Extension of Genetic Programming with Multiple Trees for Agent Learning

Takashi Ito*, Kenichi Takahashi, Michimasa Inaba
Graduate School of Information Sciences Hiroshima City University, Hiroshima, Japan.

* Corresponding author. Email: ito@cm.info.hiroshima-cu.ac.jp
Manuscript submitted July 10, 2015; accepted October 8, 2015.
doi: 10.17706/jcp.11.4.329-340

Abstract: This paper proposes an extension of genetic programming (GP) with multiple trees. In order to improve the performance, GP with control node (GPCN) and its three kinds of modification have been proposed. In GPCN, an individual consists of several trees which have the number P of executions. In previous work, the two kinds of modification, the conditional probability and the cross-cultural island model are employed. This paper proposes two methods: the new island model that combines the conditional probability with two islands in the cross-cultural island model and a method exchanges multiple trees in an individual in a suitable order. Experiments are conducted to show the performance in the garbage collection problem and the Santa Fe Trail problem.

Key words: Autonomous agent, conditional probability, genetic programming, island model.

1. Introduction

In the field of artificial intelligence, which aims at modeling human intelligence, many researchers have studied search algorithms to obtain agent decisions and action rules to reach a goal. Reinforcement learning and evolutionary learning are representative means to learn agent behavior. Evolutionary methods are known to be able to obtain optimum rules for agent action in a broad search space. Among evolutionary methods, genetic algorithm (GA) [1], genetic programming (GP) [2], [3] and genetic network programming (GNP) [4] have been investigated eagerly and widely.

In GP, the population in the next generation is produced by generating child individuals from parent individuals with high fitness values in the current generation. Each individual is comprised of a single tree structure in GP. The leaf nodes correspond to agent actions, and the other nodes correspond to branches depending on perceptual information. Individuals in the next generation are generated using genetic operations, namely crossover, mutation, and inversion. Some of individuals with high fitness values are called elites and are inherited to the next generation.

In order to improve the performance of GP, various methods have been proposed: the method generating individuals in the next generation by joining fragments of the tree structure that randomly been sampled from several parent individuals [5], the method extracting useful tree structures from individuals called frequent trees [6], [7] that are subtrees that frequently appear in the population, the island model that combines those frequent trees [8], the method using the Semantic Aware Crossover (SAC) [9] that uses the similarity of subtrees to avoid destructive of tree structures, and the method in which semantics are used for select operation to keep diversity [10].

In GP, the depth of generated trees tends to be deep to obtain complex action rules because one individual has only one tree. However, as the depth of a tree becomes deeper, the readability of the tree becomes lower, and also there is a possibility excellent action rules are destroyed by a genetic operation. In our study, genetic programming with control nodes (GP_{CN}) [11] has been proposed to improve the readability. An individual in GP_{CN} comprises several trees. The tree has 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. GP_{CN} can be a solution to the problem that GP produces deep trees. Because each tree in GPCN corresponds mutually-independent action rules. However, the increase of fitness becomes slow because the individual has multiple trees.

Therefore, we have introduced the conditional probability into GP_{CN} to improve performance of GP_{CN}; we call the method GP_{CN_CP} [12]. In GP_{CN_CP}, individuals in the next generation are generated by using either genetic operations or conditional probability tables, where the conditional probability tables are updated with conditional probabilities between connected nodes that are extracted individuals with high fitness values. Thereby, GP_{CN_CP} can maintain the diversity of individuals and inherit the structures of excellent individuals to the next generation with a high probability, where the excellent individuals are supposed to have obtained appropriate rules for agent behavior in the environment.

Additionally, we have employed the cross-cultural island model [13] to promote diversity for overcoming local optima problem and for improving the fitness because GP_{CN} has a shortcoming that it tends to be trapped by local optima; we call the method GP_{CN_IL} [14]. In GP_{CN_IL}, the population is divided into two islands of individuals: one island emphasizes maintenance of the diversity of individuals; the other emphasizes improving fitness. We call the former the diversity-oriented island. The latter is called the performance-oriented island. The island model can be expected to prevent the solution from reaching a local optimum because GP_{CN_IL} can emphasize two points such as maintaining diversity and improving the fitness.

In this paper, we propose a new island model GP_{CN_ILCP} [15] that combines the conditional probability with two islands in the cross-cultural island model as an extension of the two methods that are shown effective in previous work, namely the conditional probability and the cross-cultural island model. In GP_{CN_ILCP}, we introduce the conditional probability into the performance-oriented island to accelerate the improvement of the fitness in the performance-oriented island. The conditional probability tables are made by individuals with high fitness values in the performance-oriented island. On the other hand, in the diversity-oriented island, individuals in the next generation are generated by using the conditional probability tables considering the diversity; the conditional probability tables considering the diversity are made by converting the conditional probability table for the performance-oriented so that nodes with low probabilities in the performance-oriented conditional probability table could be chosen.

Furthermore, in this paper, we propose a new method that focuses on the reference order of multiple trees in an individual; we call the method GP_{CN_ET}. In GP_{CN}, an individual comprises several trees, and the order in which an agent refers to a tree is controlled by the identification number. However, the trees might not be in proper order. Therefore, in GP_{CN_ET}, a tree in an individual is exchanged with another tree in the individual, in order to rearrange trees in a suitable order. We select two trees from an individual, and then exchange the reference order of the selected trees.

We apply GP_{CN_ILCP} and GP_{CN_ET} to a garbage collection problem [16] and the Santa Fe Trail problem [2], [17] to compare the performance with that of the traditional methods GP, GP_{CN}, GP_{CN_CP}, and GP_{CN_IL}. We adopt the garbage collection problem and the Santa Fe Trail problem because these problems have been used to show the ability of GNP and GP in previous work. Although the symbolic regression problem exists

as another type of benchmark problem for GP and GNP, we chose the garbage collection problem and the Santa Fe Trail problem because the objective of this paper is to obtain rules for agent actions. The former is relatively easy, but the latter is difficult. Experimentally obtained results are presented to confirm the effectiveness of those methods.

2. Genetic Programming with Control Nodes (GP_{CN})

2.1. Outline of GP_{CN}

An example of an individual of GP with control nodes (GP_{CN}) [11] that has been extended to have multiple trees is depicted in Fig. 1. Individuals of GP_{CN} comprise several trees which correspond to action rules for an agent. The tree has 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 in one individual, i.e. the number of control nodes is denoted by *M* and is supposed to be determined in advance.

Additionally, each tree has its own number *P*. The number *P* is the maximum number of actions of an agent set for each problem, and the value of *Total Steps* is initially assigned to it. A non-terminal node corresponds to a branch by the perceptual information, and a 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.

The algorithm of GP_{CN} is the same as that of the traditional GP. First, GP_{CN} generates the initial population of individuals. Then, it evaluates the fitness of each individual that has been generated. If the condition to terminate processing is not met, then it performs reproduction of the population of individuals and genetic operations. It then generates a population of individuals in the next generation. Here, the condition to terminate processing is that the number of generation becomes the designated number of generations. Although the GP_{CN} individuals have several trees, the fitness is evaluated for each individual, not for each tree. Details of the genetic operations for GP_{CN} are described in the next subsection.

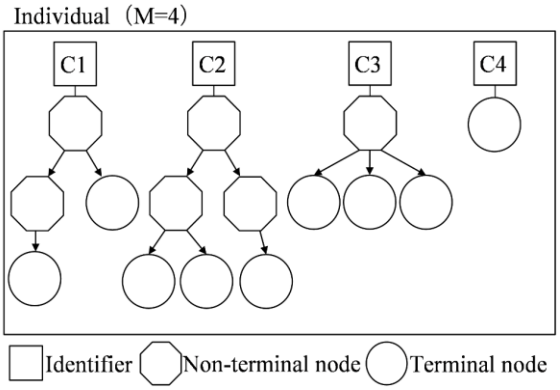


Fig. 1. An example of an individual in GP_{CN}.

2.2. Genetic Operations

Because one individual in GP_{CN} has several trees unlike normal GP [2], [3], for each genetic operation an individual is selected by tournament selection. Then one tree is selected at random from the selected individual. Each genetic operation is applied to the selected tree.

- **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.
- **Mutation:** We use mutation of two kinds: a mutation-tree and a mutation-node. A 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.
- **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.

3. GP_{CN} Using Conditional Probabilities (GP_{CN_CP})

In GP_{CN} using conditional probabilities (GP_{CN_CP}) [12], a problem that the improvement of fitness becomes slow because of multiple trees. Therefore, we have proposed to use the conditional probabilities and introduced the conditional probability into GP_{CN}. The algorithm of GP_{CN_CP} is the following.

1) Generate initial population of individuals.
2) Evaluate the fitness of the individuals.
3) Finish the evolution, if the number of generations becomes the maximum number of generations, where the maximum number of generations is the number given in advance. Otherwise execute 4.
4) Make the frequency tables from individuals with high fitness values.
5) Convert the frequency tables to the conditional probability tables.
6) Generate half of the maximum number of individuals with the conditional probability tables.
7) Generate the other half of the maximum number of individuals with genetic operations.
8) return to 2.

In GP_{CN_CP}, individuals in the next generation are generated by using conditional probabilities in addition to genetic operations. In using the conditional probabilities, the frequency tables and the conditional probability tables are made for each identification number of trees. Thereby, we can gain action rules considering roles of trees with identification numbers. First, frequency tables are made by counting the frequency of child nodes attached to branching edges for perceptual information of each nonterminal node in trees of which the numbers are the same over individuals. Individuals with high fitness values in the current generation are used to make frequency tables. The authors think that we can prevent rapid convergence by using (1), where in (1) frequency tables in the current generation is updated by inheritance frequency tables of the previous generation at a constant rate to the next generation. In (1), i is the identification number, $F_t(i)$ is the next frequency table for tree i, $F_{tp}(i)$ is the frequency table used in the

previous generation, and $F_{tc}(i)$ is the frequency table evaluated using only individuals of the current generation. We call α the inheritance probability, where $\alpha \in [0, 1]$.

$$F_t(i) = F_{tp}(i) \times \alpha + F_{tc}(i) \times (1 - \alpha) \quad (1)$$

We produce conditional probability tables from the frequency tables. In generating individuals using the conditional probabilities, first, the root node of a tree is chosen randomly based on the occurrence probability of each root node. Then, we determine child nodes by choosing nodes based on the conditional probability table of parent node. The decision of child nodes using conditional probabilities is repeated until a terminal node is selected for the child node or until the depth of child nodes reaches the maximum depth determined in advance. When the depth of a child node is the maximum depth, a terminal node is selected for the child node.

4. GP_{CN} Using the Island Model (GP_{CN_IL})

A shortcoming of evolutionary learning is that it tends to end the search with local optima in the process of evolution. We have applied the cross-cultural island model [13] to GP_{CN} in order to improve the shortcoming of evolutionary learning [14]. The purpose of the cross-cultural island model is to improve the fitness while maintaining the diversity of individuals. For that purpose, the cross-cultural island model divides individuals in two islands, namely the performance-oriented island and the diversity-oriented island, and searches for optimum solution.

First, we generate the initial population in the islands. Second, we evaluate the population of individuals and perform migration, which is one the features of the island model. Migration is an operation that moves a part of individuals of an island to another island. Migration is carried out in every generation. Individuals with low fitness values in the performance-oriented island are exchanged with those with high fitness values in the diversity- oriented island. The number of migrated individuals is designated in advance. Subsequently if the termination condition is not satisfied, then we perform selection and generate individuals in the next generation. The termination condition is the same as GP_{CN}; when the number of generations reaches the designated number, the algorithm stops.

In the performance-oriented island, we preserve elite individuals and generate individuals in the next generation with crossover. Although there are mutation and inversion as genetic operations, these operations are not used in the performance-oriented island because they may destroy good rules in excellent individuals obtained for agent behavior. Crossover used in the performance-oriented island is the depth-dependent crossover [18] instead of the usual crossover used in GP and GP_{CN}. Unlike the normal crossover, the depth-dependent crossover determines a depth in the selected tree at random and then selects a node for the crossover from nodes at that depth. Consequently, a destructive crossover is unlikely to occur. Individuals with high fitness values are preserved. In the diversity-oriented island, we generate individuals in the next generation by replacing some individuals with randomly generated individuals and individuals generated with genetic operations. In GP_{CN}, an individual has multiple trees. Thus, we replace a tree in an individual chosen randomly with a randomly generated tree.

5. Proposed Method

5.1. GP_{CN} Using the New Island Model that Combines the Conditional Probability with Two Islands

In this paper, we propose a new method [15] that combines the conditional probability with the island model described in previous work. In the performance-oriented island of the cross-cultural island model,

we preserve elite individuals and generate individuals in the next generation with crossover, because mutation and inversion may destroy good rules in excellent individuals obtained for agent behavior. However, without mutation and inversion, elite preservation and crossover can not generate new structured individuals with good rules since they only preserve good individuals and generate individuals with similar structures. Therefore, we introduce the conditional probability in addition to the two operations into the performance-oriented island. The method using the conditional probability matches the purpose of the performance-oriented island, because we can expect that individuals generated by the conditional probabilities made by individuals with high fitness values could have high fitness values.

Additionally, we introduce the conditional probability into the diversity-oriented island. We use individuals with high fitness values in the performance-oriented island to make the frequency tables. However, the conditional probabilities made in the same way as in the performance-oriented island may hinder the purpose of the diversity-oriented island; generated individuals might have low diversity, because the conditional probabilities tend to generate individuals with the similar structures. Therefore, we convert the conditional probability tables made in the performance-oriented island to the conditional probability tables considering diversity. Nodes with small probabilities in the performance-oriented island are included in individuals, but are hardly selected because of its low probability. In order to keep diversity, we make the conditional probability table considering diversity so that those nodes could have large probabilities.

Table 1. The Conditional Probability Table Calculated in the Performance-Oriented Island

Child node	The values corresponding to perceptual information	
	0	1
B	0.6	0
a	0.3	0
c	0.1	1.0
Total	1.0	1.0

Table 2. The Conditional Probability Table Converted by (2)

Child node	The values corresponding to perceptual information	
	0	1
B	0.4	0
a	0.7	0
c	0.9	1.0
Total	2.0	1.0

We explain how to convert the conditional probability tables by using Table 1. Table 1 is an example of conditional probabilities made for the performance-oriented island. First, we subtract the probability of each node from 1.0 by using (2), in order to convert low probabilities in Table 1 to high probabilities and in order to convert high probabilities in Table 1 to low probabilities for the conditional probability tables for the diversity-oriented island.

$$C_t'(i) = 1.0 - C_t(i) \tag{2}$$

In (2), $C_t'(i)$ is the converted conditional probability for node i, and $C_t(i)$ is the conditional probability made in the performance-oriented island for node i. When the probability of a node is 0 or 1.0, (2) is not applied to the node, because the authors consider that the nodes with probability 0 are not necessary and the nodes with probability 1.0 are suitable child nodes. Table 2 lists the converted conditional probabilities

calculated by (2). Then, we normalize values in Table 2 to make the conditional probability tables considering a diversity. In Table 3, we show an example of the normalized conditional probability table. In the diversity-oriented island, we use the conditional probability tables considering a diversity to generate individuals. We can generate individuals with high diversity because (2) restrains biased selection.

Table 3. The Normalized Conditional Probability Table Used for the Diversity-Oriented Island

Child node	The values corresponding to perceptual information	
	0	1
B	0.2	0
a	0.35	0
c	0.45	1.0
Total	1.0	1.0

5.2. GP_{CN} Exchanging Multiple Trees (GP_{CN_ET})

In GP_{CN}, one individual comprises several trees which correspond to rules, trees in an individual can have action rules independent of each other. Each tree has the number *P* which is the number of actions carried out by an agent and the identification number which control the order in which an agent refers to trees. The order in which an agent refers to trees does not change in the process of evolution. Thus, each tree evolves so as to be suitable for a role dependent on its order. However, the order of trees generated initially is not changed, which may require a lot of evolution time to get a suitable tree structure for a role dependent on its order, because structures of these trees may be different from roles dependent on the order in which an agent refers to a tree.

Therefore, in this study, we propose a new method GP_{CN_ET} that exchanges multiple trees in order to rearrange multiple trees of an individual in a suitable order. The algorithm of GP_{CN_ET} is the following.

1) BEGIN
2) Select a tree in an individual at random.
3) Select another tree in the individual at random.
4) If those trees are the same, return to 3.
5) Exchange those trees including numbers Ps assigned to them.
6) END.

In GP_{CN_ET}, we generate individuals in the next generation by using genetic operations and the exchange operation of multiple trees, where genetic operations are crossover, mutation, and inversion. The exchange of multiple trees occurs in every generation with the predetermined probability.

6. Experiments

6.1. Benchmark Problems

6.1.1. Garbage collection problem

The objective of the garbage collection problem [16] 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. 2. The field comprises a two-dimensional lattice plane of the size 11×11 cells, where the outermost cells are walls. We place one agent that can have two pieces of trash at most, ten pieces of trash, and one dump site on the field. The agent and trash are placed at random.

In the garbage collection problem, 5 non-terminal nodes and 4 terminal nodes can be used as the functions of the agent. In Table 4, we show the functions of non-terminal nodes and terminal nodes in the

garbage collection problem. Table 4 has nodes of two kinds: 0 denotes non-terminal nodes (branch nodes), and 1 denotes terminal nodes (action nodes).

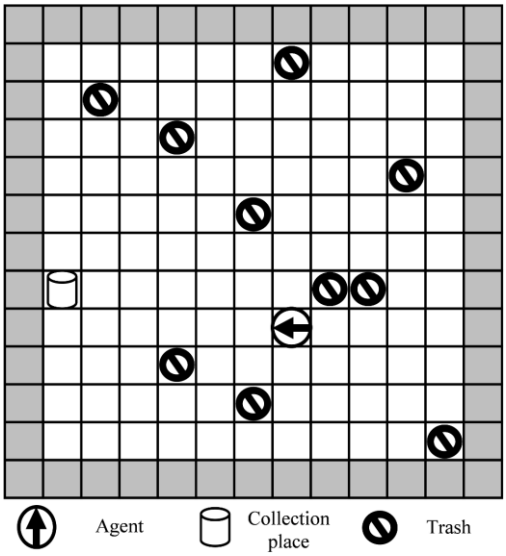


Fig. 2. An example of the field of a garbage collection problem.

Table 4. Functions of Non-Terminal Nodes and Terminal Nodes for the Garbage Collection Problem

kind	functions (the number of edges)
0	check the distance from the agent to the dump site (3)
0	how many pieces of trash the agent has (3)
0	check the direction of the agent to the dump site (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 (1)
1	turn right (1)
1	turn left (1)
1	stay (1)

We prepare 10 environments generated by placing the agent and trash at randomly selected cells in advance. We define the fitness as the number of total pieces of trash carried to the dump sites in the 10 environments in 250 steps per environment. Let N_i denote the number of collected pieces of trash in environment i. Then, the fitness value is calculated as shown below.

$$Fitness = \sum_{i=1}^{10} N_i \tag{3}$$

6.1.2. Santa Fe trail problem

The objective of the Santa Fe Trail problem [17] is that an agent picks up all pieces of food in the field. The field comprises a two-dimensional lattice plane of the size 32×32 cells. One agent and 89 pieces of food are placed in the designated cells.

In the Santa Fe Trail problem, 3 non-terminal nodes and 3 terminal nodes can be used as the functions of the agent. In Table 5, we show the functions of non-terminal nodes and terminal nodes in the Santa Fe Trail problem. Table 5 has nodes of two kinds: 0 denotes non-terminal nodes (branch nodes), and 1 denotes terminal nodes (action nodes).

We define the fitness as the total number of pieces of food picked up in 400 steps.

Table 5. Functions of Non-Terminal Nodes and Terminal Nodes for the Santa Fe Trail Problem

kind	functions (the 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 (1)
1	turn right (1)
1	turn left (1)

6.2. Parameters

In the garbage collection problem, the population size is 300, the maximum number of generations is 1000, the number of trees in an individual M is 10, and the number of actions P is 25. We employ the grow method as the generating method of the initial population, and set the maximum depth of trees to 6. Other parameter values used in the experiment are listed in Table 6.

Table 6. Parameters for the Garbage Collection Problem

	GP, GP_{CN}, GP_{CN_CP}, GP_{CN_ET}	GP_{CN_IL}, GP_{CN_ILCP}	
		Performance	Diversity
Probability of mutation	0.05	—	0.2
Probability of mutation tree	0.1	—	0.1
Probability of crossover	0.8	1.0	0.8
Probability of inversion	0.2	—	0.1
Probability of exchanging with random individual	—	—	0.1
Probability of exchange of plural trees	0.7	—	—
Tournament size	2	2	3
Elite number	1	1	—
Number of conditional probability	75	50	25
Probability of inheritance	0.0		
Migration size	—	100	

Table 7. Parameters for the Santa Fe Trail Problem

	GP, GP_{CN}, GP_{CN_CP}, GP_{CN_ET}	GP_{CN_IL}, GP_{CN_ILCP}	
		Performance	Diversity
Probability of mutation	0.05	—	0.05
Probability of mutation tree	0.1	—	0.11
Probability of crossover	0.77	1.0	0.98
Probability of inversion	0.2	—	0.2
Probability of exchanging with random individual	—	—	0.2
Probability of exchange of plural trees	0.15	—	—
Tournament size	2	2	3
Elite number	1	300	—
Number of conditional probability	200	175	25
Probability of inheritance	0.0		
Migration size	—	850	

In the Santa Fe Trail problem, the population size is 2000, the maximum number of generations is 1000, the number of trees in an individual M is 2, and the number of actions P is 200. We employ the ramped half-and-half method as the generating method of initial population, and set maximum depth of trees to 6. Other parameter values used in the experiment are listed in Table 7.

6.3. Performance Evaluation

The change of the fitness of the garbage collection problem obtained in 1000 generations is depicted in Fig. 3(a). We plot the average of the best fitness values obtained through 30 simulation runs obtained for genetic programming (GP), GP with control node (GP_{CN}), GP_{CN} using the conditional probability (GP_{CN_CP}), GP_{CN} with the cross-cultural island model (GP_{CN_IL}), and the two methods proposed in this paper, namely GP_{CN} using the new island model that combines the conditional probability with two islands (GP_{CN_ILCP}) and exchanging multiple trees (GP_{CN_ET}). We can confirm that using the conditional probability and the cross-cultural island model for GP_{CN} are effective as described in previous work, because GP_{CN_CP} and GP_{CN_IL} show better performance than GP and GP_{CN} in Fig. 3 (a). Consequently, GP_{CN_ILCP} improves performance by combining the conditional probability with the island model. We can see that exchanging multiple trees for GP_{CN} is effective, because GP_{CN_ET} shows the best performance in Fig. 3(a). However, from the point of evolution speed, GP_{CN_ET} is slower than GP_{CN_CP}, GP_{CN_IL}, and GP_{CN_ILCP}.

The change of the fitness of the Santa Fe Trail problem obtained in 1000 generations is depicted in Fig. 3(b). We plot the average of the best fitness values obtained through 50 simulation runs obtained for the traditional methods and the proposed methods. Similar to the garbage collection problem, we compare the performance of proposed methods, using Fig. 3(b). In the Santa Fe Trail problem, we can see that using the conditional probability and the cross-cultural island model for GP_{CN} are effective, because GP_{CN_CP} and GP_{CN_IL} show better performance than GP and GP_{CN} in Fig. 3(b). GP_{CN_ILCP} that combines the conditional probability with the island model shows better performance than GP_{CN_CP} and GP_{CN_IL}. We obtained similar results for GP_{CN_ILCP} in the Santa Fe Trail problem to those in the garbage collection problem. The performance of GP_{CN_ET} is lower than that of GP_{CN_IL} and as low as that of GP_{CN_CP}, because the number of trees in the Santa Fe Trail problem is 2 and exchanging trees doesn't take effect.

From the results of the two problems, although evolution of GP_{CN_ET} is slower than that of other extensions of GP_{CN}, we ascertain that GP_{CN_ET} is effective to overcoming local optima problem because evolution continues after 1000 generations.

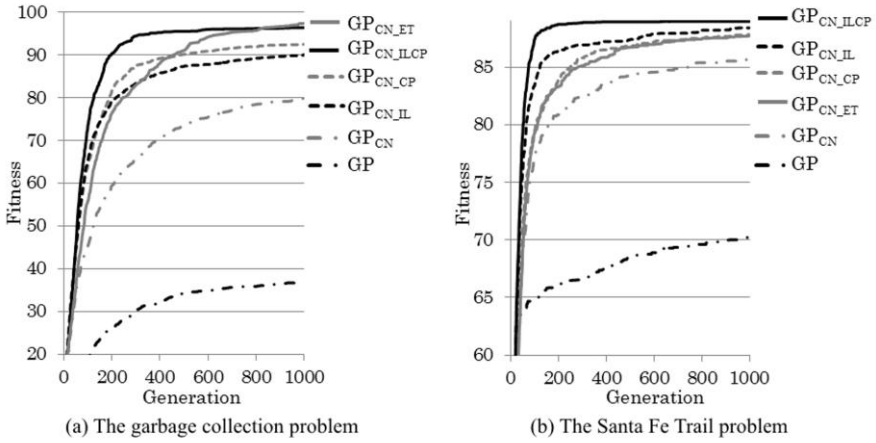


Fig. 3. Change of the fitness of two problems obtained in 1000 generations.

7. Conclusion

In this paper, we proposed two methods: GP_{CN} using a new island model (GP_{CN_ILCP}) that combines the

two methods that were shown to be effective in previous work and exchanging trees (GP_{CN_ET}). They were compared with traditional methods.

In the garbage collection problem, GP_{CN_ET} showed the best performance, and GP_{CN_ILCP} showed second best performance. In the Santa Fe Trail problem, the performance of GP_{CN_ILCP} showed the best performance, and the performance of GP_{CN_ET} was lower than that of GP_{CN_IL} and as low as that of GP_{CN_CP}.

From these results, we confirmed that combining the conditional probability into two islands in the cross-cultural island model is effective, because GP_{CN_ILCP} shows high fitness through two problems. Additionally, through experiments of two problems, we ascertained that GP_{CN_ET} is effective to overcoming local optima problem, because evolution has continued after 1000 generations.

Acknowledgment

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

References

[1] Iba, H. (2002). *Genetic algorithm*. Igaku Shuppan. Japan.

[2] Koza, J. R. (1992). *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MA: MIT Press. Cambridge.

[3] Iba, H. (2002). *A primer of Genetic Programming*. Tokyo University Press. Japan.

[4] Hirasawa, K., Okubo, M., Katagiri, H., Hu, J., & Murata, J. (2001). Comparison between genetic network programming and genetic programming using evolution of ant's behaviors. *IEEJ Transactions on Electronics, Information and System, 121(6)*, 1001-1009.

[5] Tanji, M., & Iba, H. (2010). A new gp recombination method using random tree sampling. *IEEJ Transactions on Electronics, Information and Systems, 130(5)*, 775-781.

[6] Ono, K., Hanada, Y., Shirakawa, K., Kumano, M., & Kimura, M. (2012). Depth-dependent crossover in genetic programming with frequent trees. *Proceedings of 2012 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 359-363).

[7] Ono, K., Hanada, Y., Kumano, M., & Kimura, M. (2013). Genetic programming for lighting control using frequent trees and depth information. *IEEJ Transactions on Electronics, Information and Systems, 133(11)*, 2044-2052.

[8] Ono, K., Hanada, Y., Kumano, M., & Kimura, M. (2013). Island model genetic programming based on frequent trees. *Proceedings of 2013 IEEE Congress on Evolutionary Computation* (pp. 2988-2995).

[9] Nguyen, Q. U., Nguyen, T. H., Nguyen, X. H., & O'Neill, M. (2010). Improving the generalisation ability of genetic programming with semantic similarity based crossover. *Proceedings of Genetic Programming 13th European Conference* (pp. 184-195).

[10] Galv´an-L´opez, E., Cody-Kenny, B., Trujillo, L., & Kattan, A. (2013). Using semantics in the selection mechanism in genetic programming: a simple method for promoting semantic diversity. *Proceedings of 2013 IEEE Congress on Evolutionary Computation* (pp. 2972-2979).

[11] Minesaki, T., Ueda, H., & Takahashi, K. (2009). Comparison experiment using genetic network programming. *Proceedings of the Conference Program of the 2009 (60th) Chugoku-branch Joint Convention of Institutes of Electrical and Information Engineers* (p. 546).

[12] Morioka, T., Ueda, H., & Takahashi, K. (2011). Efficient evolutionary learning of agent behavior by genetic programming using the conditional probabilities. *Proceedings of 12th International Symposium on Advanced Intelligent System* (pp. 342-345).

[13] Hara, Y., Kanagawa, A., Yamauchi, H., & Takahashi, H. (2006). *Improvement of efficiency of gp Using*

Cross-cultural Island Model (NLP2006-67) (IEICE Technical Report). pp. 11-16.

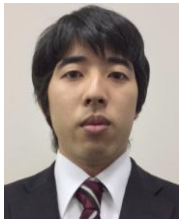
[14] Ito, T., Takahashi, K., & Inaba, M. (2014). Experiments assessing learning of agent behavior using genetic programming with multiple trees. *Proceedings of the 6th International Conference on Agents and Artificial Intelligence* (pp. 264-271).

[15] Ito, T., Takahashi, K., & Inaba, M. (2014). Extension of genetic programming specialized in agent learning. *Proceedings of Joint Agent Workshops & Symposium 2014* (pp. 253-256).

[16] Eto, S., Mabu, S., Hirasawa, K., & Huruzuki, T. (2007). Genetic network programming with control nodes. *Proceedings of 2007 IEEE Congress on Evolutionary Computation* (pp. 1023-1028).

[17] Mesot, B., Sanchez, E., Peña, C.-A., & Perez-Uribe, A. (2002). SOS++: Finding smart behaviors using learning and evolution. *Proceedings of the 8th International Conference on the Simulation and Synthesis of Living Systems* (pp. 264-273).

[18] Iwashita, M., & Iba, H. (2002). Parallel distributed gp with immigrants aging and depth-dependent crossover. *Transactions of Information Processing Society of Japan, 43(SIG10)*, 146-156.



Takashi Ito received the ME from Hiroshima City University in 2015. He is currently a doctoral student in Graduate School of Information Sciences at Hiroshima City University. His research interests include artificial intelligence and machine learning. He is a member of IEICEJ.

Kenichi Takahashi received his ME in information engineering from Nagoya Institute of Technology in 1979 and the doctor of engineering in information engineering from Nagoya University in 1986. Since 1994, he has been a professor of Hiroshima City University. His research interests include artificial intelligence, pattern information processing and machine learning. He is a member of IEEE, IEICEJ and IPSJ.

Michimasa Inaba received the PhD degree from Nagoya University in 2012. He is currently an assistant professor in Graduate School of Information Sciences at Hiroshima City University. His research interests include artificial intelligence, data mining and natural language processing. He is a member of IEEE, IEICEI, JSAI and IPSJ.