Complex & Intelligent Systems (2021) 7:1195–1206
https://doi.org/10.1007/s40747-020-00241-5

ORIGINAL ARTICLE

FMCGP: frameshift mutation cartesian genetic programming

Wei Fang[1] · Mindan Gu[1]

Received: 24 July 2020 / Accepted: 25 November 2020 / Published online: 12 January 2021

Abstract
Cartesian Genetic Programming (CGP) is a variant of Genetic Programming (GP) with the individuals represented by a two-dimensional acyclic directed graph, which can flexibly encode many computing structures. In general, CGP only uses a point mutation operator and the genotype of an individual is of fixed size, which may lead to the lack of population diversity and then cause the premature convergence. To address this problem in CGP, we propose a Frameshift Mutation Cartesian Genetic Programming (FMCGP), which is inspired by the DNA mutation mechanism in biology and the frameshift mutation caused by insertion or deletion of nodes is introduced to CGP. The individual in FMCGP has variable-length genotype and the proposed frameshift mutation operator helps to generate more diverse offspring individuals by changing the compiling framework of genotype. FMCGP is evaluated on the symbolic regression problems and Even-parity problems. Experimental results show that FMCGP does not exhibit the bloat problem and the use of frameshift mutation improves the search performance of the standard CGP.

Keywords Cartesian genetic programming · Evolutionary computation · Frameshift mutation

Introduction

Genetic programming (GP) refers to the automatic evolution of computer programs [18]. In the past 40 years, a lot of variants of GP have been proposed [1,15] and a variety of applications of GP have been studied in computer science [4,30], biomedical [11], chemistry [2], etc. Miller and Thomson proposed Cartesian Genetic Programming (CGP) [25,29], which can flexibly encode many computing structures and avoid the bloat problem in GP [21,23].

The individuals in CGP have a fixed-length genotype represented by a two-dimensional acyclic directed graph. In [21], the individuals in CGP are proposed to be of variable-length genotype to avoid the bloat problem. However, the variable-length genotype by changing the number of redundant nodes may lead to poor evolutionary fitness and extra processing [21]. In general, crossover operator in CGP is found to be disruptive to individuals [19]. Therefore, in the standard CGP, variable-length genotype is not used and the mutation operator is the unique evolutionary operator.

However, the fixed-length genotype may raise the problem that the size of CGP individual network structure, which means the length of genotype, needs to be specifically designed by the user according to the problems. Individual genotypes that are too long or too short may affect the efficiency of evolution [5]. In addition, the offspring are generated by point mutation in the parents, which means that only a few genes have changed in the offspring compared to the parent. And if the mutation occurs at inactive nodes, the fitness of the offspring individual cannot be different from the parent, and then results in increasing the computation cost. Due to the insufficient diversity, the convergence speed is dependent on the initial population.

To make individual genotype length adaptive to different problems and increase population diversity, we propose the frameshift mutation in CGP, which is inspired by the DNA mutation mechanism in biology. The proposed algorithm is called Frameshift Mutation Cartesian Genetic Programming (FMCGP). The frameshift mutation changes the compiling framework of genotype to increase diversity by randomly insertion or deletion of nodes. The individual in FMCGP, therefore, has variable-length genotype.

The structure of this paper is organized as follows. Section 2 describes the standard CGP and related work

✉ Wei Fang
fangwei@jiangnan.edu.cn

1 Jiangsu Provincial Engineering Laboratory of Pattern Recognition and Computational Intelligence, Department of Computer Science and Technology, Jiangnan University, Wuxi, China

on genetic operators in CGP. Section 3 introduces the frameshift mutation in DNA in biology and the introduction of frameshift mutation in FMCGP. Experimental results of two kinds of regression problems are given in Sect. 4. Section 5 discusses the population diversity and bloat problem of FMCGP. Conclusions and further work are drawn in Sect. 6.

Related work

Standard CGP

Encoding of standard CGP

In CGP, individuals are represented in the form of a two-dimensional directed acyclic graph [29]. The genotype of an individual is represented by integers. One gene, which is used to represent the function of the node, is called *function gene*. The remaining genes, called *connection gene*, are used to indicate where the node gets its data [22]. The selected functions are set by the user and are listed in the function lookup list. The nodes in the graph take input data from the forward nodes or program input. Some nodes ignored in the phenotype are called inactive nodes and the others are called active nodes [24]. The length of genotype is fixed, since the number of nodes in the CGP network is fixed. However, the length of the phenotype can be any value from zero to the total number of nodes, which depends on the connection of the internal nodes of the network [22].

There are three parameters in CGP that should be determined by the users, which are the number of columns N_c, the number of rows N_r, and the levels-back l [24]. The parameters N_c and N_r determine the arrangement of the CGP network, and the maximum allowable number of nodes $L_n = N_c * N_r$. The parameter l defines the range of columns of nodes that each node can get input data from. If $l = 1$, each node can only connect to the node in the previous column. When $l = N_c$, the nodes of the full graph can be freely connected forward.

The evolution strategy used in CGP is the $(\mu + \lambda)$, where μ refers to the number of individuals selected from the population to generate offspring in each generation and usually takes the value 1, and λ refers to the number of offspring individuals to produce in each generation with the value 4 in general [19].

Allelic constraints

The values that genes can take are highly constrained in CGP. When genotypes are initialized or mutated, these constraints should be obeyed [22]. The function genes of the individuals in CGP are presented by F, the connection genes are presented by C and the output genes are presented by O. The function genes F should follow:

$$0 \leq F \leq n_f, \tag{1}$$

where n_f refers to the number of allowed primitive node functions. The connection genes C should meet the requirement:

$$\begin{cases} n_i + (j-l)n_r \leq C_{ij} \leq n_i + jn_r & j \geq l \\ 0 \leq C_{ij} \leq n_i + jn_r & j < l, \end{cases} \tag{2}$$

where $i = 1, 2, \ldots, N_r$, is the row position of the node; j = 1,2,…,N_c, is the column position of the node. And n_i is the number of program inputs. The output genes O should follow:

$$0 \leq O_k < n_i + L_n, \tag{3}$$

where $k = 1, 2, \ldots, n_o$ is the output gene's index, n_o is the number of output genes, and L_n is the user-determined upper bound of the number of nodes.

Review of work on operators in CGP

Many studies on crossover operator or mutation operator in CGP [19] have been conducted for improving the optimization performance.

Clegg et al. [6] proposed a real-valued representation for CGP and a new crossover operator, which recombines two genotype by arithmetic crossover with a random weighting factor. This new crossover was evaluated on the symbolic regression problems and has shown to be useful to improve convergence speed. Slanỳ et al. [27] studied the functional-level CGP including single and multipoint crossover operators. The experimental results showed that the mutation operator and single-point crossover operator can generate the smoothest landscapes for the image operator design problems. Kalkreuth [12] introduced a new crossover technique which recombines subgraphs of two selected graphs. Experimental results on symbolic regression, boolean functions, and image operator design problems showed that the new crossover operator can improve the performance of CGP.

Goldman et al. [10] introduced a variant of the standard point mutation technique called single active-gene mutation strategy (SAGMS). The mutation operator in this strategy mutates exactly one active gene for all the offspring, which has shown to be useful for improving performance. On the basis of this word, Kalkreuth [13] proposed the advanced phenotypic mutation to activate and deactivate randomly chosen function nodes. The experimental results indicated that this method could be beneficial for CGP. This advanced phenotypic mutations is evaluated in Sect. 4 of this paper to compare with the baseline CGP and the proposed CGP.

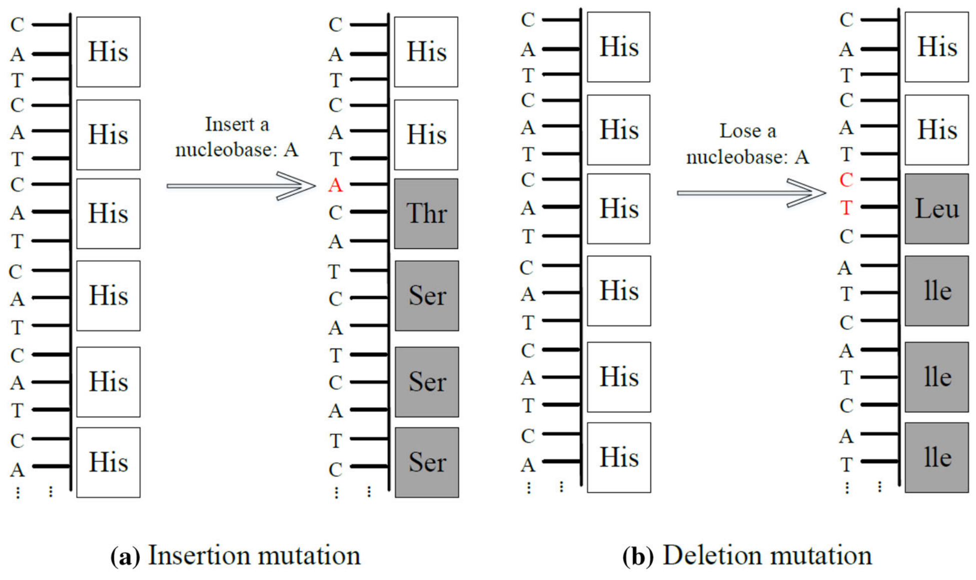

Fig. 1 An example of frameshift mutation in DNA

The proposed FMCGP

Frameshift mutation in biology in DNA

Gene mutations in biology can be generally divided into two categories, which are base substitution [9] and frameshift mutations [7]. Base substitution mutation is a kind of mutation that one base pair is replaced by another base pair in a DNA molecule. After base substitution mutation, only one amino acid corresponding to the base pair may be affected. The basic point mutation in CGP is single-point mutation, which belongs to base substitution mutation if mapping it to biology.

Compared to base substitution mutation, frameshift mutation causes a much larger change in DNA and amino acid. For frameshift mutation [7], DNA coding sequences can be misaligned when one or several base pairs are inserted or lost at a certain position in a DNA fragment. Figure 1 shows an example of the frameshift mutation in DNA. As shown in Fig. 1a, when insertion mutation happens and a base ‘A’ is inserted, the protein encoded from DNA is totally changed after the mutation point. As shown in Fig. 1b, when deletion mutation happens and a base ‘A’ is lost, the protein encoded from DNA is also totally changed after the mutation point. The frameshift mutation changes the reading frame of DNA, which makes the polypeptide chain totally different from the correct one. From the perspective of phenotype, frameshift mutation increases the diversity of gene expression, although it may harmful in biology.

Frameshift mutation in CGP

Inspired by the frameshift mutation in DNA, we introduce the frameshift mutation operator in CGP with the purpose to make the population more diverse and improve the ability of avoiding the premature convergence. The designed frameshift mutation can be caused by two operations, which are the insertion and deletion of nodes in the CGP network. These two operations make individuals in FMCGP have a variable-length genotype, which changes the size of CGP network adaptively according to the problem.

Frameshift mutation caused by insertion

As shown in the left part of Fig. 2, it is a CGP individual containing 7 network nodes and 1 input node. The number of rows is $Nr = 1$, the number of columns is $Nc = 7$, and the levels-back is $l = Nc = 7$, which means that any nodes can connect to its predecessor node without position restrictions. These 7 nodes are termed as Node 1–Node 7. Each node contains 1 function gene and 2 connection genes, and all genes follow the allelic constraints. The Node 6 is set as the output node. The genotype is as follows:

0-100 210 312 423 543 642 764-6.

As shown in the right part of Fig. 2, if the insertion mutation pointer points to Node 4, a new node named Node 8 is inserted before Node 4. The gene of the new node should also obey the allelic constraints. The genotype is changed as follows:

0-100 210 312 **823** 423 543 642 764-6.

After the insertion of the new node, the output gene is still ‘6’. However, the output node is changed implicitly from Node 6 to Node 5 since the frameshift mutation. Besides, Node 4 and Node 6 change to be inactive nodes and the Node 5 is activated. As all nodes after the new node move backward, the index of these nodes is changed. That is the reason why most genes in individuals are not altered, but the phenotype compiled has changed significantly.

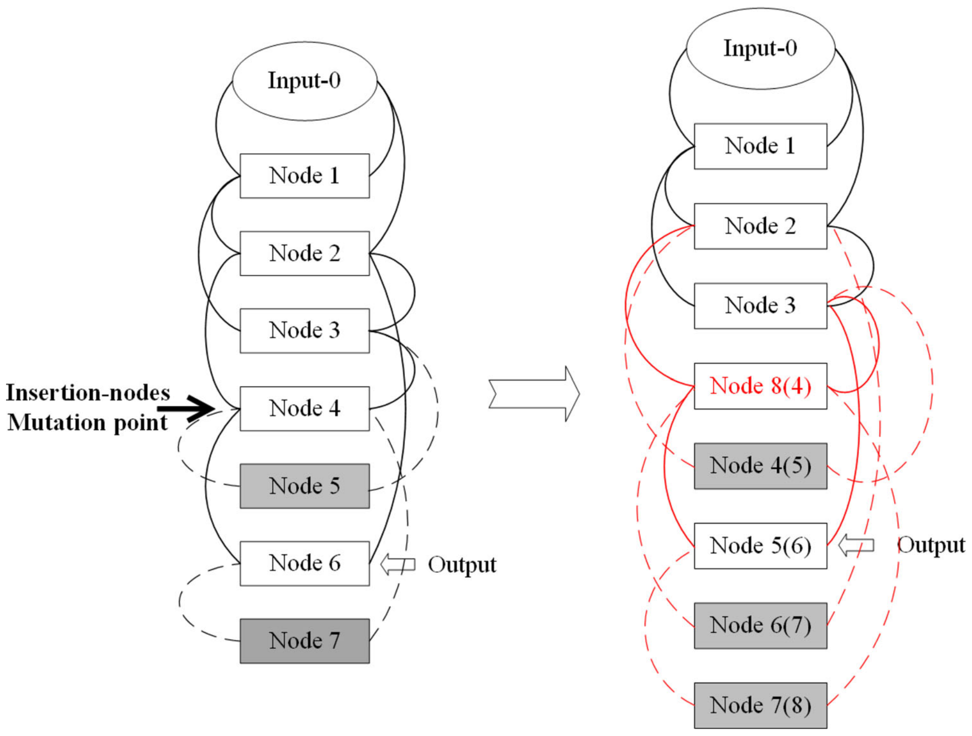

Fig. 2 Frameshift mutation caused by insertion in FMCGP. Grey nodes indicate inactive nodes, solid lines indicate active connections, and dashed lines indicate inactive connections. The red line indicates the connection that changed after the frameshift mutation. The numbers in parentheses refer to the index numbers after frameshift mutation

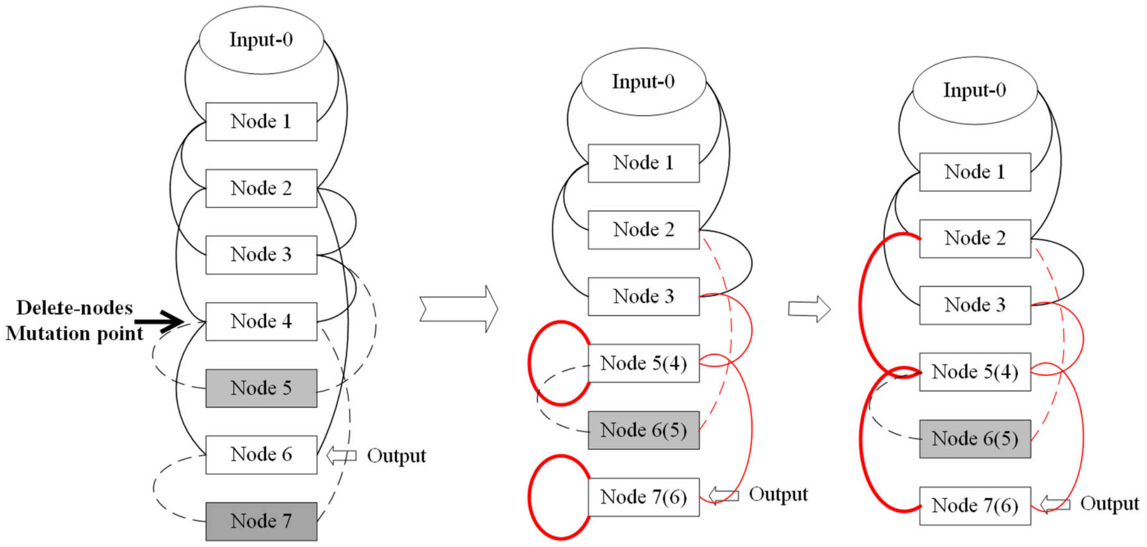

Fig. 3 Frameshift mutation caused by deletion in FMCGP. The meanings of grey nodes, solid lines, dashed lines, red lines, and the numbers in parentheses are the same as those in Fig. 2

Frameshift mutation caused by deletion

As shown in Fig. 3, the CGP network is with the same genotype as that in the previous subsection, that is:

0-100 210 312 423 543 642 764-6.

If the deletion mutation pointer points to Node 4, the Node 4 is deleted from the network. The genotype is changed as follows:

0-100 210 312 543 642 764-6.

Similarly, as all nodes after the deleted node moves forward, the index of these nodes is modified. As a result, the node that output gene refers to is changed and the active nodes of the individual are also changed. The phenotype of offspring is completely different from the parent individual after frameshift mutation, even there is no new nodes added into the network.

As can be seen from the first genotype after frameshift mutation in Fig. 3, some nodes are connected to themselves, e.g., the gene of Node 5 is '543' while the index number of Node 5 changes to '4' coincidentally. This kind of connection is not feasible, since an endless loop is compiled. Therefore, a re-assignment operation is performed when this happens. The infeasible connection is reassigned as the second mutated genotype, as shown in Fig. 3. And if the output gene is out of the range of genotype length, this re-assignment operation is also performed.

Algorithm 1 $(\mu + \lambda)$ FMCGP

```
1: for all $i$ such that $0 \le i < \mu$ do
2:     generate individual $i$ randomly
3: end for
4: fittest individual is selected as parent
5: while there is no solution or number of generations not exhausted do
6:     for all $i$ such that $0 \le i < \lambda$ do
7:         do basic mutation on parent
8:         offspring $i$ generated
9:         /* Frameshift mutation begins */
10:         if random $r_{fm}$ < probability of frameshift mutation occurs on individual($P_{fm}$) then
11:             if random $r_o$ < probability of Insertion($P_i$) then
12:                 randomly insert node in offspring $i$ to cause frameshift mutation
13:             else if random $r_o$ > (1- $P_i$) then
14:                 randomly delete node in offspring $i$ to cause frameshift mutation
15:             else
16:                 keep the individual $i$ unchanged
17:             end if
18:         end if
19:         /* End of frameshift mutation */
20:     end for
21:     generation of the fittest individual using the following
22:     if an offspring's fitness is equal or better than parents then
23:         choose the offspring
24:     else
25:         choose the parent
26:     end if
27: end while
```

From the above description, the phenotype makes a great difference, although only a few modifications are made to the genotype. The parent individuals may generate completely different offspring since the change of the compiling framework of genotype, which caused by different positions of frameshift mutation point, even the mutation point points to inactive nodes. From this point of view, frameshift mutation obviously further increases the population diversity based on the traditional point mutation.

FMCGP adopts $(\mu + \lambda)$ evolutionary strategy as standard CGP. Frameshift mutation is introduced in the process of generating offspring individuals. Algorithm 1 gives the pseudocode of FMCGP.

Experimental results

Parameter settings

Symbolic regression problems [16,28] and Even-N-parity problems [17] are selected to evaluated the performance of the proposed FMCGP. Each experiment is executed 100 independent runs for statistically meaningful results. Average values, standard deviations, and the average cost times are recorded. The population size of all experiments is 10, which means that the evolutionary strategy is $(1 + 10)$.

Individuals are presented by a directed graph with one row and a number of columns, where $N_r = 1$ and $N_c = L_n$. And the levels-back $l = N_c$, the nodes of the full graph can be freely connected forward. According to the empirical value [22], the point mutation rate is set to 1%, and function gene mutation, connection gene mutation, and output node mutation are fairly selected with a probability of $1 : 1 : 1$ for point mutation. The standard CGP with point mutation is referred to as the *baseline CGP* in the remainder of this paper.

For a thorough comparison, the Mann–Whitney U test has been carried out between the proposed FMCGP and baseline CGP. To classify the significance, the average values of FMCGP are denoted $a^{\dagger}$ if the p value is less than significance level 0.05 and $a^{\ddagger}$ if the p value is less than significance level 0.01 compared to the baseline CGP.

Control group algorithms

We use the advanced phenotypic mutations proposed by Kalkreuth [13] (referred to as TAPMCGP) as the additional control group algorithms for the experiments.

TAPMCGP is inspired by biological evolution which mutates DNA sequences by inserting and deleting nucleotides. Kalkreuth et al. [13] applied an insertion rate and a deletion rate for each offspring which is selected for mutation. If a genome is selected for the insertion mutation, one inactive function node becomes active. On the contrary, when deletion is performed, one active node becomes inactive. The minimum number of active function nodes for deletion operation has to be defined, this parameter in the experiments is set to 4 according to [13].

Symbolic regression problems

In the first group of symbolic regression experiments, Koza-3 is selected to evaluate the improvement of FMCGP with different frameshift mutation rates. In the second group, a series of objective functions are used for the comparison among baseline CGP, TAPMCGP, and proposed FMCGP. We measure the number of generations until the algorithms reach termination criteria (*generations-to-success*) on Koza-2, Koza-3, and Nguyen-4, and measure the best fitness value

Table 1 Symbolic Regression Benchmark Candidates (U[a, b, c] means **c** uniform random samples drawn from a to b for the variable [20], E[a, b, c] means samples are a grid of points evenly spaced (for this variable) with an interval of c, from a to b inclusive.)

Name	Vars	Objective function	Training set	Testing set	Function set
Koza-2 [16]	1	$x^5 - 2x^3 + x$	U[-1,1,20]	None	$\{+, -, *, /\}$
Koza-3 [16]	1	$x^6 - 2x^4 + x^2$	U[-1,1,20]	None	$\{+, -, *, /\}$
Nguyen-4 [28]	1	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	U[-1,1,20]	None	$\{+, -, *, /\}$
Keijzer-12 [14]	2	$x^4 + x^3 + \frac{y^2}{2} - y$	U[-3,3,20]	E[-3, 3, 0.01]	$\{+, -, \frac{1}{n}, -n, \sqrt{n}\}$
Keijzer-15 [14]	2	$\frac{x^3}{5} + \frac{y^3}{2} - y - x$	U[-3,3,20]	E[-3, 3, 0.01]	$\{+, -, \frac{1}{n}, -n, \sqrt{n}\}$

Table 2 Results for Koza-3 regression problem (the evolutions which has converged but the fitness of best individual is over 0.01 are record as outliers. The outliers have been removed from the average calculation)

Evolutionary	Frameshift	Generations		Cost time(s)		Number
method	rate	Mean	Std	Mean	Std	of outliers
Baseline CGP	N/A	1440.34	2328.14	118.51	135.72	2
FMCGP	0.1	1212.06	1960.87	188.81	374.78	0
	0.3	702.52†	811.74	133.62	295.22	0
	0.5	593.47‡	629.68	87.75†	128.14	0
	0.7	**516.05‡**	**533.20**	**73.53‡**	**140.99**	**0**
	0.9	832.63†	1139.68	92.00‡	281.61	0

after a predefined number of generations (*best-fitness-of-run*) on Keijzer-12, and Keijzer-15 [14,26]. The function name, objective function, training set, testing set, and function set used in this paper of benchmark symbolic regression problems are listed in Table 1 [14,16,20,28].

The fitness of each individual is represented by the cost function value, which is defined as the sum of the absolute difference between the predicted value and the training value. The termination criteria is defined as the fitness value less than or equal to 0.01. The probability of insertion or deletion at the frameshift mutation point is 0.5. The length of genotypes of initial individuals is set to 30 in the symbolic regression experiments. The results of mean generations and mean cost time (in seconds) with standard deviations after 100 runs on Koza-3 are given in Table 2 along with the Mann–Whitney U test results.

Results in Table 2 indicate that with the increasing of frameshift rate, the algorithms' performance increases except when the rate is 0.9. The FMCGP with the frameshift mutation rate 0.7 shows the best performance, while FMCGP with the mutation rate 0.1 performs the worst. It implies that if the frameshift rate is too high, the FMCGP may cause poor results. The number of outliers of baseline CGP is 2, while this number decreases to 0 after the frameshift mutation is introduced. From the perspective of generations and cost time, FMCGP with the frameshift mutation rate 0.7 performs significantly better than the baseline CGP and the FMCGP with the other frameshift mutation rates on Koza-3.

Table 3 gives the results of mean generations and cost time (in seconds), and standard deviations after 100 independent runs on Koza-2, Koza-3, and Nguyen-4. The Mann–Whitney U test results are also given. The results of best fitness values after 10000 generations on Keijzer-12 and Keijzer-15 are shown in Table 4. The frameshift mutation rate in this group of experiments is set to 0.7 according to the above experimental results.

The baseline CGP with single-point mutations and TAPMCGP may cause several outliers, while the number of outliers generated by FMCGP is 0. The experimental results in Table 3 show that FMCGP has a faster convergence speed than baseline CGP on different symbolic regression problems. TAPMCGP requires fewer average evaluation times than FMCGP on Koza2 and Nguyue-4 issues, but it runs much longer than FMCGP. As the results in Table 4, the mean best fitness of proposed FMCGP is better than baseline CGP after 10000 generations of evolution. On Keijzer-15, the TAPMCGP results in a better fitness value, but the cost time of TAPMCGP is twice as long as FMCGP.

Even-parity problems

The digital circuit establishment Even-parity problem [17] is another well-known problem to test evolutionary algorithms. Several discrete boolean problems with different number of inputs are used to investigate the performance of baseline CGP, TAPMCGP, and the proposed FMCGP.

The parameter configurations are set according to the corresponding references [17]. The function set of this group of experiments is $\{AND, OR, XOR, NOT\}$. The fitness is defined as the percentage of the candidate solutions which

Table 3 Results for Koza2, Koza-3, and Nguyen-4 regression problems (the evolutions which has converged but the fitness of best individual is over 0.01 are record as outliers. The outliers have been removed from average calculation)

Experiments	Evolutionary method	Generations Mean	Generations Std	Cost time(s) Mean	Cost time(s) Std	Number of outliers
Koza-2	Baseline CGP	2603.06	4695.94	204.14	199.21	5
	TAPMCGP	**650.05**	**1383.76**	281.73	449.29	2
	FMCGP	957.95‡	1503.50	**99.01**‡	**134.98**	**0**
Koza-3	Baseline CGP	1440.34	2328.14	118.55	**135.72**	2
	TAPMCGP	621.14	1120.59	739.50	1174.40	0
	FMCGP	**516.05**‡	**533.20**	**73.53**‡	140.99	**0**
Nguyen-4	Baseline CGP	4048.45	7756.95	266.47	310.68	9
	TAPMCGP	**1720.10**	3271.239	1327.59	1393.56	1
	FMCGP	1828.98†	**2021.76**	**166.15**‡	**176.91**	**0**

Table 4 Results for Keijzer-12 and Keijzer-15 regression problems

Experiments	Evolutionary method	Best fitness Mean	Best fitness Std	Cost time Mean	Cost time Std
Keijzer-12	Baseline CGP	5.01E06	**1.22E05**	142.51	20.79
	TAPMCGP	5.21E06	7.13E05	301.35	70.16
	FMCGP	**4.82E06**	8.07E05	**134.23**	**14.56**
Keijzer-15	Baseline CGP	7.78E05	2.40E05	**73.34**	**7.08**
	TAPMCGP	**7.14E05**	5.37E05	186.89	17.50
	FMCGP	7.18E05	**1.94E05**	95.78	14.47

Table 5 Results for Even-parity problems with different number of inputs

Experiments	Number of nodes	Evolutionary method	Generations Mean	Generations Std	Cost Time (s) Mean	Cost Time (s) Std
Even-4-Parity	10	Baseline CGP	189.34	258.87	9.05	15.98
		TAPMCGP	**81.70**	**55.71**	2.11	2.92
		FMCGP	137.76	129.98	**1.82**‡	**1.09**
Even-5-Parity	10	Baseline CGP	884.06	953.34	157.66	728.94
		TAPMCGP	**264.61**	**181.29**	**5.72**	**4.55**
		FMCGP	455.98‡	326.67	7.03‡	11.93
Even-6-Parity	10	Baseline CGP	4808.52	3492.21	2553.45	5295.11
		TAPMCGP	**1025.53**	903.61	**13.24**	**10.80**
		FMCGP	1165.46‡	**571.85**	22.32‡	25.44
Even-7-Parity	20	Baseline CGP	3594.20	3001.45	126.69	138.67
		TAPMCGP	**1273.89**	**938.37**	51.15	57.89
		FMCGP	2355.02†	1309.64	**45.62**‡	**26.12**
Even-8-Parity	20	Baseline CGP	20317.93	15314.49	436.70	335.16
		TAPMCGP	**3806.38**	2747.64	136.91	**59.88**
		FMCGP	4898.40‡	**2523.88**	**112.14**‡	63.10

generate the wrong Even-parity function value. The termination criteria are defined as the fitness equal to 0. The probability of insertion or deletion at the mutation point is 0.5. The length of genotypes of the initial individuals is set to 10 in the Even-4/5/6-parity experiments and 20 in the Even-7/8-parity experiments. The results of mean generations and mean cost time (in seconds) with standard deviations after 100 runs are displayed in Table 5 together with Mann–Whitney *U* test results. The efficiency gain (i.e., the ratio of FMCGP value to the baseline CGP, and TAPMCGP) for both generation and cost time values of FMCGP are displayed in Table 6.

As shown in Table 5, the baseline CGP needs much more generations and cost time to finish evolution than the pro-

Table 6 Efficiency gains for Even-parity problems when using FMCGP compared with the baseline CGP and TAPMCGP

Experiments	Evaluation times efficiency gains		Cost time efficiency gains	
	Baseline CGP	TAPMCGP	baseline CGP	TAPMCGP
Even-4-Parity	**1.37**	0.59	**4.97**	**1.16**
Even-5-Parity	**1.94**	0.58	**22.43**	0.74
Even-6-Parity	**4.13**	0.88	**114.40**	0.59
Even-7-Parity	**1.52**	0.54	**2.78**	**1.12**
Even-8-Parity	**4.15**	0.78	**3.89**	**1.22**

posed FMCGP. The results in Table 6 obviously show that the efficiency gains increase with the number of inputs increases. This trend indicates that FMCGP has a better performance on the evolution speed in dealing with more complex problems.

Similar to the results of the SR experiments, TAPMCGP requires less generations to reach the goal fitness, but it requires more cost time than FMCGP. The number of nodes in Even-6-Parity, Even-8-Parity, and SR experiments are 10, 20, 30, and the ratio of cost time of TAPMCGP and FMCGP is increasing. The experimental results show that TAPMCGP needs more time to evolve when the number of nodes in individuals is large. The reason lies that TAPMCGP has to check the activity of nodes before mutation operation, while FMCGP does not need. As described in Sect. 3.2, the parent individuals in FMCGP may generate different offspring since the change of the compiling framework of genotype, regardless of whether the mutation operation is applied on the active node.

The ratio of generation of FMCGP and TAPMCGP in Table 6 raises when the input number increases. These ratio values imply that the advantage of TAPMCGP in the number of generation becomes smaller when the input number increases, which may be caused by the fixed-length genotype like the baseline CGP. The evolution speed of the baseline CGP in the Even-7-parity using 20 nodes is faster than the Even-6-parity using 10 nodes. The genotype length, represented by the number of nodes in individuals, is significantly influence the speed of convergence, while the value in the baseline CGP is a fixed value. During the evolution of FMCGP, the length of genotypes is adjusted to a more suitable value, which helps to accelerate the evolution.

Discussion

Diversity

Several diversity measurements have been applied in genetic programming [3]. Ekárt and Németh [8] proposed an adapted version of the approach, which calculates diversity based on edit distance between individuals. Following this metric, two trees are brought to the same tree structure by adding "NULL" nodes to each tree. And then the difference between two trees is defined as follows:

$$dist(T1, T2) = \begin{cases} d(p, q) & if\ neither\ T1\ nor\ T2\ have\ any\ children \\ d(p, q) + \frac{1}{K} * \sum_{l=1}^{m} dist(s_l, t_l) & otherwise, \end{cases} \quad (4)$$

where $T1$ is the tree with root p and subtrees $s_1, s_2, \ldots, s_m$, and $T2$ is the tree with root q and subtrees $t_1, t_2, \ldots, t_m$, and K is a constant, signifying that a difference at any depth r in the compared trees is K times more important than a difference at depth $r + 1$. The value of K is set to 2 according to the reference [8].

To apply this metric to the analysis of diversity in CGP and FMCGP, which individuals represented in two-dimensional graphs, we must first convert the CGP and FMCGP individuals into a tree structure. Taking the individuals in Figs. 2 and 3 as examples, the tree structures of individuals before and after frameshift mutation operation are displayed in Figs. 4, 5, and 6.

As shown in Figs. 5 and 6, the frameshift mutation only applies on the Node 4, but the entire tree structure has been changed significantly. While in the point mutation operation, the function gene mutation can only change Node 4 in the tree and the connection function gene mutation can only change the subtree with Node 4 as the root node.

The distance values between different nodes in the tree are also set as [8]. The distance between "NULL" nodes and all other nodes is 5, the distance between leaf node and none-leaf node is 3, and the distance between different none-leaf nodes is 2. The average edit distance between the individual and the individual with the best fitness in the run so far of each generation is used as the diversity value of this generation. Figs. 7 and 8 show the diversity values of baseline CGP and the proposed FMCGP during the 10000 generation in the two sets of experiments.

Figures 7 and 8 indicate that the average population diversity of FMCGP remains stable at a higher level compared with baseline CGP, which is consistent with the significant

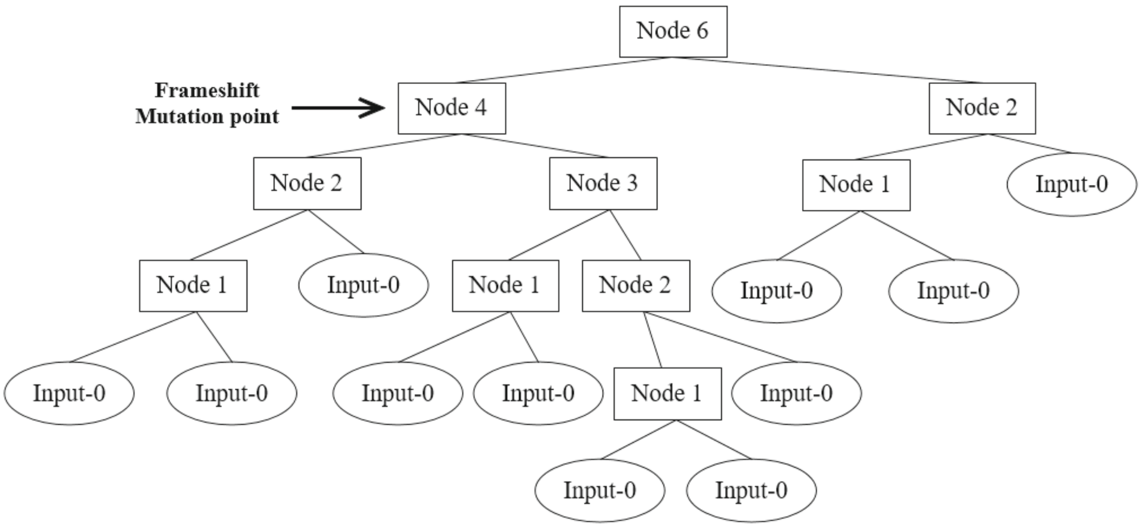

Fig. 4 The tree structure of the individuals before frameshift mutation

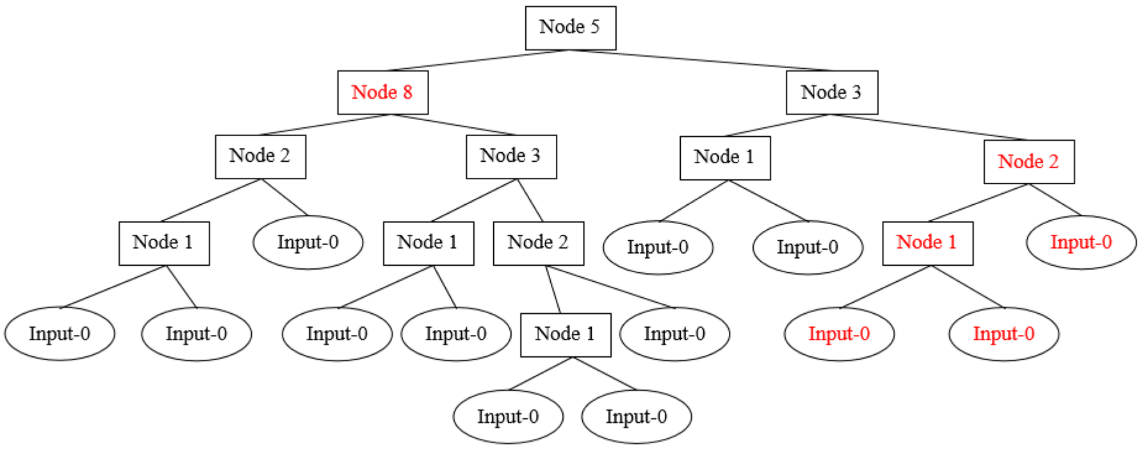

Fig. 5 The tree structure of the individuals after the frameshift mutation caused by an insertion. The red node indicates the change after frameshift mutation

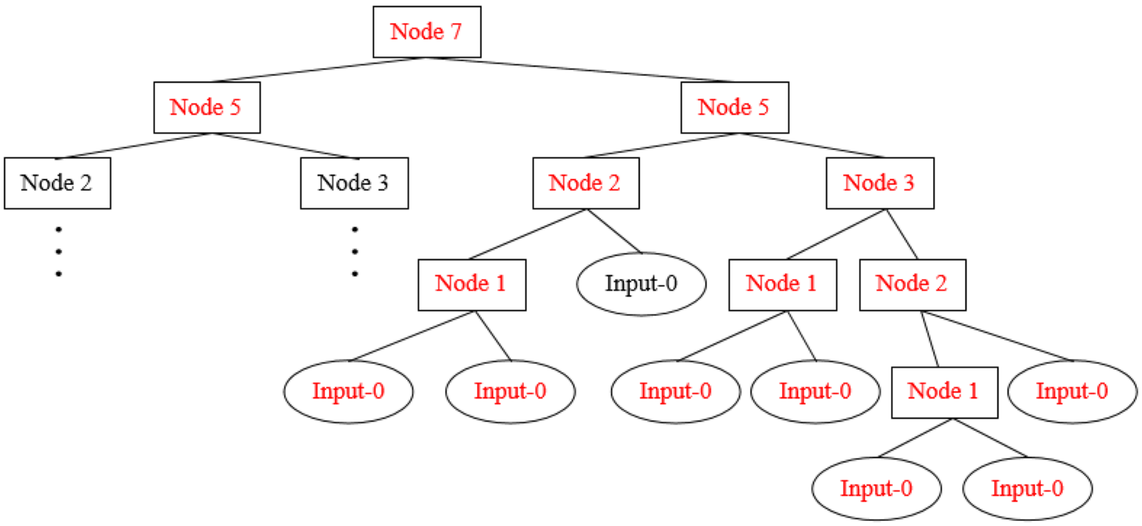

Fig. 6 The tree structure of the individuals after the frameshift mutation caused by an insertion. The red node indicates the change after frameshift mutation

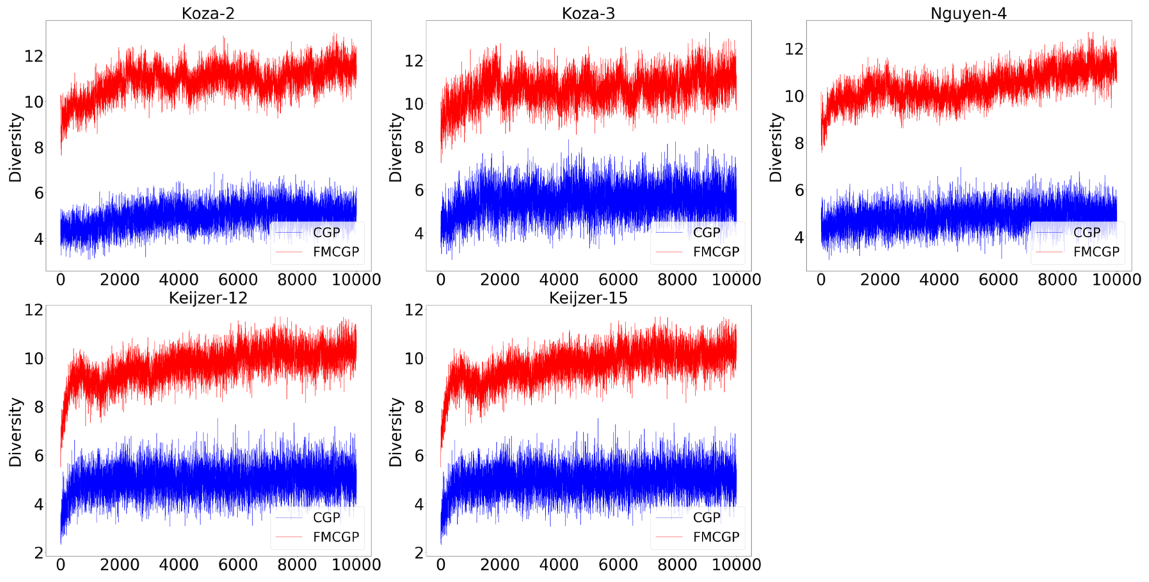

Fig. 7 The diversity of baseline CGP and the proposed FMCGP in symbolic regression experiments

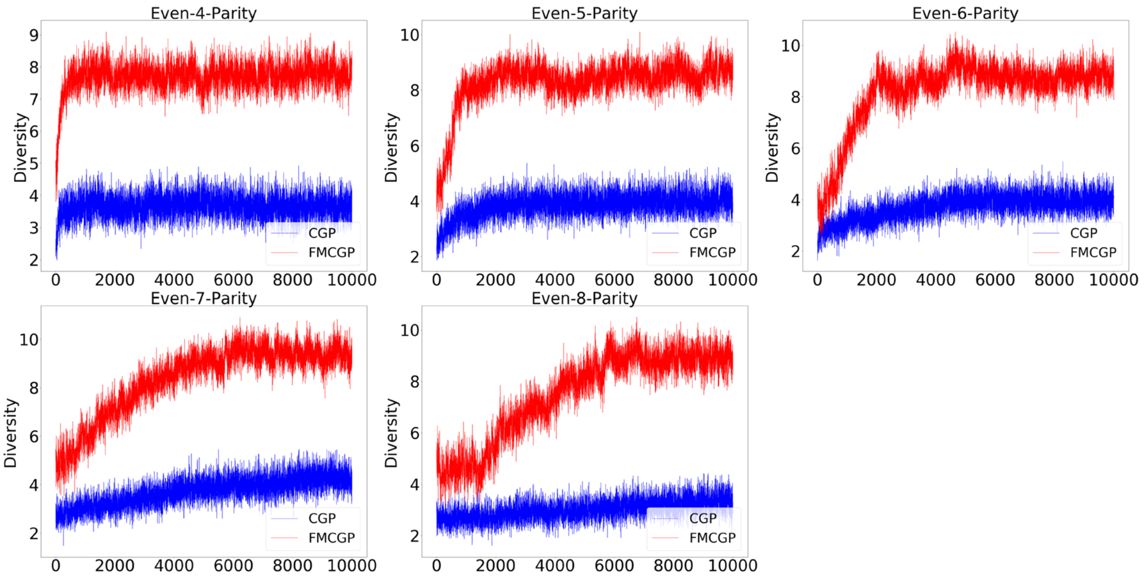

Fig. 8 The diversity of baseline CGP and the proposed FMCGP in Even-N-Parity experiments

structural changes caused by frameshift mutations on individuals.

The population diversity plays an important role in exploring the search space in the evolutionary algorithms. In the high-strength evolutionary strategy, e.g., $1 + \lambda$ used in the CGP, the evolution results are sensitive to initialization, while increasing the population diversity can help to reduce the local optimal problems.

From the experimental results in Sect. 4, it can be seen that the outliers of baseline CGP may be caused by unreasonable initial individuals, and then, the traditional point mutations cannot make the population reach the target fitness within a reasonable number of generations. As the decreasing of the numbers of outliers in the FMCGP, the increased diversity caused by frameshift mutations may help individuals to jump out of the local optimum and find the better results faster. The experimental results of FMCGP with frameshift rate 0.9 also imply that higher frameshift mutation may destroy outstanding individuals and then lead to the poor performance of the algorithm. An appropriate frameshift mutation probability is therefore important for FMCCP.

Bloat problem

The bloat problem means that the chromosomes become larger and larger, while the fitness does not increase any more

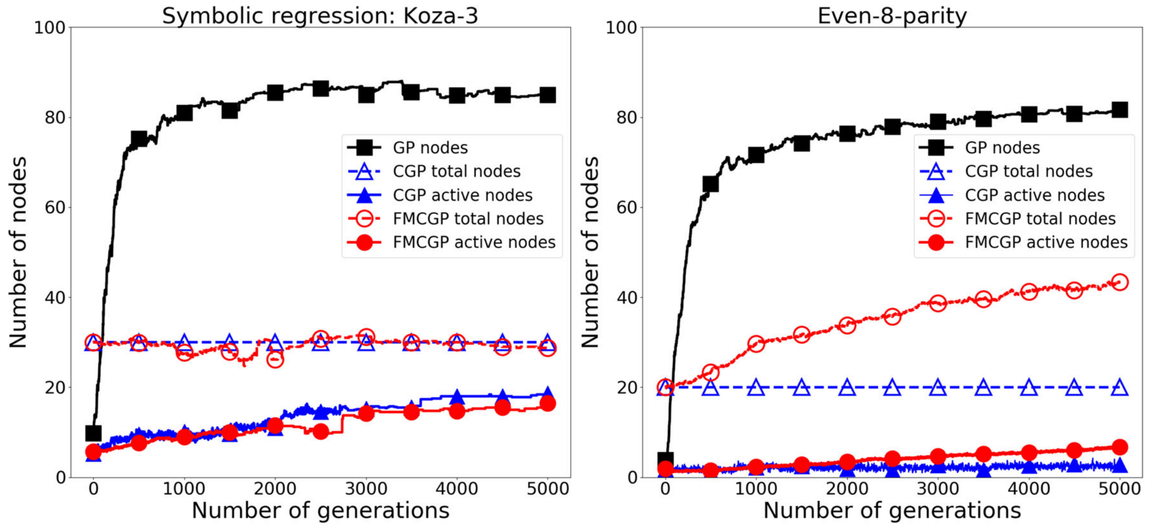

Fig. 9 The variation of program size of symbolic regression problem Koza-3 and boolean problem even-8-parity by different evolutionary method: tree-based GP, baseline CGP, and proposed FMCGP

[22], which wastes many computing resource on the redundant sub-expressions. In [21,23], Miller et al. studied the bloat problem in CGP and the results shows that CGP does not suffer from genotypic growth and phenotypic growth. This non-bloat advantage of CGP may be due to the fixed length of genotype. As FMCGP adopts variable-length genotype, it is worth further analysis whether FMCGP loses the benefit of non-bloat.

We investigate the changes of gene length of tree-based GP, standard CGP and proposed FMCGP in the Koza-3 and Even-8-parity problems. The population size of the tree-based GP in the experiment is also set to 10 and the maximum tree size is set to 100. The variations of program size with generation for 2 problems are plotted in Fig. 9. In the tree-based GP, there is a fast bloat in a few generations where the length of individuals grows rapidly. In the baseline CGP and FMCGP, the total number of nodes represents the length of genotype and the number of active nodes represents the length of phenotype. Since the fixed-length genotype in baseline CGP, the genotype length keeps unchanged in all generations and the phenotype length has a slow bloat which is much less than the tree-based GP [21].

The genotype length of the individuals in FMCGP has different trends in different problems. In the Koza-3 problem, there is a slight change in genotype size in the first 2000 generations, and then, the change becomes flat. The genotype length is consistent with the fixed value in the baseline CGP. The growth of phenotype size is slower than standard CGP. In the Even-8-Parity problem, the insertion frameshift mutation is obviously used much more than the deletion frameshift mutation. The genotype size is increased rapidly without decreasing trend. The phenotype size also increases faster than the baseline CGP, but the slow bloat in FMCGP is still much less than the near quadratic bloat in tree-based GP.

Therefore, we consider that the proposed FMCGP has not cost much more computing resources on redundant expression than baseline CGP. In other words, FMCGP does not lose the non-bloat characteristic of CGP and the growth of genotype and phenotype is within a reasonable range compared with standard tree-based GP.

Conclusion

This paper proposes FMCGP that introduces the frameshift mutation in CGP, which is inspired by DNA mutation in biology. The frameshift mutation makes the individuals have variable-length genotypes and increases the diversity of the population. Through two groups of experiments, FMCGP is shown to be able to outperform the standard CGP and the state-of-the-art TAPMCGP without bloat problem.

In the future, we will evaluate FMCGP on more functions of symbolic regression and make the improvement of FMCGP on different functions. And the frameshift mutation probability is now determined by the experiments, but it may prefer to be determined adaptively.

Acknowledgements This work was supported in part by the National Key R&D Program of China under Grant 2017YFC1601800 and 2017YFC1601000, in part by the National Natural Science foundation of China, under Grant 62073155, 61673194, and 61672263, in part by the Key Research and Development Program of Jiangsu Province, China, under Grant BE2017630, and in part by "Blue Project" in Jiangsu Universities.

References

1. Ahvanooey MT, Li Q, Wu M, Wang S (2019) A survey of genetic programming and its applications. TIIS 13(4):1765–1794
2. Banzhaf W, Lasarczyk C (2005) Genetic programming of an algorithmic chemistry. In: Genetic programming theory and practice II, pp. 175–190. Springer
3. Burke EK, Gustafson S, Kendall G (2004) Diversity in genetic programming: an analysis of measures and correlation with fitness. IEEE Trans Evol Comput 8(1):47–62
4. Can B, Heavey C (2012) A comparison of genetic programming and artificial neural networks in metamodeling of discrete-event simulation models. Comput Oper Res 39(2):424–436
5. Cattani PT, Johnson CG (2010) Me-cgp: Multi expression cartesian genetic programming. In: IEEE Congress on Evolutionary Computation. pp. 1–6. IEEE
6. Clegg J, Walker JA, Miller JF (2007) A new crossover technique for cartesian genetic programming. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. pp. 1580–1587. ACM
7. Crick F, Barnett L, Brenner S, Watts-Tobin RJ (1961) General nature of the genetic code for proteins
8. Ekart A, Nemeth SZ (2000) A metric for genetic programs and fitness sharing pp. 259–270
9. Freese E (1959) The specific mutagenic effect of base analogues on phage t4. J Mol Biol 1(2):87–105
10. Goldman BW, Punch WF (2013) Length bias and search limitations in cartesian genetic programming pp. 933–940
11. Holland JH et al (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press
12. Kalkreuth R, Rudolph G, Droschinsky A (2017) A new subgraph crossover for cartesian genetic programming pp. 294–310
13. Kalkreuth RT (2018) Towards advanced phenotypic mutations in cartesian genetic programming. Neural and Evolutionary Computing
14. Keijzer M (2003) Improving symbolic regression with interval arithmetic and linear scaling pp. 70–82
15. Khan MW, Alam M (2012) A survey of application: genomics and genetic programming, a new frontier. Genomics 100(2):65–71
16. Koza JR (1994) Genetic programming ii: automatic discovery of reusable subprograms. Cambridge, MA, USA 13(8):32
17. Koza JR, Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection, vol. 1. MIT press
18. Langdon, W.B.: Genetic programming–computers using "natural selection" to generate programs. In: Genetic programming and data structures, pp. 9–42. Springer (1998)
19. Manazir A, Raza K (2019) Recent developments in cartesian genetic programming and its variants. ACM Comput Surv (CSUR) 51(6):122
20. McDermott J, White DR, Luke S, Manzoni L, Castelli M, Vanneschi L, Jaskowski W, Krawiec K, Harper R, De Jong K et al (2012) Genetic programming needs better benchmarks. In: Proceedings of the 14th annual conference on genetic and evolutionary computation. pp. 791–798
21. Miller J (2001) What bloat? cartesian genetic programming on boolean problems. In: 2001 Genetic and Evolutionary Computation Conference Late Breaking Papers. pp. 295–302. San Francisco, California, USA
22. Miller JF (2011) Cartesian genetic programming. In: Cartesian Genetic Programming, pp. 17–34. Springer
23. Miller JF, Smith SL (2006) Redundancy and computational efficiency in cartesian genetic programming. IEEE Trans Evol Comput 10(2):167–174
24. Miller JF, Thomson P, Cartesian genetic programming
25. Miller JF, Thomson P, Fogarty T (1997) Designing electronic circuits using evolutionary algorithms. arithmetic circuits: a case study
26. Nicolau M, Agapitos A, Oneill M, Brabazon A (2015) Guidelines for defining benchmark problems in genetic programming pp. 1152–1159
27. Slanỳ K, Sekanina L (2007) Fitness landscape analysis and image filter evolution using functional-level cgp. In: European conference on genetic programming. pp. 311–320. Springer
28. Uy NQ, Hoai NX, O'Neill M, McKay RI, Galván-López E (2011) Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genetic Programming and Evolvable Machines **12**(2), 91–119
29. Vassilev VK, Miller JF (2000) The advantages of landscape neutrality in digital circuit evolution. In: International conference on evolvable systems. pp. 252–263. Springer
30. Zhao H (2007) A multi-objective genetic programming approach to developing pareto optimal decision trees. Decision Support Syst 43(3):809–826