Search Method of Number of Trees for Genetic Programming with Multiple Trees

Takashi Ito
Dept. of Industrial and Systems Engineering
Aoyama Gakuin University
Kanagawa, Japan
ito@ise.aoyama.ac.jp

2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM) | 978-1-6654-2318-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/IMCOM51814.2021.9377427

***Abstract*—Genetic programming (GP), which is an evolutionary computational method, is known to be effective for agent problems because individuals are represented by a tree structure. As an extension method, GP with control nodes (GP_{CN}) has been proposed. Because one individual has multiple tree structures, GP_{CN} can efficiently evolve and obtain highly readable behavioral rules. However, the number of trees suitable for each problem has to be manually adjusted in advance and cannot be easily applied various problems. In the previous study, a method for automatically determining the number of trees have proposed. However, because the method of the previous study changes the fitness function and uses a special population, it cannot be combined with the extension methods to improve the evolution performance. In this study, a method for searching for the appropriate number of trees using three islands is proposed. The proposed method divides the population into three islands, but because the genetic operations and the fitness function of each island are not changed, it can be combined with the existing extension methods. In the experiments, they are compared these using two benchmark problems.**

Keywords—Genetic Approach, Genetic Programming, Autonomous Agent, Multiple Trees

I. INTRODUCTION

Genetic programming (GP) [1-2] is an evolutionary computation that imitates the mechanism of biological evolution. GP extends genes of the genetic algorithm (GA) [3] to represent a tree structure, and is used to obtain the agent behavior rules and function optimization problems. Because GP is optimized during the evolution process by mating various individuals, it can be applied to problems in which a large number of sample data cannot be obtained. In addition, the behavioral rules and functions obtained are represented in a tree structure and are therefore highly readable and the contents can be easily confirmed.

Marino applied GP to the strategy generation of a real strategy game [4]. This study examined whether strategies obtained by GP are useful for the human learning process. In addition, Li et al. used multi-objective GP to generate rules for crowd behavior modeling [5]. In this study, rule generation was interpreted as a symbolic problem, and modeling rules were generated through the evolution of GP. In addition, La Cava et al. conducted research on multidimensional GP, which is an extension of GP, and proposed a crossover using semantics to improve the performance [6].

GP with control nodes (GP_{CN}), which has a multi-tree structure, has been proposed as an extension method of GP [7-8]. With GP_{CN}, because an individual is represented by multiple trees, complicated rules can be expressed more efficiently than with GP, and its effectiveness in agent problems has been shown. However, it was deemed necessary to manually change the number of trees in an individual because the suitable number of trees differs for each problem. A method for automatically determining the number of trees was previously proposed [9-10]. Using this method, individuals with various numbers of trees were generated in the population, and the numbers of trees were determined through natural selection. In addition, a fitness function and a genetic operation were changed to achieve this. However, it is difficult to determine the suitable number of trees for a problem, and this method cannot uniquely do so. In addition, the approach does not consider any other extensions because it specializes in determining the number of trees.

Therefore, in this study, we propose a method for searching the number of trees using three islands to adjust the number of trees during the evolution process. In our proposed method use to search for the number of trees, we did not change the conventional fitness function or genetic operations, and several individuals were involved with different numbers of trees on the three islands. In addition, the number of trees was changed by executing a change in the search target when the evolution was stagnant. The three islands are called *island with a small number of trees*, *island with a medium number of trees*, and *island with a large number of trees*. In addition, the change in the search target is an operation that shifts the search range of the number of trees, and is executed when the fitness is not updated during a fixed term on an island with a medium number of trees. Because this method does not specialize in determining the number of trees, as with a conventional method, it is possible to obtain a suitable behavioral rule for a problem while adjusting the number of trees during the evolution process. As another strength of this method, it can be combined with conventional extension methods. The method in the previous study cannot be combined with extended methods to improve the evolution performance because it is significant changed in the fitness function and is differed in the number of trees of individuals in the population. Our proposed method divides the population into three islands and searches for the number of trees, but it can be easily combined with the extended methods because the population, genetic manipulation, and fitness function in each island are the same as GP. The effectiveness of this method was confirmed experimentally using a garbage-collection problem [11] and a Santa Fe Trail problem and compared with the conventional method.

II. GENETIC PROGRAMMING WITH CONTROL NODES (GP_{CN})

A. Algorithm of GP_{CN}

GP_{CN} is GP extended such that one individual has two or more tree structures [7-8]. An example of an individual of GP_{CN} is shown in Fig. 1. As shown in Fig. 1, an individual of GP_{CN} is comprised of multiple trees. Each tree is composed of an identification node represented by a square, a non-terminal

978-0-7381-0508-6/21/$31.00 ©2021 IEEE

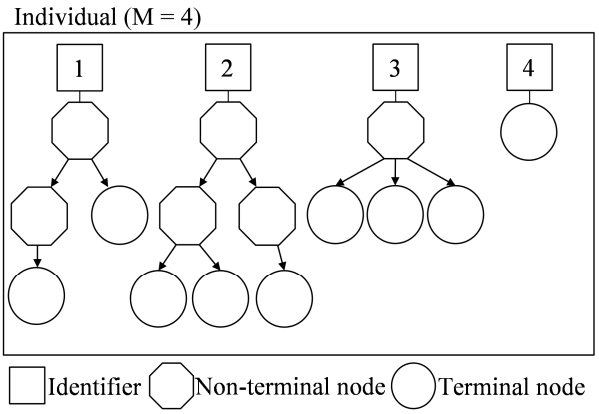

Fig. 1. Example of an individual in GP_{CN}.

node represented by an octagon, and a terminal node represented by a circle. The arrows indicate the connections between the nodes and the directions that the agent goes along with the nodes. The identification nodes are associated with a root node of each tree, the identification numbers indicating the reference order is set to they. In addition, a number P for controlling a number of references are set for each tree. The number of trees M of each individual and the number P in each tree must be determined in advance. In the first proposed GP_{CN}, the number of trees M was manually tuned for each problem by changing the number of trees during each simulation. The number P for each tree is calculated using (1). Here, M is the number of trees of an individual, and *Total_Steps* is the maximum number of actions of an agent set for each problem.

$$P = \frac{Total\_Steps}{M} \tag{1}$$

In agent problems, the non-terminal node represents a conditional branching by perceptual information, and the terminal node represents the behavior of an agent. The agent refers to a tree with the smallest identification number and acts according to its action rules. Then, when the number of actions of the agent in its tree exceeds the number P set in its tree, the agent changes the reference destination to a tree with the next identification number. The agent repeats the agent simulation until the accumulated number of actions reaches *Total_Steps*. If the number P has been changed, the number of references of the agent in a tree with the highest identification number may reach the number P in its tree before the accumulated number of actions of the agent reaches *Total_Steps*. In this case, the agent changes the reference to the tree with the smallest identification number and continues the agent simulation.

An evolutionary algorithm in GP_{CN} is similar to GP. First, a preset number of individuals is generated and an initial population is then formed by the generated individuals. Second, each individual in the population is evaluated through an agent simulation, and the fitness of each individual is calculated. It is then determined whether the maximum number of generations has been reached, and if the maximum number of generations has been reached, the processing ends. If the conditions are not met, a next-generation population will be generated. To generate next-generation populations, elite conservation, individual selection, and genetic operations are executed. In a genetic manipulation, GP_{CN} individuals have multiple tree structures, and thus one tree is selected and executed. After generating the next generation population, GP_{CN} returns to the evaluation of the population.

B. Method for automatically determining the number of trees

In the method for automatically determining the number of trees described in a previous study [9-10], the evaluation function and genetic operations have been significantly changed. First, the evaluation function was altered as shown in (2). Here, E is the value of the objective function set for each problem, *Total_Steps* is the maximum number of actions of the agent for each problem, M is the number of trees in an individual, $M_{singlenode}$ is the number of trees with only a terminal node in an individual, and α is the weight of the trees with only a terminal node.

$$Fitness = E + (Total\_Steps - M) - \alpha \times \frac{M_{single\,node}}{M} \tag{2}$$

This evaluation function consists of three terms, and second and third terms have been added. The second term gives superiority or inferiority to the number of trees, and is introduced to determine as small a number of trees as possible. In addition, the third term is introduced to improve the problem in which the evolution is hindered by trees with only terminal nodes.

In the change in genetic operations, a mutation is altered, and an operation to increase or decrease the number of trees is executed during this operation. There are two types of operations to increase the number of trees: *an operation copying trees* and *an operation adding random trees*. In addition, there are two types of operations to reduce the number of trees: *an operation deleting trees at random* and *an operation deleting trees based on semantics*. One of these four types of operations is executed with a fixed probability in one mutation.

III. GP_{CN} SEARCHING THE NUMBER OF TREES DURING EVOLUTION PROCESS (GP_{CNS})

A. Algorithm of GP_{CNS}

As a feature of this method, the evaluation function and genetic operations are not changed, unlike with a conventional method. Because the evaluation function and genetic operations in the conventional method were significantly changed, the approach specializes in determining the number of trees. Thus, it cannot determine the number of trees and obtain suitable behavioral rules for the problem at the same time. In addition, because individuals with various numbers of trees were included in one population, there were problems in which *the number of trees is not uniquely determined* and *a method proposed in previous studies to improve the performance of GP_{CN} cannot be applied*. GP_{CNS} using the three islands can be combined to improve the performance of GP_{CN} because it evolves individuals with different numbers of trees on each of the three islands. Although there are three types of islands in which the number of trees can be searched concurrently, it is also possible to search a range equivalent to that of the conventional method through an operation changing the search target. The three islands are called $island_S$, $island_M$, and $island_L$. $Island_S$ is composed of individuals with N-1 trees, $island_M$ is composed of individuals with N trees, and $island_L$ is composed of individuals with N+1 trees.

An algorithm of this method is shown in Fig. 2. First, an initial population is generated, and individuals are evaluated on each island. If the number of generations is smaller than a set maximum number of generations Ng, it is then judged whether a change in the search target should be executed. Here, if the fitness of a best individual does not change for N_c generations on island$_M$, the search target is changed. Because a change in the search target is an operation for changing the population, the selections of individuals and genetic operations are not executed during the generation in which they are executed, and the process returns to a fitness evaluation. The selections of individuals and genetic operation are executed only when a change in the search target is not executed, and the process again returns to an evaluation of the fitness. Because the three islands are independent of each other, individuals of different islands do not intersect through genetic operations.

B. Change in the search target

With this method, change the default search target, if the fitness of the best individual in island$_M$ does not change for N_c generations, a change in the search target is executed.

The algorithm for changing the search target is shown in Fig. 3. In Fig. 3, *Fitness*$_S$ and *Fitness*$_L$ are the fitness of the best individual of island$_S$ and island$_L$, respectively, N_s is the number of individuals to be saved or moved, and *Max_Number* is the maximum number of individuals in each island. First, the fitness of the best individual in island$_S$ is compared with that in island$_M$, and if the fitness of island$_S$ is greater than or equal to the fitness of island$_L$, the search moves to a smaller range. Otherwise, it moves to a larger range. When the smaller range is selected, the operations are executed in order from island$_L$ (left side in Fig. 3). In the opposite case, the operations are executed in order from island$_S$ (right side in Fig. 3). On the first island of each operation, original individuals on the island are outside the search range. To avoid completely deleting the obtained behavioral rules, the top N_s individuals on its island are saved to a sleep space. If individuals with the same number of trees have already been saved, they will be overwritten. Next, we

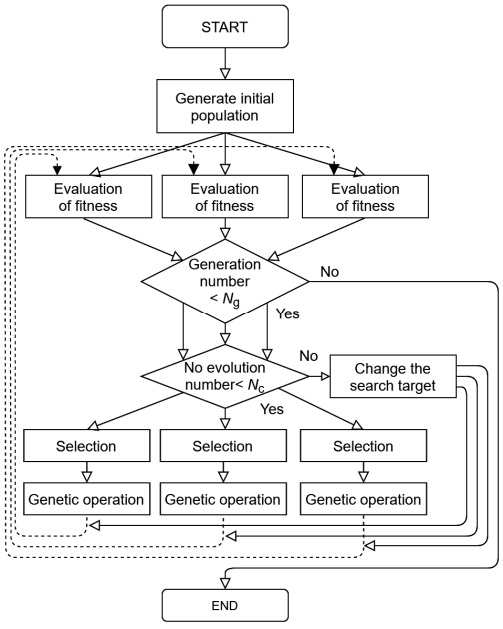

Fig. 2. Flowchart of GP$_{CNS}$.

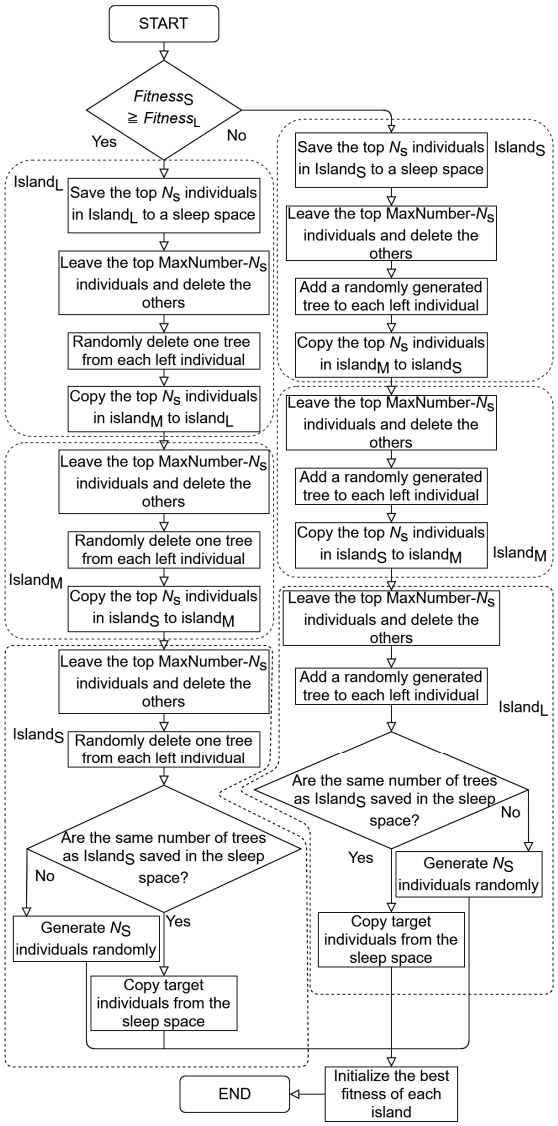

Fig. 3. Flowchart of the change in search target.

leave only the top *Max_Number*-N_s individuals on the current island, and delete the others. Then, to match the number of trees in the remaining individuals with the number of changed trees, a tree in each individual is deleted or added. Finally, the top N_s individuals are copied from the next island operation. However, because there is no following island in the third operation, if there are individuals with the same number of trees in the sleep space, they will be copied. Otherwise, N_s individuals will be generated randomly.

IV. Experiments

A. Benchmark problems

Two benchmark problems were used in this study. The first benchmark problem is the garbage-collection problem [11]. The object of an agent in this problem is to bring all trash in a field to a garbage-collection location with limited energy. An example field of the garbage-collection problem is shown in Fig. 4. The field of this problem is a two-dimensional lattice plane of the size 13 × 13, and the outermost cell is a wall that the agent cannot enter. In the field, there is one agent, 10 trash items, and 1 garbage collection location. The agent and trash are placed in random cells, and

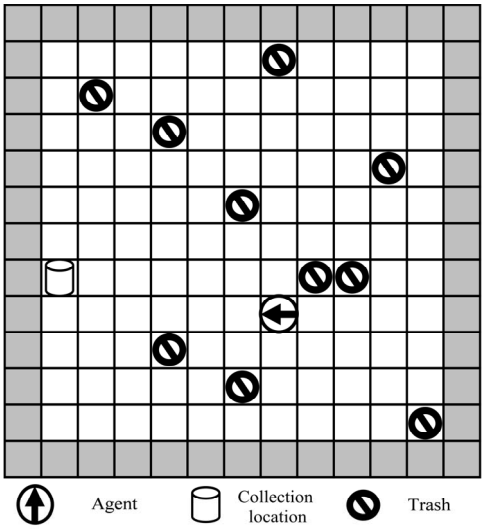

Fig. 4. Example of the field of the garbage-collection problem.

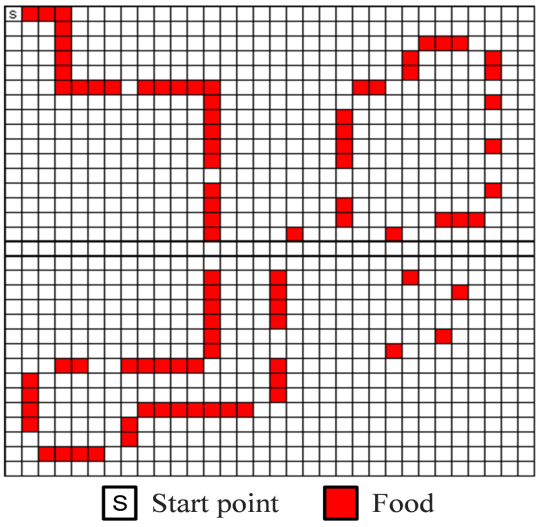

Fig. 5. Field of Santa Fe Trail problem.

for the agent, a direction is also set randomly. The location of the garbage collection location is fixed. The agent acts using behavioral rules that consist of conditional branches and the behaviors shown in Table I. The trash is collected when the agent reaches a cell where its trash is located. When the agent reaches a cell of the garbage collection location, the held trash items are discarded, and the number of held trash items in it is set to zero. In addition, the number of trashes that the agent can hold at the same time is limited to two. Ten environments are created in this problem, and the fitness is the total number of trash discarded within 250 steps in each environment. Therefore, the maximum fitness is 100 because 10 trash items are placed in each environment.

The second problem is the Santa Fe Trail problem. With this problem, the objective is to picks up all food items lined in a field. The field of the Santa Fe Trail problem is shown in Fig. 5. The field comprises a two-dimensional lattice plane with a cell size of 32 × 32; a total of 89 food items are placed in the designated cells. The agent acts using behavioral rules that consist of conditional branches and behaviors, as shown in Table II, and can also pick up food by reaching the cell where it exists. Unlike the garbage-collection problem, the starting position of the agent is fixed at the upper left. The fitness of this problem is the total number of food items picked up during 400 steps.

TABLE I. FUNCTIONS OF NON-TERMINAL AND TERMINAL NODES FOR THE GARBAGE-COLLECTION PROBLEM

kind	functions (number of edges)
0	Check the distance from the agent to the garbage-collection location (3)
0	How many pieces of trash the agent has (3)
0	Check the direction of the agent to the garbage-collection location (8)
0	Check the direction of the agent to the nearest trash (9)
0	Check the direction of the agent to the second nearest trash (9)
1	Move forward (0)
1	Turn right (0)
1	Turn left (0)
1	Stay (0)

TABLE II. FUNCTIONS OF NON-TERMINAL AND TERMINAL NODES FOR THE SANTA FE TRAIL PROBLEM

kind	functions (number of edges)
0	if there is food ahead (2)
0	act X; then Y (2)
0	act X, then Y; then Z (3)
1	move forward (0)
1	turn right (0)
1	turn left (0)

B. Parameters

With the garbage-collection problem, the population size is 300, the maximum number of generations N_g is 1,000, and the maximum depth of the trees is 6. In GP_{CN}, the number of trees in an individual M is 10, and the number of actions P is 25. In GP_{CNS}, the number of individuals on one island is set to 100, the number of trees on $island_M$ is set to 5, the interval N_C for executing the change of the search target is 20 generations, and the number of individuals to be saved or moved N_S is 50. Other parameter values used in the experiment are listed in Table III.

With the Santa Fe Trail problem, the population size is 2,000, the maximum number of generations N_g is 1,000, and the maximum depth of the trees is 6. In GP_{CN}, the number of trees in an individual M is 2, and the number of actions P is 200. In GP_{CNS}, the number of individuals on one island is set to 600, the number of trees on $island_M$ is set to 5, the interval N_C for executing the change in the search target is 20 generations, and the number of individuals to be saved or moved N_S is 300. Other parameter values used in the experiment are listed in Table III.

TABLE III. COMMON PARAMETER VALUES

	Values
Probability of crossover	0.8
Probability of mutation	0.05
Probability of mutation tree	0.1
Probability of inversion	0.2
Tournament size	1
Elite number	6

C. Performance Evaluations

The graph in Fig. 6 shows the change in fitness during 1,000 generations for the garbage-collection problem. We plotted the average fitness values obtained by the best individual in each simulation through 30 simulations for genetic programming (GP), GP with control node (GP_{CN}), GP_{CN} using a method to automatically determine the number of trees (GP_{CN_IMi}), and GP_{CN} for searching the number of trees during the evolution process (GP_{CNS}). In addition, GP_{CNS} shows a lower fitness value than GP_{CN_IMi}, which is a conventional method for automatically determining the number of trees; however, because GP_{CNS} can improve its performance and combine with other extension methods, it can actually show a higher fitness value than GP_{CN_IMi}. Looking at the inclination of the evolution, GP_{CN_IMi} is stagnant in approximately 600 generations, whereas GP_{CNS} evolves with an upward slope even during 1,000 generations. Therefore, the search for the number of trees in GP_{CNS} is considered to be continuously executed without being completed early.

The graph of Fig. 7 shows the change in fitness during 1,000 generations in the Santa Fe Trail problem. We plotted the average fitness values obtained by the best individual in each simulation through 30 simulations for genetic programming (GP), GP with control node (GP_{CN}), GP_{CN} using the method to automatically determine the number of trees (GP_{CN_IMi}), and GP_{CN} searching the number of trees during the evolution process (GP_{CNS}). From Fig. 7, unlike the garbage-collection problem, GP_{CNS} shows a higher fitness value than GP, GP_{CN}, and GP_{CN_IMi} in the Santa Fe Trail problem, and the best fitness value in GP_{CNS} is 88.2 out of a maximum fitness of 89. In addition, for GP_{CNS}, the evolution continues up to the 1000th generation unlike GP_{CN} and GP_{CN_IMi}.

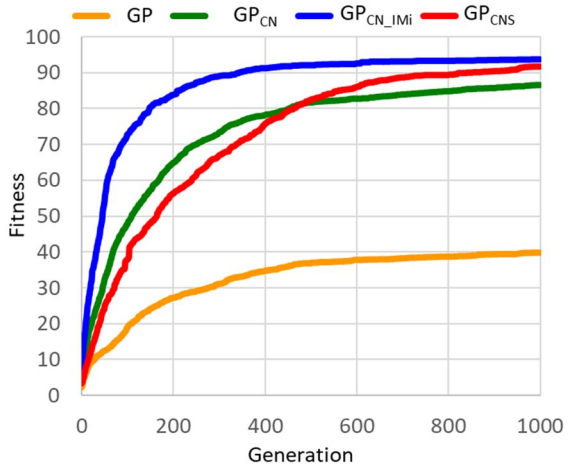

Fig. 6. Change in fitness obtained for 1,000 generations in the garbage-collection problem.

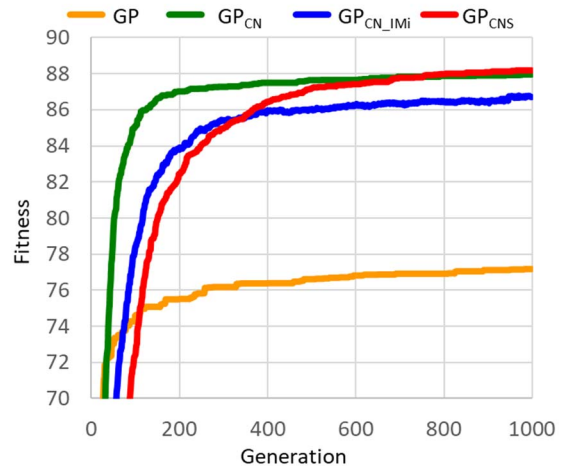

Fig. 7. Change in fitness obtained for 1,000 generations in the Santa Fe Trail problem.

Table IV shows the frequency of the number of trees *M* obtained by the best individual during each trial in GP_{CNS} and GP_{CN_IMi}. In addition, the top-three frequencies of the number of trees obtained for each method are indicated by color. For the garbage-collection problem, the number of trees of the best individual of GP_{CNS} are concentrated within 2 and 10 trees. The simplest number of trees is 2 because the garbage-collection problem requires a behavioral rule for picking up trash and a behavioral rule for dumping it. In addition, 10 trees is the optimum number when the number of trees is manually adjusted. Therefore, we considered that GP_{CNS} can obtain a suitable number of trees for this problem. In the Santa Fe Trail problem, the number of trees obtained by the best individual of GP_{CNS} is distributed within the range of 2-6. Because a small number of trees can be obtained without changing the fitness function, we considered the GP_{CNS} to achieve a sufficient performance as a method for adjusting the number of trees. The number of trees obtained by the best individual in GP_{CN_IMi} falls within range of 2-4 for both problems. However, this result was forcibly obtained by changing the fitness function, and we believe that GP_{CN_IMi} can not sufficiently search the various numbers of trees because evolution is stagnant around 400 generations from Fig. 7.

Based on the performance results of the best individual, the search process as indicated from the slope of the graph, and the number of trees obtained by the best individual, the proposed method was shown to be effective for adjusting the

TABLE IV. FREQUENCY OF THE NUMBER OF TREES M OBTAINED BY THE BEST INDIVIDUAL IN EACH TRIAL IN GP_{CNS} AND GP_{CN_IMi}

Number of trees	Obtained number of *M*			
	Garbage-collection problem		SFT problem	
	GP_{CNS}	GP_{CN_IMi}	GP_{CNS}	GP_{CN_IMi}
2	6	13	20	15
3	0	7	6	18
4	0	5	6	8
5	1	1	9	2
6	1	1	6	3
7	2	1	1	3
8	0	2	1	0
9	4	0	1	0
10	11	0	0	1
11	3	0	0	0
12	1	0	0	0
13	1	0	0	0

number of trees.

V. Conclusion

In this paper, to solve a problem in which a suitable number of trees for a target problem must be adjusted in advance for GP with multiple trees, we proposed a method for adjusting the number of trees during the evolution process. Our proposed method does not aim to uniquely determine the suitable number of trees for the target problem, but aims to automatically adjust the number during the process of evolution by setting the approximate number of trees. This method showed the same or better performance than the method described in a previous study. In addition, we confirmed that the number of trees when applying this method was continuously adjusted up to 1,000 generations from the slope of the graph. Therefore, the method proposed in this study is considered to be an effective approaches for adjusting the number of individual trees for GP with multiple trees. In addition, because the method can be easily combined with various extension approaches, a further improvement in performance can be expected.

In our future research, we will consider a method for automatically adjusting the number of processes on each tree.

References

[1] JR. Koza, Genetic programming: on the programming of computers by means of natural selection, MA: MIT Press. Cambridge, 1992.

[2] H. Iba, A primer of genetic programming, Japan: Tokyo University Press, 2002 (in Japanese).

[3] H. Iba, Genetic algorithm, Japan: Igaku Shuppan, 2002 (in Japanese).

[4] J. RH. Marino, "Learning Strategies for Real-Time Strategy Games with Genetic Programming," Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 15, no. 1, pp. 219–220, 2019.

[5] D. Li and J. Zhong, "Dimensionally Aware Multi-Objective Genetic Programming for Automatic Crowd Behavior Modeling," ACM Transactions on Modeling and Computer Simulation (TOMACS), 30(3), pp. 1–24, 2020.

[6] W. La Cava and J. H. Moore, "Learning feature spaces for regression with genetic programming," Genetic Programming and Evolvable Machines, 20, pp. 433–467, 2020.

[7] T. Minesaki, H. Ueda, and K. Takahashi, "Island model genetic programming based on frequent trees," The Conference Program of the 2009 (60th) Chugoku-branch Joint Convention of Institutes of Electrical and Information Engineers, p. 546, 2009 (in Japanese).

[8] T. Ito, K. Takahashi, M. Inaba, "Extension of genetic programming with multiple trees for agent learning," JOURNAL OF COMPUTERS, vol. 11 (4), pp. 329–340, 2016.

[9] T. Ito, K. Takahashi, and M. Inaba, "Obtaining the Number of Tree in the Evolution for Genetic Programming with Multiple Trees," The 5th IIAE International Conference on Intelligent Systems and Image Processing 2017 (ICISIP 2017), pp. 425–432, 2017.

[10] T. Ito, K. Takahashi, and M. Inaba, "The methods to determine automatically the number of trees in GP with multiple trees," Journal of Japan Society for Fuzzy Theory and Intelligent Informatics, vol. 30, no. 3, pp. 571–580, 2018 (in Japanese).

[11] S. Eto, S. Mabu, K. Hirasawa, T. Huruzuki, "Genetic network programming with control nodes," 2007 IEEE Congress on Evolutionary Computation, pp. 1023–1028, 2007 (in Japanese).