https://doi.org/10.20965/jaciii.2021.p0177

Paper:

Self-Organized Subpopulation Based on Multiple Features in Genetic Programming on GPU

Keiko Ono[*] **and Yoshiko Hanada**[**]

[*]Doshisha University
1-3 Tatara Miyakodani, Kyotanabe, Kyoto 610-0394, Japan
E-mail: kono@mail.doshisha.ac.jp
[**]Kansai University
3-3-35 Yamate-cho, Suita, Osaka 564-8680, Japan

[Received August 9, 2019; accepted December 3, 2020]

Genetic Programming (GP) is an Evolutionary Computation (EC) algorithm. Controlling genetic diversity in GP is a fundamental requirement to obtain various types of local minima effectively; however, this control is difficult compared to other EC algorithms because of difficulties in measuring the similarity between solutions. In general, common subtrees and the edit distance between solutions is used to evaluate the similarity between solutions. However, there are no clear guidelines regarding the best features to evaluate it. We hypothesized that the combination of multiple features helps to express the specific genetic similarity of each solution. In this study, we propose a self-organized subpopulation model based on similarity in terms of multiple features. To reconstruct subpopulations, we introduce a novel weighted network based on each normalized feature and utilize network clustering techniques. Although we can regard similarity as a correlation network between solutions, the use of multiple features incurs high computational costs, however, calculating the similarity is very suitable for parallelization on GPUs. Therefore, in the proposed method, we use CUDA to reconstruct subpopulations. Using three benchmark problems widely adopted in studies in the literature, we demonstrate that performance improvement can be achieved by reconstructing subpopulations based on a correlation network of solutions, and that the proposed method significantly outperforms typical methods.

Keywords: genetic programming, subpopulation model, genetic diversity, multiple features

1. Introduction

In Genetic Programming (GP), it is important to maintain genetic diversity, which can be achieved by improving the replacement strategies, such as re-selection and crowding [1], and the sharing function [2]. A subpopulation model easily encourages genetic diversity and avoids premature convergence toward local optima; it frequently shows a better search performance than single population models. Studies have been conducted on subpopulation models in GP to further improve its performance. For example, Hu et al. [3, 4] proposed the hierarchical fair competition model (HFX), in which individuals are separated into subpopulations according to their fitness values and evolve in a fair competition environment. Hornby [5, 6] proposed the age-layered population structure model (ALPS), in which individuals are evaluated by their age, that is, a measure of how long the genetic material has been in the population, and separated into subpopulations according to this age measure. Luna et al. [7] proposed a meaning-based model for data mining to extract mining rules. Moreover, other researchers have proposed migration models [8–11], in which some individuals migrate to the next subpopulation to maintain genetic diversity. HFX and ALPS require appropriate boundary parameters for each subpopulation, which are difficult to determine: a fitness value in HFX and an age value in ALPS. By contrast, in migration models, it is not necessary to set boundary parameters, and these models can be easily implemented in various applications. Therefore, we focus on migration models in this study and consider enhancing the subpopulation models in the GP framework.

To enhance the subpopulation model, it is important to consider the amount of genetic diversity that should be maintained during evolution. The higher the genetic diversity of a subpopulation, the wider the range of characteristics of each solution, which hinders the convergence to local minima. Previous subpopulation models focused on increasing genetic diversity by dividing solutions into some populations; the level of genetic diversity was not controlled. The genetic diversity can be easily controlled if the distance between solutions can be measured as Euclidian distance. However, this distance is difficult to measure because solutions in GP are typically expressed by a tree. In general, the similarity based on common subtrees and edit distance between solutions is used to evaluate the distance between solutions. Nevertheless, the best manner to express distance is uncertain because each problem domain has its own most effective feature. We hypothesize that multiple features extracted from each solution can enable an improved evaluation of the distance between solutions compared to a single feature, and adaptive reconstruction of subpopulations consisting of similar individuals in terms of multiple features can enhance local search. Moreover, both mutations in each subpopulation and independent evolution using the subpopulation model can enhance genetic diversity.

In this study, we propose a novel self-adaptive subpopulation model that, on the basis of multiple features, creates a subpopulation in which similar individuals exist and enhances mutation in this subpopulation. More specifically, to create subpopulations with similar features, we introduce a weighted network of individuals based on similarity obtained from features, and separate individuals into subpopulations using network clustering. Using these strategies, we can control the balance between local search and genetic diversity to avoid redundant searching. Obtaining similarity based on multiple features is a time-consuming task, whereas calculating the similarity between individuals is very suitable for parallelization. Therefore, we use CUDA to construct a correlation network. It is well known that a considerable computational effort is required to evaluate fitness in GP. The first study using CUDA (GPUs) in GP was reported in [12]; other researchers have also proposed using CUDA models in GP to enhance acceleration [13–16]. To the best of our knowledge, our proposed method is the first subpopulation model in GP, executed on CUDA, which reconstructs subpopulations according to the individuals' features.

The remindar of this paper is organized as follows. In Section 2, we formulate the optimization problem discussed in this study and provide a detailed explanation of the subpopulation model in GP. In Section 3, we describe the CUDA-based proposed adaptive subpopulation model in GP in detail. We first present an overview of the proposed method, and then explain the CUDA-based approach of the proposed method with the objective of reducing computational effort. Moreover, we explain a technique for adaptively reconstructing subpopulations using the proposed method. In Section 4, we examine the effectiveness of the proposed method through experiments using three benchmark problems widely adopted in studies in the literature. We first explain the implementation of the proposed method, and then introduce two comparison methods to analyze the effects of the proposed method.[1]

2. Problem Definition and Subpopulation Model in GP

In the framework of GP, we consider the problem of minimizing a function $f(x)$, where its input variable x is represented as a labeled ordered tree. Given the total population size N and a crossover-mutation-selection strategy, a common approach to improving a GP search is to

Algorithm 1 Algorithm for subpopulation model

```
R0: Set $g \leftarrow 1$.
R1: Initialize individuals in each subpopulation $P^m$.
R2: If $g \le G$, then perform Steps R3 to R8; otherwise, stop.
R3: Apply genetic operations, such as selection and crossover.
R4: If $g$ is a migration generation, then perform Steps R5 to R7; otherwise, perform Step R8.
R5: Choose the best $n$ individuals $B^m$ and the worst $n$ individuals $W^m$ in $P^m$.
R6: Send $B^m$ to the next subpopulation $P^{m+1}$.
R7: Replace $W^m$ with $B^{m-1}$, that is, the best individuals in the next subpopulation $P^{m-1}$.
R8: Set $g \leftarrow g+1$.
```

apply subpopulation model methods. In the GP subpopulation model, the total population V is partitioned into M subpopulations, and the subpopulations execute GP searches for minimizing $f(x)$ in parallel, although they exchange information by migration. There are several migration topologies such as ring, random, and grid, but the ring topology is the most basic. Thus, we focus on the ring topology and refer to it as the subpopulation model in GP in this paper. It should be noted that the number of individuals in each subpopulation does not change during searching.

We begin by recalling the details of the subpopulation model in GP with ring topology. Let P^m $(m = 1,\dots,M)$ denote the m-th subpopulation, where the number of individuals in P^m is $|P^m| = |V|/M$. The individuals in P^m are denoted by $V^m = \{v_i^m;\ i = 1,\dots,I\}$, and the total population is obtained by

$$V = \bigcup_{m=1}^{M} V^m,$$

where $I = |V|/M$, and $|V|$ is the number of all individuals. The total population is $V = \{v_i;\ i = 1,\dots,I \times m\}$. It should be noted that the set of subpopulations $\mathscr{P} = \{P^m;\ m = 1,\dots,M\}$, and the maximum generation is G. The algorithm of the subpopulation model in GP is shown as **Algorithm 1**. Here, the topology type of the subpopulation model in GP is a ring; therefore, the subpopulation P^0 receives individuals from P^M, and P^M sends them to P^0.

3. Self-Organized Subpopulation Model in GP on GPU: SoS-GP

3.1. Overview of SoS-GP

In sequence, we consider enhancing the subpopulation model in GP for the minimization problem of the objective function $f(x)$.

In GP, the structure of an individual can be expressed by a tree. There are multiple features of an individual,

1. In our conference paper [17], we proposed a basic framework for such a migration strategy. This paper extends our preliminary study [17] by improving the technique of measuring the similarity between subpopulations incorporating mutation.

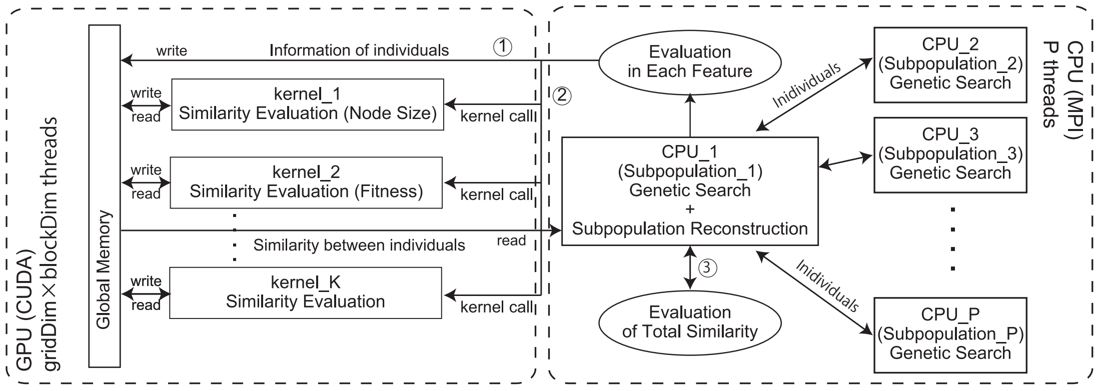

GPU (CUDA)
gridDim × blockDim threads
Global Memory
write
Information of individuals
①
write
read
kernel_1
Similarity Evaluation (Node Size)
kernel call
②
write
read
kernel_2
Similarity Evaluation (Fitness)
kernel call
Similarity between individuals
read
write
read
kernel_K
Similarity Evaluation
kernel call
Evaluation in Each Feature
Individuals
CPU_1 (Subpopulation_1) Genetic Search + Subpopulation Reconstruction
③
Evaluation of Total Similarity
Inidividuals
CPU_2 (Subpopulation_2) Genetic Search
CPU_3 (Subpopulation_3) Genetic Search
CPU_P (Subpopulation_P) Genetic Search
CPU (MPI)
P threads

Fig. 1. Proposed model using CUDA.

such as fitness, tree size, and subtree. However, fitness is used only as a feature for migration in the common subpopulation model in GP; elite individuals with lower fitness values than others in the same subpopulation migrate to other subpopulations. According to the building block hypothesis, partial solutions are assembled into the entire solution. Here, we hypothesize that incorporating the structure information of individuals in the migration strategy is important for enhancing the combination of partial solutions. We also assume that the balance between local and genetic searches is important. If abrupt changes occur in the search area, individuals randomly evolve, which leads to disruption of genetic operations. To effectively explore partial solutions, we incorporate a local search into the subpopulation model in GP. Thus, from the viewpoint of encouraging local search to combine partial solutions, we suppose that automatic reconstruction of subpopulations, which consist of similar individuals in terms of their multiple features, enhances local search and improves the combination of partial solutions. Therefore, we propose a self-adaptive subpopulation model in GP. In the proposed method, we generate a weighted network of individuals based on fitness and node size and create subpopulations by network clustering.

However, fitness evaluation is the most time-consuming part of reconstructing subpopulations. Moreover, the proposed method evaluates the similarity between individuals based on their features, which also increases the computational effort. To solve this issue, we use GPGPU to create a similarity network using multiple features. **Fig. 1** shows the model of the proposed method. The proposed method forms subpopulations in which individuals are evolved in parallel. In spite of migration, subpopulations are automatically reconstructed by the similarity network of individuals V. In Section 3.2, we describe the proposed model using GPGPU in detail. The algorithm of the proposed method is shown as **Algorithm 2**.

3.2. Feature Evaluation Using GPGPU

In the proposed method, self-organized subpopulation model in GP on GPU (SoS-GP), an automatic reconstruction of subpopulations that consist of similar individuals in terms of their features, is conducted to enhance local search. In GP, as an individual can be represented by a tree, there are several features that can be used to measure their similarity. We assume that the similarity can be accurately measured using multiple features. However, there are $|V|^2/2$ combinations for one type of feature when the number of individuals is $|V|$, and the required computational effort is thus increased. To decrease the computational effort when multiple features are applied, we use GPGPU to create a similarity network.

Algorithm 2 Algorithm for the proposed method

```
P0: Set $g \leftarrow 1$.
P1: Initialize individuals in each subpopulation $P^m$, where the number of individuals $I$ is set to $|V|/M$.
P2: If $g \leq G$, then perform Steps P3 to P13; otherwise, stop.
P3: Apply selection and crossover.
P4: Apply mutation $\gamma$ times.
P5: If $g$ is a reconstruction generation, then perform Steps P5 to P11; otherwise, perform Step P13.
P6: Send $V^m$ in each $P^m$ to the subpopulation $P^0$.
P7: If the subpopulation is $P^0$, then perform Steps P7 to P11; otherwise, perform Step P12.
P8: Evaluate the similarity matrix $H$ between $v_i$ and $v_j$ $(i \neq j)$.[2]
P9: Create a network based on $H$.
P10: Divide the population $V$ into $M$ subpopulations $\hat{V}^m$ by network clustering to reconstruct $V^m$.[3]
P11: If $|V^m|$ is not 1, then perform Steps P11 to P12; otherwise, perform Step P13.
P12: Send $\hat{V}^m$ to $P^m$.
P13: Replace $V^m$ with $\hat{V}^m$.
P14: Set $g \leftarrow g+1$.
```

Figure 1 shows the model of the proposed method that uses both CUDA and MPI. The proposed method evolves subpopulations in parallel and applies the same type of genetic operators in all subpopulations. MPI is

2. Here, the similarity H is a $V \times V$ matrix, and the element of H is h_{uv}. For more detail, see Section 3.2.
3. For more detail, see Section 3.3.

mostly known for message-passing multiprocessing programming and allows a subpopulation model to be easily implemented using multiple processors on a single system or a cluster of systems. However, the proposed method evaluates the similarity between individuals in terms of each feature in a different CUDA kernel in parallel to achieve acceleration. Therefore, the proposed method uses both CUDA and MPI. The three steps for evaluating the similarity are as follows:

1. Gather individuals in each subpopulation to subpopulation 1; the information of individuals is sent to a global memory;

2. Evaluate the similarity in terms of each feature using a different kernel: node size or fitness. Each similarity result is sent to subpopulation 1;

3. Combine the similarity results using cuBLAST.

Here, to combine the similarities evaluated by different kernels, we use cuBLAST, which is the CUDA basic linear algebra subroutine library on NVIDIA CUDA runtime. It automatically allocates the required matrices and vectors in the GPU memory space. cuBLAST has many calculators of matrix vectors divided into three levels in terms of parallelization. In the proposed method, each similarity consists of $|V|^2/2$ elements; therefore, combining these elements requires a matrix and matrix multiplication operation. This operation is a BLAST Level 3 operation, which has higher scalability than other levels. Therefore, the proposed method uses the cublasGetMatrix operator in cuBLAST to process multiple feature similarities.

3.3. Newman Clustering

Newman clustering [18, 19] is the most well-known algorithm for extracting communities. It uses a quality function known as "modularity Q" over the possible divisions of a network. By optimizing the modularity, density-connected groups of vertices with only sparse connections between groups are detected. As in the proposed method we reconstruct subpopulations based on the similarity between individuals, Newman clustering can be applied to detect subpopulations in which individuals have a similar characteristic easily if we refer the similarity matrix H to an adjacency matrix in a weighted network. In this study, we used the Newman clustering algorithm to reconstruct subpopulations. We refer to an individual as a vertex, and an element of similarity H is the weight between vertices. The modularity Q is defined as

$$Q = \sum_i (e_{ii} - a_i^2),$$

where e_{ii} is a portion of the weight of the joint vertices in community i in the total weight, and a_i is a portion of the weight of nodes that connect vertices in community i to other groups in the total weight. Let A_{uv} be an element of the similarity matrix H. Thus[4]

Algorithm 3 Algorithm for Newman clustering

```
S1: Evaluate the initial values of $\Delta Q_{ij}$ and $a_i$ according to Eqs. (1) and (2).
S2: Select the largest element of each row of the matrix $\Delta Q$, and save it to the max-heap.
S3: Select the largest $\Delta Q_{ij}$ from the max-heap, and join the corresponding communities.
S4: Update $\Delta Q$, the max-heap and $a_i$ according to Eq. (3), and increment $Q$ by $\Delta Q_{ij}$.
S5: Repeat Steps 2 and 3 until only one community remains.
S6: Extract communities where $Q$ is the maximum value.
```

$$A_{uv} = \begin{cases} h_{uv} & \text{if the weight between vertices } u \text{ and } v \text{ is not 0,} \\ 0 & \text{otherwise.} \end{cases}$$

a_i is obtained by

$$a_i = \sum_j e_{ij},$$

$$e_{ij} = \frac{1}{2W} \sum_u \sum_v A_{uv} \delta(c_u, i) \delta(c_v, j),$$

where W is the amount of weight and δ function $\delta(i, j)$ is 1 if $i = j$ and 0 otherwise. If community i belongs to the community c_u, then $\delta(c_u, i) = 1$. As the first step, let each vertex be the sole member of a community of one; then, $e_{ij} = 1/2W$ if i and j are connected and 0 otherwise, and $a_i = k_i/2W$, where

$$k_u = \sum_v A_{uv}.$$

Thus, the initial ΔQ_{ij} is obtained by

$$\Delta Q_{ij} = \begin{cases} \frac{1}{2W} - \frac{k_i k_j}{(2W)^2} & \text{if the weight between vertices } i \text{ and } j \text{ is not 0,} \\ 0 & \text{otherwise,} \end{cases} \quad \ldots\ldots (1)$$

and

$$a_i = \frac{k_i}{2W}. \quad \ldots\ldots (2)$$

The Newman clustering algorithm with the similarity matrix H is shown as **Algorithm 3**.

$$\Delta Q'_{jk} = \begin{cases} \Delta Q_{ik} + \Delta Q_{jk} & \text{if community } k \text{ is connected to both } i \text{ and } j, \\ \Delta Q_{ik} - 2a_j a_k & \text{if community } k \text{ is connected to } i \text{ but not to } j, \\ \Delta Q_{jk} - 2a_j a_k & \text{if community } k \text{ is connected to } j \text{ but not to } i. \end{cases} \quad \ldots\ldots (3)$$

4. A_{uv} is an element of the adjacency matrix of the network.

4. Experimental Evaluation

Through experiments using three benchmark problems widely adopted in studies in the literature, we evaluated the effectiveness of the proposed method. Moreover, we analyzed the effects of mutation and genetic diversity, and verified that the proposed framework based on similarity between individuals can control genetic diversity.

4.1. Implementation

Here, we explain the implementation of Step P7 of the proposed method in the experiments. Although there are several features for measuring the similarity between individuals i and j, we used the difference in the fitness value and in the node size between i and j for simplicity. The similarity matrix H^f in terms of the difference in the fitness value is obtained by

$$H^f_{ij} = \begin{cases} h^f_{ij} & f(v_i) \neq f(v_j), \\ 0 & \text{otherwise}, \end{cases}$$

where $h^f_{ij} = \eta / |f(v_i) - f(v_j)|$ and η is a real-valued parameter. We set it to 1.0 in our experiments. H^f_{ij} becomes large when $f(v_i)$ and $f(v_j)$ are scalar. Moreover, the similarity matrix H^n in terms of the difference in node size is obtained by

$$H^n_{ij} = \begin{cases} h^n_{ij} & |v_i| \neq |v_j|, \\ 0 & \text{otherwise}, \end{cases}$$

where $h^n_{ij} = \eta / ||v_i| - |v_j||$. We constructed the similarity of individuals H as follows. Let $\hat{H}^f$ and $\hat{H}^n$ be the normalized matrices of H^f and H^n, respectively. Let H denote the total similarity with respect to the fitness values and node size. Then, we define H by

$$H = \alpha \hat{H}^f + (1-\alpha)\hat{H}^n.^5 \quad \dots\dots\dots (4)$$

A matrix and matrix multiplication operation were used to normalize the matrices H_f and H^n and to obtain H in Eq. (4); therefore, we used the cublasGetMatrix operation in cuBLAST.

4.2. Comparison Methods

The proposed method evaluates the similarity between individuals using a combination of H^f and H^n. H^f measures the difference between individuals in fitness values, and H^n measures the difference in node size. To analyze the effect of H^f and H^n separately, we introduce two methods as extreme cases of the proposed method and compare them with the proposed method, as described in Section 4.5. First, we investigate the effectiveness of similarity in terms of fitness values. The similarity H is defined by

$$H = \hat{H}^f.$$

5. α is a parameter. In our experiments, we adopted $\alpha = 0.5$.

We refer to this method as the *F-method*. Then, we investigate the effectiveness of the similarity in terms of node size. Similarity H is then defined by

$$H = \hat{H}^n.$$

We refer to this method as the *N-method*.

4.3. Benchmark Problems

We evaluated the effectiveness of the proposed method on well-known benchmark problems, the symbolic regression problems. Here, we briefly explain these problems (see [20–22] for detailed descriptions).

We investigated the symbolic regression problem for the function space $\mathscr{X}$ constructed by the labeled ordered trees of functional nodes $\{+, -, \times, /, \text{sin}, \text{cos}, \text{log}\}$ and terminal nodes $\{s, 0.05, 0.10, 0.15, 0.20, \ldots, 0.95, 1.00\}$, where s denotes a variable. Our training set was composed of 30 data points $\{(s_j, x_*(s_j)) \in \mathbf{R}^2;\ j = 1, \ldots, 30\}$, where $s_j = 0.2(j-1)$, and $x_*(s) \in \mathscr{X}$ is the true function to be identified. For any $x(s) \in \mathscr{X}$, we define the fitness $f(x)$ by

$$f(x) = 50 \sum_{j=1}^{30} |x(s_j) - x_*(s_j)|,$$

and consider the maximization problem of $f(x)$. In our experiments, we employed three functions $x_*(s)$,

Function A :

$$x_*(s) = (2 - 0.3s)\sin(2s)\cos(3s) + 0.11s^2,$$

Function B :

$$x_*(s) = s\cos(s)\sin(s)\left(\sin^2(s)\cos(s) - 1\right),$$

Function C :

$$x_*(s) = s^3\cos(s)\sin(s)e^{-s}\left(\sin^2(s)\cos(s) - 1\right).$$

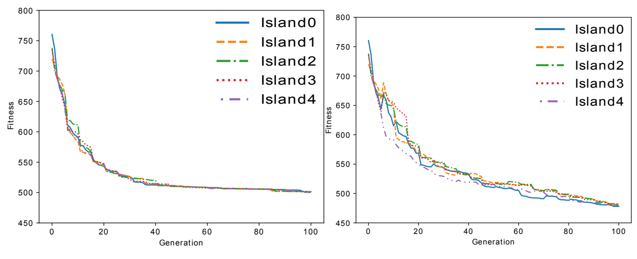
4.4. Experimental Setting

To evaluate the benchmark problems, we used the following standard parameter settings [20–22]: the recombination rate was 1.0, the mutation rate was 0.0, random selection was used (non-elitist), the total population size was $N = 250$, the number of subpopulations was $M = 5$, the maximum depth was 16, the number of generations was $I = 100$, the migration interval was 10, and the initial individuals were created by using "ramped half-and-half" with maximum depth. **Fig. 2** shows the history of fitness in terms of Ring and the proposed method. From this figure, we observe that the Ring method converged around the 50th generation; therefore, we set the number of generations to $I = 100$.

All our experiments were conducted on a single PC with 6 Intel Xeon E5-1660 3.3 GHz processors, with 24 GB of memory and NVIDIA Quadro K6000 running on Linux. The NVIDIA Quadro K6000 is capable of approximately 5.2 TFlops and 2880 CUDA cores.

4.5. Performance Evaluation

We first compared the proposed method with the N, F, Ring, and Simple methods in terms of solution qual-



(a) Ring (b) Proposed

Fig. 2. History of fitness (Function A).

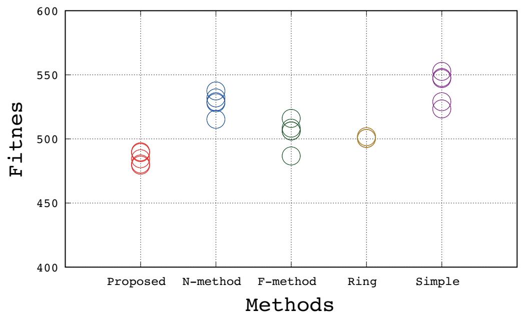
ity. In the experiments, we conducted 20 trials and evaluated the best individuals x_f^m in P^m at the final generation, that is, $f(x_f^m) \geq f(x)$ $(\forall x \in V^m)$. **Fig. 3** graphically presents $f(x_f^m)$, where $m = 1, \ldots, M$ and $M = 5$.

Figure 3 shows that the Simple method was not superior to the other methods in any functions. This result implies the necessity of migration in a subpopulation model. **Fig. 3(a)** shows the results of Function A. The N-method was not superior to the proposed, F and Ring methods in performance, and the F-method was comparable to the Ring method. Moreover, the performance of the proposed method was superior to that of these methods.

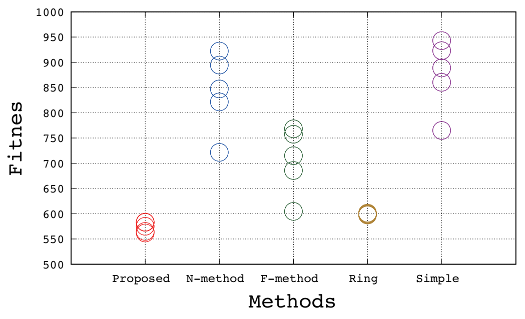
Figure 3(b) shows the results for Function B. It is observed that the performance of the Ring method compared to that of the F-method was superior for Function B, and similar for Functions A and C. By contrast, the performance of the F-method was superior to that of the N-method. These results imply that similarity based on fitness is effective for solution quality. In Section 4.6, we analyze the reason why the F-method was superior to the N-method.

From **Fig. 3(c)**, it is observed that the performance of the proposed method was comparable to that of the F-method for Function C, whereas for other functions the proposed method outperformed the N and F methods. These results imply that the proposed method that reconstructs subpopulations based on multiple features of individuals, that is, incorporating both the similarity of node size $\hat{H}^n$ and the similarity of fitness $\hat{H}^f$, leads to performance improvement.

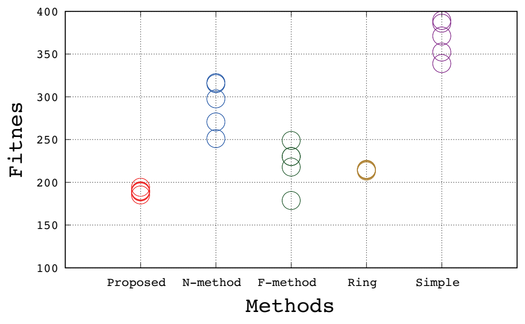
Next, we evaluated the effects of mutation in Step P4. We hypothesized that the proposed SoS model can enhance local search by constructing a subpopulation; it can simultaneously maintain genetic diversity through mutation. Here, individuals in the same subpopulation have relatively the same features. We evaluated the fitness of each individual every five generations while changing the number of mutations γ. **Figs. 4–6** show the distributions of individuals every five generations using the proposed method. We found that fitness became lower than that of the 5th generation as the number of generations increased. In particular, the results showed better performance in all problems when γ was set to 10. From these results, we confirmed that the balance in terms of mutation rate is important for performance improvement.



(a) Function A



(b) Function B

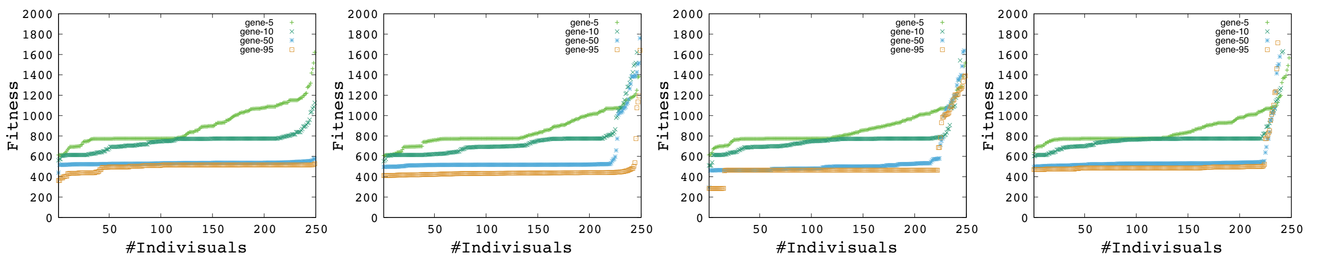


(c) Function C

Fig. 3. Performance evaluation.

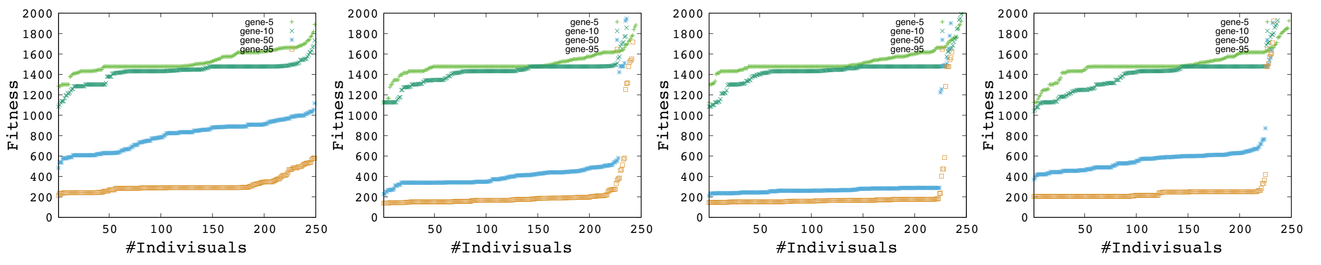
To verify why the proposed method showed better performance when γ was set to 10, we analyzed the differences between the elite and the other individuals. **Figs. 7–9** show the differences between the best individual and each individual in terms of fitness and edit distance, in which individuals evolve without mutation when γ is set to 0. Compared to $\gamma = 5$, 10, and 15, the plots in **Figs. 4(a)**, **5(a)**, and **6(a)** were gathered around the upper-left corner. These results without mutation show less diversity, as expected from the high similarities in each subpopulation. By contrast, diversity increases as the number of mutations increases. From these results, we verify that the proposed method can control local search and genetic diversity by the proposed SoS and mutation, and an appropriate balance exists. These results demonstrate the effectiveness of the proposed method.

Moreover, we evaluated the average of nodes in terms of the best individuals x_f^m in P^m at the final generation to investigate the influence of the bloat phenomenon in GP.



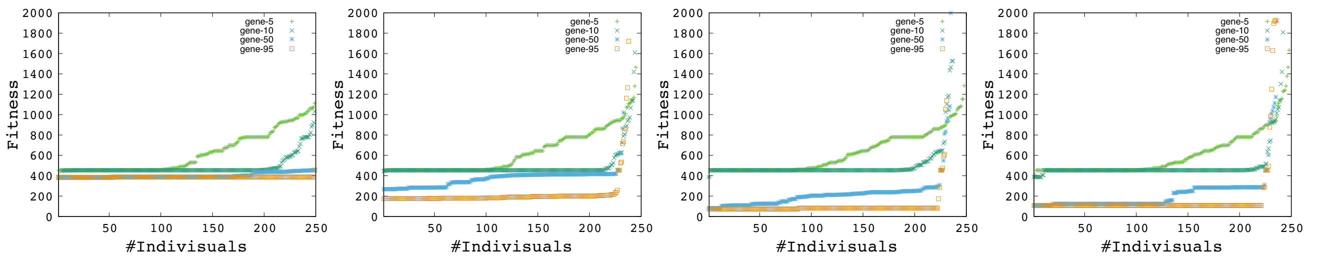
(a) $\gamma = 0$ (b) $\gamma = 5$ (c) $\gamma = 10$ (d) $\gamma = 15$

Fig. 4. Distribution of individuals (Function A).



(a) $\gamma = 0$ (b) $\gamma = 5$ (c) $\gamma = 10$ (d) $\gamma = 15$

Fig. 5. Distribution of individuals (Function B).



(a) $\gamma = 0$ (b) $\gamma = 5$ (c) $\gamma = 10$ (d) $\gamma = 15$

Fig. 6. Distribution of individuals (Function C).

A common method to prevent bloat is to limit the maximum number of nodes; therefore, we introduced a new function shown in Eq. (5).

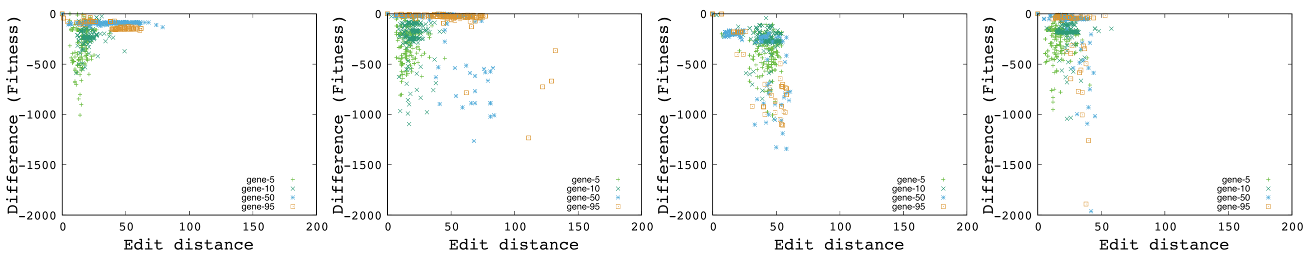
$$g(x) = 50 \sum_{j=1}^{30} |x(s_j) - x_*(s_j)| + \beta \cdot ||x||, \quad . \quad . \quad . \quad (5)$$

where $||\ ||$ indicates the number of nodes in x and β is a weight parameter that controls the number of nodes. **Fig. 10** shows the fitness values in terms of fitness $g(x)$ and $f(x) = g(x) - \beta \cdot ||x||$ with respect to β using the proposed method with $\gamma = 0$. From **Figs. 3** and **10**, Functions A and B with $\beta = 1$ showed better performance than the proposed method. As the first step, the proposed method does not adopt an approach to reduce bloat; however, these results indicate that we should consider the bloat in GP in the proposed method for further performance improvement by incorporating a bloat resistance method.

4.6. Behavior Analysis

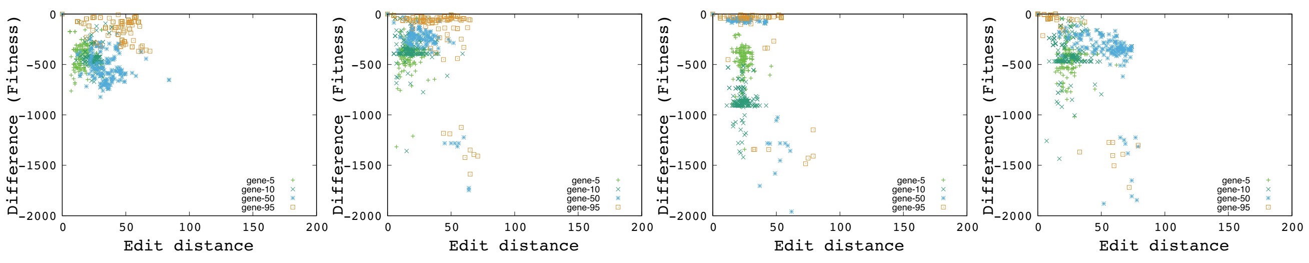
In the proposed method, multiple features are used to create a correlation network of individuals. Therefore, considerable computational effort is required by the proposed method compared to the Ring and Simple methods for evaluating the similarity between individuals. First, we investigated the scalability of the proposed method in terms of the number of individuals and used this method over 20 trials to analyze the best and worst trials based on the average execution time, as shown in **Fig. 11**.

We set the number of individuals as 50, 150, 250, 350, and 450. The average execution time is different in each problem because although the maximum depth of an individual is limited, the number of nodes of individuals is not constant. In **Fig. 11**, we see that the execution time increased linearly as the number of individuals increased from 50 to 150. However, linear computational effort with respect to the number of individuals was not required when the number of individuals exceeded 150.



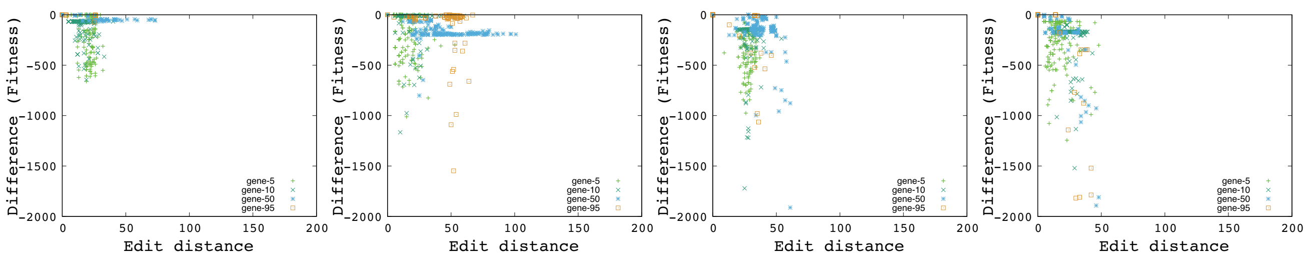
(a) $\gamma = 0$ (b) $\gamma = 5$ (c) $\gamma = 10$ (d) $\gamma = 15$

Fig. 7. Differences between the elite and the others (Function A).



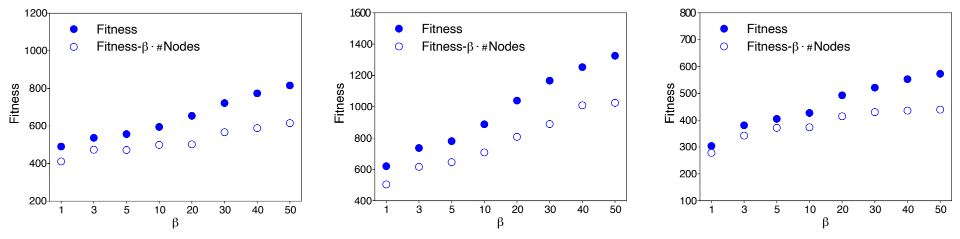
(a) $\gamma = 0$ (b) $\gamma = 5$ (c) $\gamma = 10$ (d) $\gamma = 15$

Fig. 8. Differences between the elite and the others (Function B).



(a) $\gamma = 0$ (b) $\gamma = 5$ (c) $\gamma = 10$ (d) $\gamma = 15$

Fig. 9. Differences between the elite and the others (Function C).



(a) Function A (b) Function B (c) Function C

Fig. 10. Influence of β on performance.

The proposed method uses CUDA to evaluate the similarity between individuals to provide acceleration; we set the number of threads at $|V|^2/2$, where $|V|$ is the number of individuals. Therefore, the proposed method can adaptively reconstruct subpopulations without a substantial increase in computational effort. These results demonstrate the effectiveness of the proposed method.

Finally, we investigated the difference in the solution quality of the proposed, N, and F methods, as described in Section 4.5. Thus, we evaluated the success rate of subpopulation reconstructions of the three methods. In Step P11 in Section 2, we describe an error procedure in the case where the subpopulation reconstruction fails. These methods use Newman clustering to adaptively di-

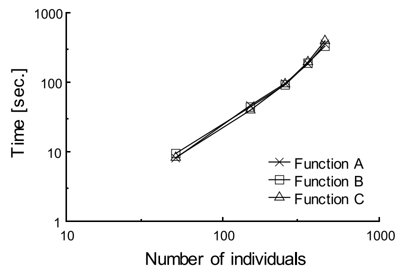


Fig. 11. Scalability in terms of the number of individuals.

Table 1. Success rate and average of $|P^m|$.

		Average of $|P^m|$	
Method	Success rate [%]	Best trial	Worst trial
Proposed	100	44.5	65.7
N-method	30	–	–
F-method	100	56.1	65.2

vide the population $\mathscr{P}$ into subpopulations P^m based on the features of individuals. If the value of each H_{ij} is similar, the number of communities extracted by Newman clustering may be 1. **Table 1** presents the success rate of subpopulation reconstruction over 20 trials. From the table, it can be seen that the proposed and F-methods can reconstruct subpopulations with a 100% success rate, but the N-method failed several times to reconstruct subpopulations. This result shows that the weighted network with only the difference in the node size cannot create subpopulations, whereas the proposed method using multiple features succeeded in creating subpopulations and exhibited a better performance than the F-method. We hypothesized that incorporating the multiple structure information of individuals in the migration strategy is important for enhancing the combination of partial solutions and leads to performance improvement. These results support our hypothesis.

5. Conclusion

To improve the subpopulation models in GP, it is essential to obtain a search strategy that has a good balance between local search and genetic diversity. We attempted to enhance subpopulation models by adaptively changing subpopulations according to the features of individuals to improve local search, where individuals can be represented by a tree structure in GP. In this paper, we introduced a novel similarity H to reconstruct subpopulations based on the similarity of fitness values and also node size between individuals, and proposed a novel subpopulation model (SoS-GP). The proposed method generates a weighted network of individuals according to the proposed measure H, and creates subpopulations using a network clustering technique. In this case, considerable computational effort is required to evaluate the similarity H. We proposed a CUDA-based model to shorten this time-consuming task. Moreover, we adopted a mutation in the proposed method. We hypothesized that mutation can enhance the generation of various individuals with different structures, and can maintain the balance between local search and genetic diversity by cooperating with the SoS model.

In well-known benchmark problems widely adopted in studies in the literature, we confirmed that the proposed subpopulation model offers a significant performance advantage over the comparison method. We also analyzed the behavior of the proposed method and demonstrated its validity.

The experimental results indicated that the performance of the proposed method and F-method is high, and confirmed that performance improvement can be achieved by incorporating multiple features and reconstructing subpopulations. Although in this paper we showed only the results for three problems, we achieved similar results in other problems. Our immediate future work is to evaluate the proposed method using various other benchmark problems. Moreover, other subpopulation models such as ALPS require appropriate boundary parameters; however, in the proposed method, some parameters like the number of mutations γ and crossover rate are fixed as the first step. We will investigate methods for adaptively controlling these parameters, and compare the proposed method with them. The proposed method uses fitness and node size to evaluate the similarity between individuals for simplicity. However, various features exist for estimating the similarity between individuals. Therefore, future work includes combining these features to create a weighted network with the objective of further performance improvement. Additionally, we will attempt to develop a more effective parallel method using CUDA to reduce computational effort.

Acknowledgements
This work was supported by JSPS KAKENHI Grant Number JP19H02032 and JP20K11982.

References:

[1] K. A. De Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," Ph.D. thesis, University of Michigan, 1975.

[2] B. L. Miller and M. J. Shaw, "Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimization," Proc. of 1996 IEEE Int. Conf. on Evolutionary Computation (ICEC'96), pp. 786-791, 1996.

[3] J. Hu, E. D. Goodman, and K. Seo, "Continuous Hierarchical Fair Competition Model for Sustainable Innovation in Genetic Programming," R. Riolo and B. Worzel (Eds.), "Genetic Programming Theory and Practice," pp. 81-98, Kluwer Academic Publishers, 2003.

[4] J. Hu, E. D. Goodman, K. Seo, and M. Pei, "Adaptive Hierarchical Fair Competition (AHFC) Model for Parallel Evolutionary Algorithms," Proc. of the Genetic and Evolutionary Computation Conf. 2002 (GECCO 2002), pp. 772-779, 2002.

[5] G. S. Hornby, "ALPS: the age-layered population structure for reducing the problem of premature convergence," Proc. of the 8th Annual Conf. on Genetic and Evolutionary Computation (GECCO'06), pp. 815-822, 2006.

[6] G. S. Hornby, "Steady-state ALPS for real-valued problems," Proc. of the 11th Annual Conf. on Genetic and Evolutionary Computation (GECCO'09), pp. 795-802, 2009.

[7] J. M. Luna, J. R. Romero, C. Romero, and S. Ventura, "Discovering Subgroups by Means of Genetic Programming," Proc. of the

16th European Conf. on Genetic Programming (EuroGP 2013), pp. 121-132, 2013.

[8] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," Swarm and Evolutionary Computation, Vol.1, Issue 2, pp. 61-70, 2011.

[9] D. Andre and J. R. Koza, "A Parallel Implementation of Genetic Programming That Achieves Super-Linear Performance," Proc. of the Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'96), Vol.III, pp. 1163-1174, 1996.

[10] W. F. Punch, "How Effective Are Multiple Populations in Genetic Programming," Genetic Programming 1998: Proc. of the 3rd Annual Conf., pp. 308-313, 1998.

[11] M. Harwerth, "Experiments on Islands," Proc. of the 14th European Conf. on Genetic Programming (EuroGP 2011), pp. 239-249, 2011.

[12] S. Harding and W. Banzhaf, "Fast Genetic Programming and Artificial Developmental Systems on GPUs," Proc. of the 21st Int. Symp. on High Performance Computing Systems and Applications (HPCS'07), doi: 10.1109/HPCS.2007.17, 2007.

[13] S. Harding and W. Banzhaf, "Genetic programming on GPUs for image processing," Int. J. of High Performance System Architecture, Vol.1, Issue 4, pp. 231-240, 2008.

[14] I. Arnaldo, K. Veeramachaneni, and U.-M. O'Reilly, "Flash: A GP-GPU Ensemble Learning System for Handling Large Datasets," Proc. of the 17th European Conf. on Genetic Programming (EuroGP 2014), pp. 13-24, 2014.

[15] W. B. Langdon and M. Harman, "Genetically Improved CUDA C++ Software," Proc. of the 17th European Conf. on Genetic Programming (EuroGP 2014), pp. 87-99, 2014.

[16] W. B. Langdon, "A Many Threaded CUDA Interpreter for Genetic Programming," Proc. of the 13th European Conf. on Genetic Programming (EuroGP 2010), pp. 146-158, 2010.

[17] K. Ono and Y. Hanada, "A CUDA-based self-adaptive subpopulation model in genetic programming: CuSASGP," Proc. of the 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 1543-1550, 2015.

[18] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," Proc. of the National Academy of Science of the United States of America, Vol.99, No.12, pp. 7821-7826, 2002.

[19] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," Physical Review E, Vol.70, Issue 6, doi: 10.1103/PhysRevE.70.066111, 2004.

[20] K. Yanai and H. Iba, "Estimation of distribution programming based on Bayesian network," Proc. of the 2003 Congress on Evolutionary Computation (CEC'03), Vol.3, pp. 1618-1625, 2003.

[21] W. B. Langdon and R. Poli, "Foundations of Genetic Programming," Springer, 2002.

[22] W. B. Langdon and R. Poli, "Why Ants are Hard," Genetic Programming 1998: Proc. of the 3rd Annual Conf., pp. 193-201, 1998.

Name:
Keiko Ono

Affiliation:
Doshisha University

Address:
1-3 Tatara Miyakodani, Kyotanbe, Kyoto 610-0394, Japan

Brief Biographical History:
2010- Assistant Professor, Ryukoku University
2014- Associate Professor, Ryukoku University
2020- Associate Professor, Doshisha University

Main Works:
• K. Ono, Y. Hanada, M. Kumano, and M. Kimura, "Enhancing Island Model Genetic Programming by Controlling Frequent Trees," J. of Artificial Intelligence and Soft Computing Research, Vol.9, No.1, pp. 51-65, 2019.

Membership in Academic Societies:
• The Institute of Electrical and Electronics Engineers (IEEE)
• Information Processing Society of Japan (IPSJ)
• The Japanese Society for Evolutionary Computation (JPNSEC)

Name:
Yoshiko Hanada

Affiliation:
Kansai University

Address:
3-3-35 Yamate-cho, Suita, Osaka 564-8680, Japan

Brief Biographical History:
2008- Assistant Professor, Kansai University
2016- Associate Professor, Kansai University

Main Works:
• K. Matsumura, Y. Hanada, and K. Ono, "Probabilistic Model-Based Multistep Crossover Considering Dependency Between Nodes in Tree Optimization," R. Lee (Ed.), "Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing," pp. 187-200, Springer, 2017.

Membership in Academic Societies:
• The Institute of Electrical and Electronics Engineers (IEEE)
• Information Processing Society of Japan (IPSJ)
• The Institute of Energy Economics, Japan (IEEJ)