2018 Joint 10th International Conference on Soft Computing and Intelligent Systems and 19th International Symposium on Advanced Intelligent Systems

A Study on Multimodal Genetic Programming Introducing Program Simplification

Kei Murano
Ritsumeikan University
Shiga, Japan
is0284fv@ed.ritsumei.ac.jp

Shubu Yoshida
Ritsumeikan University
Shiga, Japan
is0242rv@ed.ritsumei.ac.jp

Tomohiro Harada
Ritsumeikan University
Shiga, Japan
harada@ci.ritsumei.ac.jp

Ruck Thawonmas
Ritsumeikan University
Shiga, Japan
ruck@is.ritsumei.ac.jp

***Abstract*—In this research, we introduce a program simplification method into Multimodal Genetic Programming (MMGP) and investigate its effectiveness on multimodal program optimization benchmark. In recent years, multimodal optimization that simultaneously acquires a global and multiple local optimum solutions is studied in evolutionary algorithms (EAs). MMGP we proposed is an extension of genetic programming to multimodal optimization that simultaneously acquires a global and local optimum programs in a single run. However, since MMGP divides the solution set to several clusters depending on a tree similarity measurement, some programs cannot be assigned to appropriate cluster when a redundant subtree is generated in the optimization process. To overcome this problem, this research introduces a simplification method of a program into MMGP to remove redundant subtrees and appropriately calculate similarity of programs. The experiment that compares MMGP with and without the simplification method is conducted. The experimental result reveals that the simplification does not significantly improve the search ability of MMGP because the simplification does not much affect the optimization process of MMGP on the benchmark problem used in this research.**

***Index Terms*—genetic programing, simplification, multimodal optimization**

I. Introduction

In recent years, multimodal optimization that simultaneously acquires a global and multiple local optimum solutions is studied in the evolutionary algorithm (EA) domain. The local optimum solution has a lower (worse) fitness than the global one, but it is a solution that is optimal in the local search region. Our previous research proposed Multimodal Genetic Programming (MMGP) [2], which introduces the concept of multimodal optimization into genetic programming (GP) [1].

In MMGP, programs or mathematical expressions, which are represented as a tree structure, are optimized as same as GP and MMGP aims to simultaneously acquire a global and local optimum programs. To accomplish multimodal search in GP, MMGP divides population (set of solutions) into a several clusters based on the similarity of programs calculated according to a tree similarity, and optimization is performed for each cluster.

However, in MMGP, since a tree similarity is used for clustering the population, the similarity of programs is not properly evaluated if a redundant subtree (e.g., $x - x$) is generated in the process of optimization. In order to solve this problem, this paper applies a simplification mechanism [3] to remove redundant subtrees of programs to MMGP. In the simplification in GP, the tree structure of the program is scanned from the terminal node, and if a redundant subtree is found, the simplification is applied to the root node. Finally the simplification method outputs a simplified program.

In order to analyze the influence of the simplification in MMGP, this paper compares MMGPs with and without the simplification on the benchmark problem of multimodal program optimization proposed in the previous research. In this paper, two methods are examined as MMGP with the simplification. The first method is to apply the simplification to all programs generated by genetic operations and evolve simplified programs. The second one is, although the simplification is applied when calculating tree similarity, optimization is performed by *not* simplified programs.

The remaining of this paper is as follows. Section II describes overview of GP and multimodal optimization. Section III explains multimodal genetic programming, MMGP, proposed in the previous paper. Section IV proposes MMGP with the simplification method, and Section V conducts the experiment that compares the conventional and proposed MMGPs and discusses its result. Finally, Section VI describes the summary of this paper and future works.

II. Related research

A. Genetic Programing (GP)

GP [1] is an extension method of Genetic Algorithm (GA) [5] invented by John Koza. GP handles mathematical equations and programs as optimization targets. In GP, a solution is expressed by using a tree structure. This makes it possible to express a solution with structure such as mathematical formulas and program codes which is hardly expressed with a linear chromosome used in GA. Figure 1 shows an example of a tree structure expressing a mathematical formula ($2\times(x-x)$). This tree structure consists of operators ("$\times$" and "$-$" in the figure), variables ("x" in the figure) and constant value ("2" in the figure).

B. Multimodal optimization problem

Multimodal optimization problems do not only aim to obtain a global optimal solution but also obtain all local optimal solutions simultaneously. A local optimal solution is defined as an optimal solution in a local search region. Figure 2

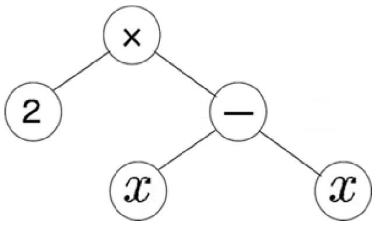


Figure 1: An example of a tree structure used in GP

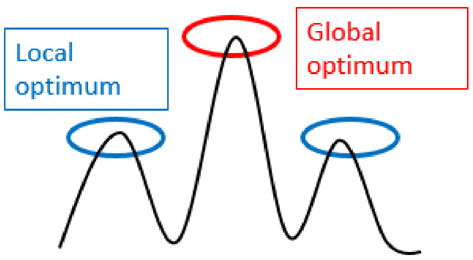


Figure 2: An example of multimodal optimization problem

shows an example of a multimodal optimization problem. In Fig. 2, the vertical axis represents fitness, and the horizontal axis represents search space. Higher fitness indicates better solution. In this figure, since the solution indicated by the red circle has the highest fitness in the whole search space, this is the global optimum solution. While the solutions indicated by the blue circle have inferior fitness to the global optimum one but they have the best fitness in the local search region. For this, these solutions are defined as local optimal solutions.

III. Multimodal Genetic Programming (MMGP)

Multimodal GP (MMGP) our previous research [2] proposed is an extension of GP to multimodal optimization. MMGP can acquire a global and local optimal programs simultaneously by dividing a population into several clusters by clustering using the similarity of a tree structure as a program expression. Optimization of MMGP is performed for each cluster, thereby each different structure of program is searched separately while maintaining multimodality.

In the following subsection, a benchmark problem of multimodal program optimization proposed in [2] is expressed, and then, the algorithm of MMGP is described.

A. Multimodal program optimization problem

In this paper, we use a benchmark problem of the multimodal program optimization problem proposed in the previous study [2]. This benchmark problem has four variables x, y, z, w as inputs and is defined as a symbolic regression problem based on input-output values given by the following equation:

$$\begin{aligned} f(x,y,z,w) &= x^2+y^2, & (1)\\ z &= (x+y+\alpha)^2+\delta, & (2)\\ w &= \alpha^2/2+xy+x\alpha+y\alpha+\delta, & (3) \end{aligned}$$

where α is a constant and δ is the error value given to the variables z and w. If δ is 0, this equation is reconstructed as follows:

$$\begin{aligned} f(x,y,z,w) &= x^2+y^2 & (4)\\ &= z-2w, & (5) \end{aligned}$$

From Eq. (1), given input-output values can be expressed by Eq. (5) using variables z and w instead of variables x and y. If δ is not equal to 0 and an error is added to z and w, the output value calculated by Eq. (5) is considered as a local optimal solution because output values differ from the correct ones given by the global optimum one expressed by Eq. (1) due to error values. That is, when δ = 0, $z-2w$ and x^2+y^2 have the same calculation result. However, when $\delta \neq 0$, the calculation result of $z-2w$ is a little different from the one of x^2+y^2, so that its accuracy decreases and it can be regarded as a local optimal solution.

B. Algorithm of MMGP

Algorithm 1 shows the pseudo code of MMGP. In Algorithm 1, P_g denotes the population of gth generation. C_g denotes the clustered population of the gth generation and C_g^i denotes the ith cluster of C_g. `rand`(a,b) generates random value from a to b. `tournament`(C) selects one solution from C by using tournament selection, while `negative_tournament`(C) selects one solution from C by using *negative* tournament selection where one worst solution is selected from two randomly selected solutions from C. `crossover`(p_1,p_2) and `mutation`(p) conduct crossover and mutation operators, simultaneously. `clusterint`(P) divides a population P into several clusters.

In MMGP, new individuals are generated from lines 5 to 18. In line 11, crossover is performed on the selected two individuals. In line 14, mutation is performed on the selected individual. In the crossover, parent p_1 is selected from a certain cluster C_g and another parent p_2 is selected from randomly selected cluster C_r. According to this selection, individuals can be locally optimized within each cluster while evenly referring to all the clusters. If the partner solution of crossover is selected from the same cluster, it is hard to search a wide range of search space and there is a high possibility to loss opportunity to find local optimal solution. In the mutation, a parent is selected from the cluster C_i. Next, in line 20, clustering is performed based on the similarity calculation by Rui et al. [4]. The hierarchical clustering is used in MMGP and the population is divided until the minimum similarity of clusters becomes less than a certain threshold d (see detail in [2]). Using the current population P_g and the new population group P_g^i, individuals with lower evaluation values are deleted until the population returns to the original population size in lines from 24 to 32. At this time, in order to prevent the individuals in the cluster from disappearing, limit the number of individuals in each cluster not to be less than half, or not less than 1. These processes are repeated until the number of generations reaches to the maximum one.

Algorithm 1 A flow of multimodal GP

```
1:  $P_0 \leftarrow random\ initialization$
2:  $C_0 \leftarrow clustering(P_0)$
3:  for $g = 0\ to\ G$ do
4:      $P'_g \leftarrow \emptyset$
5:      for $i = 1\ to\ |C_g|$ do
6:          for $j = 1\ to\ |P_g|/|C_g|$ do
7:              if $\texttt{rand}(0,1) < crossover\ probability$ then
8:                  $p1 \leftarrow \texttt{tournament}(C_g^i)$
9:                  $r \leftarrow \texttt{randInt}(0, |C_g|)$
10:                 $p2 \leftarrow \texttt{tournament}(C_g^r)$
11:                 $o \leftarrow \texttt{crossover}(p_1, p_2)$
12:             else
13:                 $p \leftarrow \texttt{tournament}(C_g^i)$
14:                 $o \leftarrow \texttt{mutation}(p)$
15:             end if
16:             $P'_i \leftarrow P'_g \cup \{o\}$
17:         end for
18:     end for
19:     $P_{g+1} \leftarrow P_g \cup P'_g$
20:     $C_{g+1} \leftarrow \texttt{clustering}(P_{g+1})$
21:     for $i = 1\ to\ |C_{g+1}|$ do
22:         $L_i \leftarrow \texttt{max}(1, |C_g^i|/2)$
23:     end for
24:     while $|P_{g+1}| > |P_g|$ do
25:         for $i = 1\ to\ |C_{g+1}|$ do
26:             if $P_{g+1} == |P_g| \vee |C_{g+1}^i| \leq L_i$ then
27:                 $break$
28:             end if
29:             $p \leftarrow \texttt{negative\_tournament}(C_{g+1}^i)$
30:             $C_{g+1}^i \leftarrow C_{g+1}^i \setminus \{p\}$
31:         end for
32:     end while
33: end for
```

IV. Proposed Method

In the previous MMGP, clustering based on tree structure similarity plays an important role in order to acquire global and local optimum solutions simultaneously. However, when redundant subtrees (e.g., $x - x$) is included in a program, an appropriate similarity of programs can not be calculated, and a program including redundant subtrees is not assigned to appropriate cluster. Specifically, programs having similar functions except for including redundant subtrees are classified into different clusters, or programs having completely different functions are classified into the same cluster due to redundant subtrees. For example, considering programs like $(x^2 + y^2) + (x - x)$ and $(z - 2w) + (x - x)$, these programs are the global and local optimal program in the benchmark shown in Section III-A, so that they should be classified into different clusters to maintain both of them. However, due to the redundant subtree $x - x$, the similarity of these programs increases, and they may be classified into the same cluster. Hence, it is possible to remove the local optimal program $(z - 2w) + (x - x)$ from the population because it has lower fitness than $(x^2 + y^2) + (x - x)$ assigned to the same cluster. In order to solve this problem, we introduce a simplification method of program to remove redundant subtrees in MMGP.

Table I: Translation rules of simplification presented in [3]

Precondition		Effective Result
if< $0(a, B, C)$	$\rightarrow$	B if $a < 0, else\ C$
if< $0(A, B, B)$	$\rightarrow$	B
$a + b$	$\rightarrow$	$c, c = a + b$
$a - b$	$\rightarrow$	$c, c = a - b$
$a \times b$	$\rightarrow$	$c, c = a \times b$
$a \div b$	$\rightarrow$	$c, c = a \div b$
$a + (b + c)$	$\rightarrow$	$c + C, c = a + b$
$a + (b - c)$	$\rightarrow$	$c - C, c = a + b$
$a - (b + c)$	$\rightarrow$	$c + C, c = a - b$
$a - (b - c)$	$\rightarrow$	$c - C, c = a - b$
$a \times (b \times c)$	$\rightarrow$	$c \times C, c = a \times b$
$a \times (b \div c)$	$\rightarrow$	$c \div C, c = a \times b$
$a \div (b \div c)$	$\rightarrow$	$c \times C, c = a \div b$
$a + (B + c)$	$\rightarrow$	$b + B, b = a + c$
$a + (B - c)$	$\rightarrow$	$b + B, b = a - c$
$a - (B + c)$	$\rightarrow$	$b - B, b = a - c$
$a - (B - c)$	$\rightarrow$	$b - B, b = a + c$
$a \times (B \times c)$	$\rightarrow$	$b \times B, b = a \times c$
$a \times (B \div c)$	$\rightarrow$	$b \times B, b = a \div c$
$a \div (B \div c)$	$\rightarrow$	$b \div B, b = a \times c$
$A \div 1$	$\rightarrow$	A
$A \div A$	$\rightarrow$	1
$0 \div A$	$\rightarrow$	0
$0 \times A = A \times 0$	$\rightarrow$	0
$A \times 1 = 1 \times A$	$\rightarrow$	A
$A + 0 = 0 + A$	$\rightarrow$	A
$A - 0$	$\rightarrow$	A
$A - A$	$\rightarrow$	0
$A \times \frac{1}{B} = \frac{1}{B} \times A$	$\rightarrow$	$\frac{A}{B}$
$A \times \frac{B}{A} = \frac{B}{A} \times A$	$\rightarrow$	B

A. Simplification

The purpose of the simplification of the program is to obtain a smaller program by removing redundant subtrees included in it. For example, subtrees such as $x - x$ and y/y can be simplified to 0 and 1, respectively. Although various simplification methods have been proposed by many researchers, this research employs a simplification method proposed by Wong et al. [3] that is based on several translation rules defined in advance.

Table I shows the translation rules of simplification presented in [3]. In Table I, constants are represented by lowercase letters (i.e., a, b, c), while variables and subtrees are represented by uppercase letters (i.e., A, B, C). "Precondition" column indicates a subtree before applying rule, while "Effective Result" indicates a simplified result after rule application. This simplification method is firstly applied to terminal nodes of the tree structure. If a rule in Table I that can be applied is found, this rule is applied to a subtree. If all nodes are checked and there are no applicable rules, the simplification method completes and the simplified tree is outputted.

B. MMGP with program simplification

In this paper, the program simplification proposed by Wang et al. is introduced into MMGP to realize appropriate clustering and to improve the search performance of MMGP. As the

Table II: Parameters

Maximum depth	4, 8
Threshold of clustering (d)	0.5, 0.6, 0.7, 0.8, 0.9
Maximum generations	500
Population size	500
Crossover probability	0.9

timing of introducing the simplification, this study examines two methods. The first method is to apply simplification to all programs during the optimization process and to evolve simplified program. This method is named as MMGP with full simplification (MMGP-FS in short). The second method is to apply simplification only when calculating the similarity of programs for clustering and to optimize programs without the simplification. This method is named as MMGP with simplification for clustering (MMGP-SC in short).

V. Experiment

A. Settings

In order to investigate the effectiveness of the simplification method in MMGP, this paper conducts an experiment using the multimodal program optimization problem presented in Section III-A. In the experiment, we compare MMGP with and without the simplification, and two approaches of introduction of the simplification, MMGP-FS and MMGP-SC.

Table II shows the parameter settings used in this experiment. The maximum depth of the program tree is set as 4, which is employed in the previous research [2], and 8, which allows to generate more complex programs. In MMGP, since the result of MMGP depends on the threshold d for terminating clustering, we compare with threshold $d = \{0.5, 0.6, 0.7, 0.8, 0.9\}$. 20 independent trials are performed and the number of generations is set to 500 generations. The population size is 500, while the crossover probability is 0.9. Four variables x, y, z, w, four function nodes $+, -, \times, \%$ (protected devision), and the real constant value c are used.

The training data has 121 input-output pairs calculated by variables of x and y by increasing them from -1 to 1 with step 0.2, while α in Eqs. (2) and (3) is 2. δ is randomly given depends on the normal distribution with the mean of 0 and the standard deviations of 0.01. The following equation is used as the fitness function:

$$fitness = \sum_{i=1}^{D} |result_i - target_i| \,, \tag{6}$$

where D is the number of the training data ($D = 121$), $result_i$ is the calculation result of the program for the i^{th} input value, and $target_i$ is the correct output value of the i^{th} training data. Depending on the value of the standard deviation, the fitness of the local optimal solution is about 20, 2, and 0.2, respectively.

B. Evaluation criteria

In order to evaluate whether the proposed method and MMGP can acquire the global and local optimal solutions simultaneously, this experiment confirms the best individual when one of the variables is restricted. Specifically, the best fitness among programs in the population that do not contain one variable of x, y, z or w obtained by each trial is confirmed. For example, in the case where the local optimal solution shown in Eq. (5) is acquired, solutions with lower (better) fitness can be found even if the variable x or y is restricted. In other words, when the variable x or y is restricted, the best fitness represents the one of the local optimal program, while when the variable z or w is restricted, the best fitness represents the one of the global optimum program.

Table III: Success rate of solution (δ = 0.01 Depth4)

	MMGP		MMGP-FS		MMGP-SC	
d	global	local	global	local	global	local
0.5	100%	100%	100%	100%	100%	100%
0.6	100%	100%	100%	100%	100%	100%
0.7	100%	100%	100%	100%	100%	100%
0.8	100%	100%	100%	100%	100%	100%
0.9	100%	90%	100%	80%	100%	90%

Table IV: Success rate of solution (δ = 0.01 Depth8)

	MMGP		MMGP-FS		MMGP-SC	
d	global	local	global	local	global	local
0.5	100%	100%	100%	100%	100%	100%
0.6	100%	100%	100%	100%	100%	100%
0.7	100%	100%	100%	100%	100%	85%
0.8	100%	95%	100%	100%	100%	70%
0.9	100%	60%	100%	80%	100%	40%

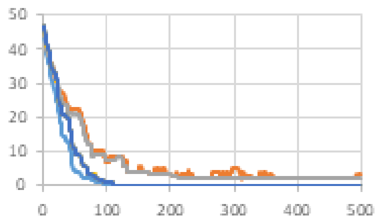
C. Experimental result

Tables III and IV show the success rate of finding the global and local optimum solution in MMGP with and without simplification when the maximum depth of programs is 4 and 8. In these tables, "d" column indicates the difference of the threshold of the clustering.

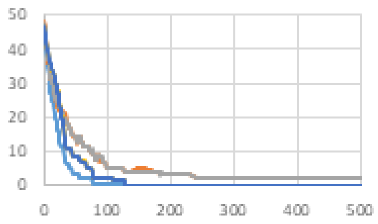
From these results, it can be indicated that the global and local optimum solution can be acquired at a high rate by appropriately setting the threshold d in all methods. Especially, when $d = 0.6, 0.7$ all method achieve the success rate of 100%. On the other hand, when $d = 0.8, 0.9$, it is found that the success rate of the local optimum solution decreases. In particular, it can be seen that MMGP-SC which applies simplification only at the clustering shows that the success rate of the local optimal solution is lower than other methods.

Next, Figs. 3, 4 show the transition of the best fitness without each variable. The vertical axis shows the fitness, while the horizontal axis shows the generation. In these figures, the results of the clustering threshold $d = 0.6$ are shown. The light blue, orange, gray, yellow, and blue lines represent the results with no variable restriction (No restriction), restricting x (without x), restricting y (without y), restricting z (without z), and restricting w (without w), respectively.

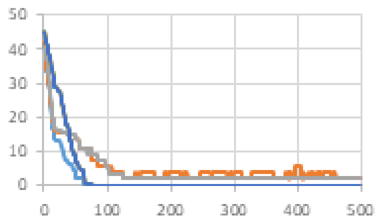
The reason why the result of the threshold $d = 0.6$ is only shown in this paper is that the speed of discovering the local optimal solution is relatively faster than other thresholds. From these figures, the higher the threshold, the slower the speed of finding the local optimal solution, and the further the MMGP-SC when the threshold $d = 0.6$ is slower than the



(a) Original MMGP ($d = 0.6$)

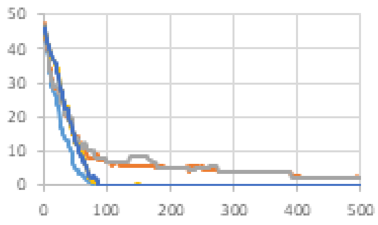


(b) MMGP-FS ($d = 0.6$)

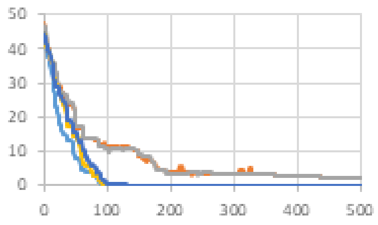


(c) MMGP-SC ($d = 0.6$)

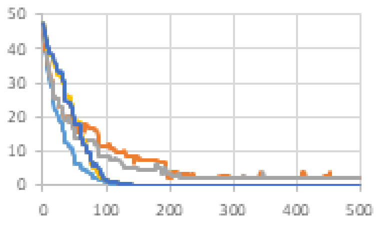
Figure 3: Transition of the best fitness Without each variable (Maximum depth = 4). Light blue: No restriction. Orange: Without x. Gray: Without y. Yellow: Without z. Blue: Without w.



(a) Original MMGP ($d = 0.6$)



(b) MMGP-FS ($d = 0.6$)



(c) MMGP-SC ($d = 0.6$)

Figure 4: Transition of the best fitness Without each variable (Maximum depth = 8). Light blue: No restriction. Orange: Without x. Gray: Without y. Yellow: Without z. Blue: Without w.

other methods. However, in all other cases, there is not much difference in the transition, and it can be seen that the local optimum solution can be acquired.

From these results, it is revealed that the simplification does not significantly affect the search performance of MMGP. In addition, it is shown that the search performance of the local optimal solution significantly decreases when the clustering threshold is set large value.

D. Discussion

The experimental results showed that the program simplification does not significantly affect the search performance of MMGP. In this section we will discuss this reason. Figure 5 shows the number of clusters during the search process. In Figure 5, the vertical axis represents the number of clusters, while the horizontal axis represents the number of generations. On the other hand, Figure 6 shows the number of simplifications applied per generation. In Figure 6, the vertical axis represents the number of simplifications applied, while the horizontal axis represents the number of generations. In both figures, light blue color indicates the result of MMGP without simplification, orange color indicates that of MMGP-FS, while gray color indicates that of MMGP-SC. In both cases, the results are shown when the maximum depth is 8, and the threshold $d = 0.6$.

In the following subsections, we discuss the results of two proposed method separately.

1) MMGP-FS: From Figure 5, it can be seen that, by introducing simplification, the number of clusters of MMGP-FS is smaller than that of conventional MMGP. This is because, in the conventional MMGP, programs that contain redundant subtrees have possibility to have low similarity among them, and they are possibly classified into the different clusters. On the other hand, in MMGP-FS, such programs are simplified every generations and are classified into the same cluster. By this, the number of clusters in MMGP-FS decreases rather than the conventional MMGP.

However, from Figure 6, it is confirmed that the simplification of the program is applied mostly at the initial stage of the evolution process (especially before 10th generation). After that, the simplification is hardly applied. Specifically, the simplifications are applied to about 25 individuals out of 500 ones generated every generation, which is about 5% per one generation. From this analysis, it is indicated that the simplifications affects the search progress only in the initial stage of the search in MMGP-FS. This causes that no difference is found between MMGP-FS and the conventional one.

2) MMGP-SC: Similar to MMGP-FS, MMGP-SC, which applies simplification only in the clustering phase, also decreases the number of clusters at the beginning of the search

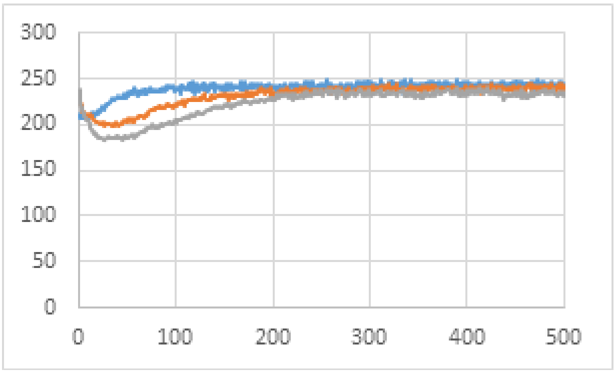


Figure 5: The number of clusters in $d = 0.6$. Light blue: Original MMGP, Orange: MMGP-FS. Gray: MMGP-SC

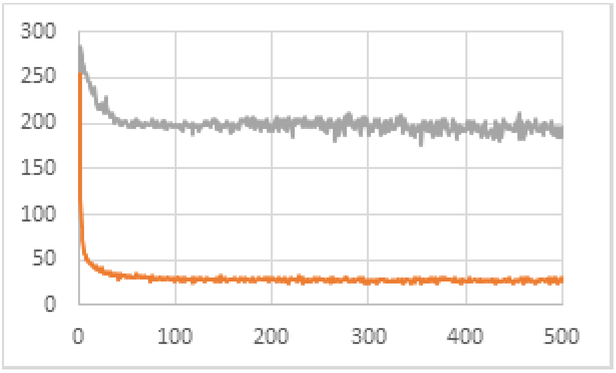


Figure 6: The number of applications of simplification in $d = 0.6$. Orange: MMGP-FS. Gray: MMGP-SC

as shown in Fig. 5. However, the number of clusters increases as the search progress proceeds. On the other hand, the number of simplifications has been in the range of 150 to 200 from Fig. 6. This is because the simplification is applied only in the clustering phase but programs are evolved as the structure without being applied the simplification.

The reason why there is no big difference in optimization performance compared with the conventional MMGP is that the simplification does not significantly affect the result of survival selection or genetic operation in MMGP. For example, in the case where there are two programs of a program $x^2 + y^2$ and a program $x^2 + y^2 + z - z$ including a redundant subtree, these programs are not classified into the same cluster in the conventional MMGP. Both of them have high fitness and therefore have a high probability of survive in the next generation. On the other hand, in MMGP-SC, these programs are classified into the same cluster, but since both have high fitness, they are not eliminated. Additionally, in genetic operations (especially crossover), parent individuals are selected from a target cluster and a randomly selected cluster. According to this selection method, if the number of clusters is large, the result of a crossover operator is not so different whether two programs with and without redundant subtree are classified into the same cluster or not. From this, it is considered that no significant difference was found between the results of the conventional MMGP and MMGP-SC.

VI. Conclusion

This paper introduced the simplification method to MMGP in order to solve the problem that the similarity of tree structures (programs) is not properly evaluated due to redundant subtrees. Specifically, this paper proposed two approaches of introduction of the simplification, one applies the simplification to all programs and evolve simplified programs (named as MMGP-FS), while another applies the simplification only when calculating similarity of programs in the clustering process and evolves programs before being simplified (named as MMGP-SC).

To analyze the influence of the simplification in MMGP, the experiments were conducted on the multimodal program optimization benchmark problem proposed in the previous research. In the experiment, MMGPs with and without the simplification are compared. The experimental results showed that there is no big difference in the search performance among all MMGP variants. This is because in MMGP-FS, simplification is hardly applied except for the initial generation, and the influence is small. On the other hand, in MMGP-SC, clustering results due to simplification have little influence on survival selection and genetic operation results, especially when the number of clusters is large.

In this experiment, it was clarified that the introduction of simplification for deleting redundant subtrees has a small influence on the search performance of MMGP. However, this result may change depending on genetic manipulation changes and characteristics of multimodal program optimization problem dealt with. Therefore, from now on, we will analyze the influence of genetic operations and the effect of simplification, and apply MMGP to more complex multimodal program optimization problems other than benchmark problem used in this research.

References

[1] John Koza, “Genetic Programming On the Programming of Computers by Means of Natural Selection,” MIT Press, 1992.

[2] Shubu Yoshida, Tomohiro Harada, Ruck Thawonmas, “Multimodal Genetic Programming by Using Tree Structure Similarity Clustering,” IEEE 10th International Workshop on Computational Intelligence and Applications (IWCIA2017), pp. 85-90, 2017.

[3] Phillip Wong and Mengjie Zhang, “Algebraic simplification of GP programs during evolution,” GECCO ’06 Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 927-934, 2006.

[4] Yang Rui, Panos Kalnis, and Anthony KH Tung, “Similarity evaluation on tree-structured data,” Proceedings of the 2005 ACM SIGMOD international conference on Management of data, ACM, pp. 754-765, 2005.

[5] David E. Goldberg, “Genetic Algorithms in Search, Optimization and Machine Learning,” Addison-Wesley Long man Publishing Co., Inc., Boston, MA, USA, 1989.