Obtaining the Number of Tree in the Evolution for Genetic Programming with Multiple Trees

Takashi Ito*a, Kenichi Takahashi*b, and Michimasa Inaba*a

*aGraduate School of Information Sciences, Hiroshima City University, 3-4-1 Ozukahigashi, Asaminami-ku, Hiroshima 731-3194, Japan
*bCorresponding Author: ito@cm.info.hiroshima-cu.ac.jp

Abstract

This paper proposes a method for automatic determination of the number of trees in genetic programming with control nodes (GP_{CN}). GP_{CN} is genetic programming with multiple trees. An individual in GP_{CN} comprises multiple trees, and each tree has a number $P$ that indicates the number of repetitive actions based on the tree. In previous work, a method for inheriting the number $P$ has been proposed to determine the number $P$ in evolution. In this work, a method to determine the number of trees $M$ in evolution is proposed. First, the fitness is changed to an advantageous fitness for individuals with small $M$. Second, a semantics for an agent's problem is proposed. Third, mutation of the individual is proposed to generate individuals with various $M$ for searching a suitable $M$.

Keywords: Genetic Approach, Genetic Programming, Autonomous Agent, Multiple Trees, Semantics.

1. Introduction

In recent years, autonomous agents like a vacuum cleaning robot that thinks and acts by oneself are popular. Artificial intelligence, which has gotten attention from the people, aims at modeling human intelligence on a computer. Agent learning is one of active fields in artificial intelligence. For agent learning, evolutionary learning, which imitates the mechanism of biological evolution, is known to be effective. In this study, we use genetic programming (GP)\textsuperscript{1,2}, which is one of evolutionary learning. The individual of GP is comprised of a single tree structure representing agent behavior. The leaf nodes correspond to agent actions, and the other nodes correspond to branches depending on perceptual information. In GP, excellent individuals are selected by tournament selection, etc., in the population. The population in the next generation is generated by using selected individuals. In this way, the population in GP is updated to next generations.

In order to improve the performance of GP, various methods have been proposed: the method using the Semantic Aware Crossover (SAC) that uses the similarity of subtrees to avoid destructing of tree structures\textsuperscript{3}, the method in which semantics is used for selecting operation to keep diversity\textsuperscript{3}, and a mutation of subtrees and a crossover using semantic backpropagation\textsuperscript{4}. In order to prevent the bloat phenomenon, GP removing a crossover operator and using multi-objective methods that add a distance between individuals to the objective function\textsuperscript{5}, and multi-step search crossover based on neighbor search considering the dependency relationship of not only parent nodes but also child nodes\textsuperscript{6} are proposed.

In GP, the depth of generated trees tends to be deep to obtain complex action rules because one individual has only one tree. Deep trees with complex action rules have shortcomings that there is a possibility that excellent action rules might be destroyed by a single genetic operation. In our study, genetic programming with control nodes (GP_{CN}) has been proposed to improve the readability\textsuperscript{7}. An individual in GP_{CN} comprises multiple trees. The tree has the following two numbers: the identification number that indicates the order in which an agent refers to a tree, and the number $P$ that indicates the number of repetition by which an agent carries out the action designated by the leaf node in a tree. In GP_{CN}, an individual has multiple trees, and each tree represents different an action rule. Therefore, GP_{CN} can solve the problem that GP produces deep trees, because in GP_{CN}, complex action rules represented by a single tree in GP can be divided into multiple simple trees.
In addition, a method for inheriting the number $P$ which automatically decides the number $P$ of each tree in the evolution is proposed. Moreover, the fitness function is changed in order to solve a problem that trees with a single node impede the evolution in $\text{GP}_{\text{CN}}$. In this study, we propose a method for automatic determination of the number of trees $M$, which must be decided in advance in $\text{GP}_{\text{CN}}$. For automatic determination of the number of trees $M$, we change the fitness function. Individuals of $\text{GP}_{\text{CN}}$ can have multiple trees. However, we want to make the number of trees $M$ small in order to be evolved quickly. Thus, we change the fitness function to it advantageous to individuals with the small number of trees $M$. In addition, we propose the mutation of individuals for generating individuals with various numbers of trees $M$ in the evolution. The mutation of individuals comprises two operations for increasing trees and two operations for decreasing trees. Two operations for increasing trees are the method copying trees in an individual and the method adding trees generated at random. Two operations for decreasing trees are the method deleting trees selected at random and the method deleting trees based on semantics.

We apply the proposed methods to a garbage collection problem to compare the performance with that of the previous methods.

2. Genetic Programming with Control Nodes ($\text{GP}_{\text{CN}}$)

An example of an individual of $\text{GP}$ with control nodes ($\text{GP}_{\text{CN}}$) that has been extended to have multiple trees is depicted in Fig. 1. Individuals of $\text{GP}_{\text{CN}}$ comprise multiple trees which correspond to action rules for an agent. The tree has the following two numbers: the identification number that indicates the order in which an agent refers to a tree, and the number $P$ that indicates the number of repetition by which an agent carries out the action designated by the leaf node in a tree. The number of trees $M$ in one individual and the number $P$ in each tree are supposed to be determined in advance for $\text{GP}_{\text{CN}}$. Each tree comprises a non-terminal node and a terminal node as $\text{GP}$. The non-terminal node corresponds to a branch by the perceptual information, and the terminal node corresponds to an action that an agent can execute. An agent refers to a tree with the smallest number and carries out an action according to the tree. When the number of actions that an agent carries out using the tree exceeds a designated number $P$, the agent refers to a tree with the next number. After the tree with the largest number is processed, the agent refers to the tree with the smallest number. At that time, the number of actions that an agent carries out using the tree in each tree is initialized to 0. Until the accumulated number of actions of trees which an agent refers to becomes Total Steps, the agent repeats receiving perceptual information from the environment and then carrying out an action. When the total number of actions in trees that the agent carries out so far reaches Total Steps, agent simulation for the individual stops, and then its fitness value is evaluated. Trees not referred in agent simulation are removed from individuals.

The algorithm of $\text{GP}_{\text{CN}}$ is shown in Fig. 2. First, $\text{GP}_{\text{CN}}$ generates the initial population of individuals. Then, using agent simulation, it evaluates the fitness of each individual that has been generated. If the generation number is less than $N_s$, then it performs the selection of individuals and genetic operations. It then generates a population of individuals in the next generation. Here, $N_s$ is the maximum number of generations. Although the $\text{GP}_{\text{CN}}$ individuals have multiple trees, the fitness is evaluated for each individual, not for each tree. Details of the genetic operations for $\text{GP}_{\text{CN}}$ are described as following.

(a) Crossover

Crossover is the operation that exchanges subtrees in trees of two parent individuals. First, two trees are selected from two parent individuals respectively, and nodes are selected at random for crossover from all nodes of each tree. Second, subtrees whose root nodes are the selected nodes are exchanged. However, no crossover is executed when a tree consists of only a root node.

(b) Mutation

We use two kinds of mutation: a mutation-tree and a mutation-node. The mutation-tree is an operation that randomly selects one node from all nodes in a tree of a parent individual and then replaces the subtree subsequent
to the selected node with a randomly generated subtree. The mutation-node is an operation that changes the content of the selected node after selecting a node in a tree of a parent individual. In mutation-node, if the selected node is a non-terminal (terminal) node, then the node content is replaced with another content of a non-terminal (terminal) node. When any content of a non-terminal node is changed, the edge number might change. If the number of edges of a new content becomes smaller, then the extra edges and the succeeding subtrees are removed. However, if the number of edges becomes larger, then randomly generated subtrees are connected to the increased edges.

(c) Inversion

The inversion operation selects only a non-terminal node at random from all nodes of a tree in a selected individual and selects at random two child nodes of the node. Then it exchanges the subtrees that have the two child nodes as the root nodes. However, if the root node of the selected tree is terminal node, the inversion operation is not applied to the selected tree.

In mutation and inversion that generate one child individual from one parent individual, the generated child individual inherits the value of $P$ from the parent individual, because we think that the tree structure of its child individual resembles the parent individual. In crossover that generates two child individuals from two parent individuals, each child individual inherits the value of $P$ from the parent individual, which resembles the generated child individual in tree structure.

3. Extention Methods in Previous Study

3.1 The method for inheriting the number $P$

In GP$_{CN}$, we must determine the number of trees $M$ and the number $P$ in advance. In order to determine the number $P$ automatically in evolution, the method for inheriting the number $P$ has been proposed. The number $P$ in excellent individuals is inherited to child individuals by this method when crossover is executed. Thus, the number $P$ in excellent individuals is spread to the population in the next generation.

[1] Select a tree $A$ from a parent individual at random.
[2] Select a tree $B$ from another parent individual at random.
[4] Compare the fitness values of the two parent individuals, and then execute the following.

If the fitness value of the parent individual with tree $A$ (tree $B$) is higher than the parent individual with tree $B$ (tree $A$), the value of $P$ of the parent individual with tree $A$ (tree $B$) is inherited to trees $X$ and $Y$.

3.2 The fitness function to eliminate trees with a single terminal node

In GP, an individual is comprised of a single tree, and the individuals with a tree comprised of a single terminal node (i.e. the tree that is comprised of a single agent action) is eliminated in evolution. For example, in a garbage collection problem that an agent picks up all pieces of trash scattered in the field and carries them to a garbage dump site, the agent that can only move forward or turn right can’t carry all trash to a garbage dump site. Those individuals are eliminated in evolution, because the fitness values of them become low.

However, in GP$_{CN}$, an individual is comprised of multiple trees. Individuals with trees comprised of a single terminal node are functionally same as individuals comprised of a single tree with multiple agent actions. Thus, in the garbage collection problem, these individuals with trees consisting only of one terminal node can survive, because the agent who refers to multiple trees can move and carry some trash to a garbage dump site. On the other hand, individuals with complex action rules comprised of non-terminal nodes and terminal nodes tend to be eliminated in evolution, because these individuals include
many unnecessary actions of the agent at early generations, which makes their fitness values low.

In order to solve this problem, we added an evaluating value for eliminating trees with a single terminal node to the previous fitness function, and changed to the fitness function for eliminating trees with a single terminal node.

Let \( E \) denote an evaluating value of an agent, for example the number of trash collected by individual in the garbage collection problem. Then, the changed fitness of the individual \( i \) is calculated as shown below.

\[
Fitness(i) = \max \{ 0, E(i) - \alpha \times \frac{M_{\text{single node}}(i)}{M(i)} \} \quad (1)
\]

In (1), the value of \( \alpha \) is a weight of the trees with a single node, \( M(i) \) is the number of trees in the individual \( i \), and \( M_{\text{single node}}(i) \) is the number of trees with a single node in the individual \( i \).

4. Proposed Methods

4.1 The fitness function considering the number of trees

The number of trees \( M \) is one of a characteristic parameter in GP\(_{\text{CN}}\). If the number of trees \( M \) becomes too large, the improvement of fitness becomes slow. Thus, for deciding the small number of trees \( M \), we add an evaluation term for considering the number of trees to the fitness function in addition to the evaluation term to eliminate trees with a single terminal node. The proposed fitness function of individual \( i \) is as shown below.

\[
Fitness(i) = E(i) + \left( \text{TotalSteps} - M(i) \right) - \alpha \times \frac{M_{\text{single node}}(i)}{M(i)} \quad (2)
\]

Here, Total Steps is the maximum number of actions of an agent set for each problem, which is used to prevent the calculated fitness from becoming less than 0.

4.2 Semantics for the agent’s problem

In GP, semantics of term used in order to evaluate the similarity of tree. The similarity of trees based on semantics represents the one between functions of trees not structures of trees. For example in a symbolic regression problem, values are calculated from multiple terminal nodes to the root node according to a tree structure representing an equation. And the way to let a change of values in the calculating process be semantics is proposed. However, in agent’s problems, that way is not easily approach to semantics of an individual because the output is determined in the way of passing from the root node to one of multiple terminal nodes according to the environment. Semantics of individuals for agent’s problems is proposed by E. Galvan-Lopez et al\(^4\). In the semantics, every time the agent moves to a different place, directions of the agent are recorded. They regard its tracks as semantics. However, this semantics can not represent the behavior in which an agent stays in the same place.

In this study, we generate various environments in advance, and define semantics as the agent’s actions obtained by simulation in those environments. The proposed semantic is described in detail in Section 5.

4.3 The mutation of individuals

To obtain automatically the optimal number of trees \( M \) in the evolution, we need to search various numbers of trees \( M \). However, with the proposed fitness function, we can not search various the numbers of trees \( M \). Thus, we proposed the mutation of individuals.

The algorithm of GP\(_{\text{CN}}\) using the mutation of individuals (GP\(_{\text{CN,M}}\)) is shown in Fig. 3. First, GP\(_{\text{CN,M}}\) generates the initial population of individuals. Then, using agent simulation, it evaluates the fitness of each individual that has been generated. If the generation number is less...
than \( N_g \), then it performs the selection of individuals. Next, it checks whether the number of trees of all selected individuals is the same or not. If so, the mutation of individuals is performed with a probability for the mutation of individuals. Otherwise, it is not performed. Then, it performs genetic operations. It then generates a population of individuals in the next generation. Here, \( N_g \) is the maximum number of generations. Although the \( \text{GP}_{\text{CN}} \) individuals have multiple trees, the fitness is evaluated for each individual, not for each tree.

In the mutation of individuals, first, we select the operation for increasing the number of trees or the operation for decreasing the number of trees with a probability. The operation for increasing the number of trees is selected with a probability of the increasing operation. Otherwise, the operation for decreasing the number of trees is selected. In the mutation of individuals, first, we select either the operation for increasing the number of trees or the operation for decreasing the number of trees with a pre-determined probability. However, if the number of trees is two, the operation for increasing the number of trees is always performed. If the operation for increasing the number of trees is selected, we select the operation copying trees with a probability of copying trees, otherwise select the operation adding the random trees. If the operation for decreasing the number of trees is selected, we select the operation deleting trees at random with a probability of the random deletion, otherwise select the operation deleting similar trees in an individual based on semantics. Finally, the selected operation is executed. The two operations for increasing the number of trees and the two operations for decreasing the number of trees are described in detail below.

(a) The operation copying trees
The operation copying trees is an operation increasing the number of trees by copying a tree in an individual. First, one tree is randomly chosen in the individual selected by tournament selection, and the chosen tree is copied. Next, the copied tree is inserted to the selected individual. The place where the copied tree is inserted is chosen at random. When the copied tree is not inserted into the last place, the tree at the inserted place and the successive trees were shifted behind. Finally, the number of trees \( M \) is increased by 1. In addition, by repeating the above process twice, we increase the number of trees by 2 to expand a search space.

(b) The operation adding random trees
The operation adding random trees is an operation increasing the number of trees by adding trees generated at random. First, a random tree is generated. Next, the random tree is inserted to the individual selected by tournament selection. The place where the random tree is inserted is chosen at random. When the random tree is not inserted into the last place, the tree at the inserted place and the successive trees were shifted behind. Finally, the number of trees \( M \) is increased by 1. In addition, by repeating the above process twice, we increase the number of trees by 2 to expand a search space.

(c) The operation deleting trees at random
The operation deleting trees at random is an operation deleting a tree chosen at random in an individual. First, one tree is randomly chosen in the individual selected by tournament selection, and the chosen tree is deleted. When the place of the deleted tree is not last, we shift trees placed after the deleted trees ahead. Finally, the number of trees \( M \) is decreased by 1.

(d) The operation deleting trees based on semantics
The operation deleting trees based on semantics is an operation deleting similar trees in an individual based on semantics.

Let \( M \) denote the number of trees, and let tree \( i \) denote tree with the identification number \( i \). Then the algorithm of the operation deleting trees based on semantics is following.

1. Initialize \( i \) to 1.
2. With semantics, calculate similarity between the tree \( i \) and the tree \( M \).
3. If similarity is greater than a threshold, delete the tree \( M \) and decrease \( M \) by 1.
4. If \( M = 2 \), pass Step 10.
5. With semantics, calculate similarity between the tree \( i \) and the tree \( i+1 \).
6. If the value of similarity is less than threshold, pass to Step 10.
7. Delete the tree \( i+1 \), and decrease \( M \) by 1.
8. When \( i+1 < M \), shift trees which are placed after the deleted tree ahead.
9. Increase \( i \) by 1, and if \( i+1 \leq M \), return to Step 4.
10. Finish.

There is a possibility that multiple trees are deleted, and a possibility that no tree is deleted according to the similarity of two trees.

5. Experiments

5.1 The garbage collection problem
The objective of a garbage collection problem is that an agent picks up all pieces of trash scattered in the field
and carries them to a garbage dump site. An example of the field of the garbage collection problem is depicted in Fig. 4. The field comprises a two-dimensional lattice plane of the size 11x11 cells, and the outermost cells are walls. The garbage collection problem has one agent, ten pieces of trash, and one dump site on the field. The agent can move forward, turn left or right, or stay at each step. The agent can also pick up a piece of trash by reaching the cell where it exists and then can carry it to the dump site. The maximum number of pieces of trash that the agent can carry is assumed as two.

Table 1 presents the functions of non-terminal nodes and terminal nodes in the garbage collection problem. We have two kinds of nodes: 0 denotes non-terminal nodes (branch nodes), and 1 denotes terminal nodes (action nodes). We prepare 10 environments generated by placing the agent, trash, and the dump site in advance. Here, the agent and trash are placed at randomly selected cells.

We define $E_{GC}$ as the agent’s evaluation in the garbage collection problem. $E_{GC}$ is as a total of the number of trash carried to the dump sites in the 10 environments in 250 steps per environment. Let $N_i(j)$ denote the number of trash collected by individual $j$ in environment $i$ and $E_{GC}(j)$ denote the total number of trash collected by individual $j$. Then, $E_{GC}(j)$ is calculated as shown below.

$$E_{GC}(j)=\sum_{i=1}^{10} N_i(j)$$

We define semantics for the garbage collection problem as following. We generate 10 environments to calculate semantics in advance. In 10 environments, the agent’s states (i.e., the place, the direction, and the number of trash kept by the agent) and trash on field are changed at random. Proposed semantics is calculated after the evaluation of individuals. First, we apply the agent to 10 environments for calculating semantics. Let $a_i$ be the first action executed by the agent in the environment $i$, and let semantics of individuals be 10-dimensional column vector $S_i = (a_1, a_2, ..., a_{10})$. Because the individual of GP$_{CN}$ in this study has multiple trees, we calculate semantics of trees.

### 5.2 Parameters

In the garbage collection problem, the population size is 300, and the maximum number of generations is 1,000. The number of trees in an individual $M$ is chosen randomly between [2, 20], and the number $P$ is chosen randomly between [1, Total Steps/$M$]. We employ the grow method as the generating method of the initial population, and set the maximum depth of trees to 6. Common parameter values to each GP$_{CN}$ used in the experiment are listed in Table 2, and parameter values for the mutation of individuals used in the experiment are listed in Table 3. In addition, we use the fitness function considering the number of trees.

### 5.3 Performance Evaluations

Let $E_{GC}$ denote the total number of trash carried to the
dump sites, and we depict the change of $E_{GC}$ obtained in 1,000 generations in Fig. 5. We plot the average values of $E_{GC}$ of the individual with the best fitness in population obtained through 30 simulation runs.

In GP$_{CN}$, when population is generated at the beginning, individuals have a variety number of trees. In this study, we execute the recombination of the tree structure in individuals and the convergence of the number of trees simultaneously in the process of the evolution. In Fig. 5, even in such the disadvantageous situation, GP$_{CN}$ shows a higher value of $E_{GC}$ than GP. GP$_{CN,1}$ using the method for inheriting the number $P$ showed a higher value of $E_{GC}$ than GP$_{CN}$ because each tree obtained a suitable value for $P$. GP$_{CN,Mi}$ using the mutation of individuals did not show better $E_{GC}$ than GP$_{CN}$ but almost the same. The authors believe that this because the suitable number $P$ for each tree was not obtained. As expected, GP$_{CN,Mi}$ introducing the mutation of individuals and the method for inheriting the number $P$ showed the highest value of $E_{GC}$.

We define the converged number in the simulation $i$ as the number of trees of individuals with the largest number of survivors at 1000 generations in the simulation $i$ of GP$_{CN,Mi}$. We list the number of simulations that converged to each number of trees to Table 4. We regard the number of trees with the largest converged number as the suitable number to the number of trees for the garbage collection problem. Thus, the number 2 of trees with the largest converged number in Table 4 is the suitable number for the number of trees for the garbage collection problem. In addition, the number of simulations obtaining the highest $E_{GC}$ in 13 simulations converging to the number 2 was 11, and $E_{GC}$ of each simulation was 100. Because each of 11 simulations obtained different value for the number $P$, we conducted experiments to determine which value for $P$ is suitable to. As the results of the experiment, GP$_{CN}$ with $P_1 = 24$ and $P_2 = 18$ shows a highest value of $E_{GC}$.

6. Conclusions

In this paper, to automatically determine the number of trees in GP$_{CN}$, we proposed the fitness function considering the number of trees and the mutation of individuals. In the garbage collection problem, GP$_{CN,Mi}$ introducing the mutation of individuals which is the proposed method and the method for inheriting the number $P$ in the previous work showed the best performance. In addition, we obtained the suitable value for the number of trees and the number $P$ for the garbage collection problem through this experiment. As our future works, we will compare GP$_{CN}$ using obtained values for the number of trees and GP$_{CN}$ using values for $P$ set manually in the previous work.

<table>
<thead>
<tr>
<th>Number of trees</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of converged simulations</td>
<td>13</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Acknowledgment

This research was in part supported by a Hiroshima City University Grant for Special Academic Research (General).

References

(2) Hitoshi Iba : “Genetic algorithm,” Igaku Shuppan, Japan, 2002