally the same, while the differences among them still exist, which leads to different performance. They are mainly about two issues: 1 The relationship among different constant domains of various genes, the scope of ai, is ai randomly generated or specified? 2 Does the constant domain CD participate in the evolving directly or indirectly?

In order to formally explain how the population covers solution space, the following concepts are introduced:

Definition 1 (expression). In GEP, given a operator set Op = \{+, -, *, /\}, a constant set C = \{c_1, c_2, ..., c_n\}, and a real number r, a recursive function \(\phi\) is an expression of r on \{Op, C\} and r is capable of being expressed by \{Op, C\} if and only if \(r = \phi(f_1, f_2, ..., f_n, c_1, c_2, ..., c_n)\).

Definition 2 (expression space). Given a recursive function set \(\Psi\), Real number set R = \{r | r = \phi(\text{Op, C}) \land \phi \in \Psi\} is called the expression space of \{\text{Op, C}\} (ExpSpace(\Psi, \text{Op, C}), for simplicity ExpSpace(\Psi)).

Definition 3. In GEP, let the population P = \{Genes_{00}, Genes_{10}, ..., Genes_{ij}, ..., Genes_{11}, ..., Genes_{1n}, ..., Genes_{ig}\}, s be the population size, each individual composed of g genes, b be the length of the gene head, the function set F = \{+, -, *, /\}, and the length of constant domain C_p be n. The expression space of \{F, C_p\} is defined as: ExpSpace(\Psi) = \text{ExpSpace}(\text{Genes}_{00}, \text{Genes}_{10}, \text{Genes}_{20}) \times \text{ExpSpace}(\text{Genes}_{01}, \text{Genes}_{11}, ..., \text{Genes}_{0n}) \times \text{ExpSpace}(\text{Genes}_{1g}, \text{Genes}_{2g})\} and ExpSpace(\text{Genes}_{00}, \text{Genes}_{10}, ..., \text{Genes}_{ig}) = \bigcup_{i=1}^{s} \text{ExpSpace}(\text{Genes}_{ij}),(i = 0, 1, ..., g).

Lemma 1. Let the cardinal number of ExpSpace(P) be |ExpSpace(P)|, then |ExpSpace(P)|

\[= \prod_{j=0}^{g} |\text{ExpSpace}(\text{Genes}_{0j}, \text{Genes}_{1j}, ..., \text{Genes}_{ij})|.

2.2.1 Meta-Constant based GEP (MC-GEP)
Zuo and Tang introduced MC-GEP in [5] to solve the problem of time-series prediction with following features: all genes share a same constant domain C_p specified manually. It should be reasonable enough so that GEP
expression may express all possible values in the specified intervals. And its resulting distribution must be dense. In our work, \( C_D \) is set as \((0.1315, 0.2128, 0.3443, 0.5571, 0.9015, 1.4588, 0.3605, 3.8195, 6.1804, 10.0007)\) and \( n = 10 \). For more information on the specification of \( C_D \), refer to [5] please. In the process of evolving, \( C_D \) is kept unchanged. Thus the evolving ability mainly depends on the evolution of chromosomes.

**Lemma 2.** In the same context of Definition 3, let the expression space in each identical run be \( \text{ExpSpace}(P)(i = 1, 2, \ldots, t) \), then

\[
1) \quad \max(\text{ExpSpace}(P)_{\text{MC-GEP}}) = \left( \sum_{i=0}^{h} 4^i \cdot n^{i+1} \right)^g
\]

2) \( \text{ExpSpace}(P)_{\text{MC-GEP}} \) is invariable.

**Lemma 3.** In the same context of Definition 3, let the expression space in each identical run be \( \text{ExpSpace}(P)(i = 1, 2, \ldots, t) \), then

\[
1) \quad \max(\text{ExpSpace}(P)_{\text{MUC-GEP}}) = \left( \sum_{i=0}^{h} 4^i \cdot n^{i+1} \right)^g
\]

2) \( \text{ExpSpace}(P)_{\text{MUC-GEP}} \) is invariable.

**2.2.2 Meta-Uniform-Constant based GEP (MUC-GEP)**

MUC-GEP is a variation of MC-GEP. The only alteration is that \( C_D \) in MUC-GEP is generated randomly using a uniform probability distribution. In our work, \( LB \) is set as \(-1\), \( UB \) is set as \( 1 \), and the value of the expression tree is set to be the maximum allowed.

All the lemmas and theorem in this paper are proved by mathematical induction method on website http://cs.scu.edu.cn/~xukaikuo/appendix1.doc.

**2.2.3 Uniform-Constant based GEP (UC-GEP)**

Each gene of UC-GEP has its own constant domain; all elements of \( C_D \) are generated randomly using a uniform distribution. In our work, \( LB \) is set as \(-1\), \( UB \) is set as \( 1 \), and \( n \) is set as \( 10 \). One significant characteristic of UC-GEP is that a genetic operator called Insertion Sequence (IS) transposition is used to evolve \( C_D \). This operator is similar to that in simple GEP: the transposable elements are fragments of \( C_D \) that can be activated and be able to jump to another place in the same constants domain. Since any sequence in \( C_D \) can become an IS element, these elements are randomly selected throughout the constants domain. The transposition is copied and the copy is then inserted at a randomly chosen point in \( C_D \). Unlike simple GEP, the first position is allowed in UC-GEP. Let a constant domain is \( X \), corresponding to a vector \( x = (x_0, x_1, \ldots, x_{n-2}, x_{n-1}, x_n) \), the starting position of IS elements is \( n - 2 \), the length is \( 3 \), and the target position is \( 1 \). Then its child corresponds to a vector \( (x_0, x_{n-2}, x_{n-1}, x_n, x_1, \ldots, x_{n-3}) \). Typically, a transposition rate \( p_t \) of 0.1 and a set of three IS elements of different length are used. This operator can make the constant domain evolve independently so that \( C_D \) participates in the evolving directly.

**Theorem 1.** In the same context of Definition 3, let the expression space in each identical run be \( \text{ExpSpace}(P)(i = 1, 2, \ldots, t) \), then

\[
1) \quad \max(\text{ExpSpace}(P)_{\text{UC-GEP}}) = \left( \sum_{i=0}^{h} 4^i \cdot n^{i+1} \right)^g
\]

2) \( \text{ExpSpace}(P)_{\text{UC-GEP}} \) is a variable.

Note Theorem 1 shows that UC-GEP covers more solution space than MUC-GEP and MC-GEP. According to Lemma 2 and Lemma 3, MUC-GEP and MC-GEP have equal expression ability. However MC-GEP is less stochastic either MUC-GEP or UC-GEP.

For GEP methods, the basic parameter setting is shown in Figure 2. For all the experiments of UC-GEP, the constants domain specific IS transposition rate is 0.8, and the different lengths of three IS elements are \( \{1, 2, 3\} \).

In our experiments, all related algorithms are implemented by modifying the zGEP program provided by Zuo [11]. When the size of head is equal to zero, GEP is indeed a simulation of GA. Thus, zGEP is also modified to implement FP-GA in our experiments.

**3. Experiments**

The empirical study first compares GEP methods and FP-GA; then carries out comparisons among GEP methods. Nine functions are chosen as the benchmark functions for our two experiments. Part of their properties is listed in Figure 3. Among them, functions from f1 to f7 are for experiment 1, while functions f8 and f9 are for experiment 2.

**3.1. Comparison between GEP methods and FP-GA**

For all functions except F7, each experiment consists of running the algorithm for 20000 function evaluations. For function F7, 100000 function evaluations are used. The best performance is the smallest value of the objective function obtained over all function evaluations.

For FP-GA, 1000 runs are conducted at each combination of a mutation size, a mutation rate, and a crossover rate. The mutation rates range from 0.05 to 0.3 in steps of 0.05, mutation sizes range from 0.1 to 0.3 in steps of 0.1, and crossover rates range from 0.1 to 0.9 in steps of 0.1. Thus, for each function, 162 combinations of mutation rate, mutation size and crossover rate are experimented. For both MC-GEP and MUC-GEP, 1000 runs are conducted at each mutation rate from 0.05 to 0.3 in steps of 0.05. Thus, for each function, 6 mutation rates are experimented. For UC-GEP, 1000 experiments are conducted at each combination of a mutation size and a IS transposition rate. The mutation rates range from 0.05 to 0.3 in steps of 0.05, and IS rates range from 0.1 to 0.8 in steps of 0.1.
Thus, for each function, 48 combinations of mutation rate, and IS rate are experimented. For each function and each type of algorithm (FP-GA, MC-GEP, MUC-GEP, and UC-GEP), the parameter combinations that give the optimal performance measure are shown in Figure 4. The standard deviation is computed over the 1000 runnings at that parameter combination.

On 5 out of the 7 functions, GEP methods outperform FP-GA. The two exceptions are F3 and F4. Although they are both unimodal (i.e., containing only one optimum), special difficulties are with them: F3 is a representative of the problem of flat surfaces, which does not give any information of the favorable direction. F4 is a simple unimodal function padded with noise. The Gaussian noise makes sure that the algorithm never gets the same value on the same point. So it demonstrates that GEP methods are not good at overcoming these two difficulties. On all the three multimodal functions, GEP methods perform better than FP-GA. GEP methods reach the global optimums, and are much more stable than FP-GA. This indicates that multimodal surfaces pose no particular problem for GEP methods. From experiment results only UC-GEP reaches the global optimums on all the 7 functions (On F4, the result is negative, which can be taken as global optimum).
3.2. Comparison among GEP methods

To further compare the performance of GEP methods on multimodal problems, Rastrigin and Griewangk are chosen as the benchmark functions. The dimensions of the functions range from 2 to 12 in steps of 2. The parameters for GEP are set as shown in Table 2. Each experiment is executed 50 times. As a measure of performance, the average number of generations is considered as the GEP methods require to generate a solution with a certain high fitness value (called the threshold). The average number of generations is obtained by executing the experiment repeatedly (in our case, 50 times) with different and randomly chosen initial populations. The performance of GEP methods is also evaluated in terms of the number of runs for which the GEP methods get stuck at a local minimum. When a GEP method fails to reach the global optimum after a sufficiently large number of generations, it is concluded to have gotten stuck at a local optimum. The experiment results are presented in Figure 5. Table 5 shows that UC-GEP outperforms MUC-GEP on all dimensions while the performance of MUC-GEP is better than MC-GEP. This proves our initial conclusion in section 2.2.3. For MC-GEP, the number of instances for which it gets stuck increase sharply along the increasing of the dimensions, and the average number of generations is much higher. So MC-GEP is not suitable for practical applications. On the contrary, the performance of UC-GEP is hardly affected by the increasing dimensions: the number of instances for which it gets stuck keeps almost unchanged and the average number of generations is much lower. To overall sum up, UC-GEP is proved to be the most suitable for high dimensional problem among the three GEP methods.

4. Conclusions

UC-GEP is presented for solving real parameter problems. The key feature of this approach is the structure of the chromosome: each gene has a constants domain \( C_D \); all the elements of \( C_D \) are generated randomly using a uniform distribution; special genetic operator IS is applied to \( C_D \). Experiments are performed to check the performance of GEP methods. The following important results by applying GEP to solving real parameter optimization problem are obtained from our work: (1) Comparing with FA-GA, GEP methods perform better on optimizing multimodal problems, while they are not suitable for solving flat surface problems and problems with noisy. (2) UC-GEP is the best among the GEP methods in solving high dimensional problems at least for the given conditions.

5. References