Evolutionary method of Genetic Network Programing Considering Breadth and Depth

Shinji Eto
Graduate School of Information, Production and Systems, Waseda University
Hibikino 2-7, Wakamatsu-ku, Kitakyushu
Fukuoka, 808-0135, Japan
eto@suou.waseda.jp

Shingo Mabu
Graduate School of Information, Production and Systems, Waseda University
Hibikino 2-7, Wakamatsu-ku, Kitakyushu
Fukuoka, 808-0135, Japan
mabu@asagi.waseda.jp

Kotaro Hirasawa
Graduate School of Information, Production and Systems, Waseda University
Hibikino 2-7, Wakamatsu-ku, Kitakyushu
Fukuoka, 808-0135, Japan
hirasawa@waseda.jp

ABSTRACT
Many methods of generating behavior sequences of agents by evolution have been reported. A new evolutionary computation method named Genetic Network Programming (GNP) has also been developed recently along with these trends. In this paper, a new method for evolving GNP considering Breadth and Depth is proposed. The performance of the proposed method is shown from simulations using garbage collector problem.

Categories and Subject Descriptors
I.2.8 [**Artificial intelligence**]: Ploblem Solving, Control Methods, and Search—*Graph and tree search strategies*

General Terms
Evolution performance

Keywords
evolutionary computation, genetic network programming, artificial intelligence

1. INTRODUCTION
Various research on the planning of agents has been done in the field of artificial intelligence and robotics[1]. These research discusses how an agent can achieve a given goal under a certain environment . When the agent is a living thing, this corresponds to the design of an artificial brain of the living thing.

Such an artificial brain has been designed by designers, so far. However, it is easily understood that the design of such an artificial brain becomes difficult as the purpose and environment of the agent systems become more complex. Therefore, the artificial brain of the agent system should be acquired by evolution rather than it is given by designers. Genetic Network Programming (GNP) has also been developed recently along with these trends.

A lot of research on the evolutionary design of agent systems has been reported. There have been used Genetic Algorithm (GA)[2], Genetic Programming (GP)[3], and Parallel Algorithm Discovery and Orchestration (PADO)[4] for the above purpose.

Most conventional methods determine the actions of agents mostly by using the current information on the environment. But, in this paper, the actions are created by Genetic Network Programming (GNP) [5, 6, 7] using only the necessary information on the environment.

GNP can generate programs evolutionally by connecting the nodes in GNP to form a directed graph. Genetic Algorithm (GA) uses the sequence of bits as the gene of an individual, while Genetic Programming (GP) uses a tree structure as the gene of an individual. Generally, it is recognized that the graph structure has higher expression capability compared with character sequences or tree structures. Moreover, it is expected that the graph structure of GNP can improve the performance by finding appropriate judgment nodes and processing nodes as needed in evolutionary processes.

Various evolutionary method of GNP could be studied, because GNP has a directed graph structure. The aim of this paper is to improve the performance of GNP by extending the evolutionary method of it. In conventional GNP, the activated node isn't compulsorily transferred to the start node. However, in the proposed method, plural start nodes are set and the activated node is properly transferred to one of the start nodes. It is found from the simulations that the performance of GNP could be improved by the proposed method.

2. OUTLINE OF GENETIC NETWORK PROGRAMMING
In this section, Genetic Network Programming (GNP) is explained briefly. Basically, GNP is an extension of GP in terms of gene structures. The original idea is based on the more general representation ability of directed graphs than that of trees.

The features of GNP are as follows.

- Only the necessary information is used for the judgment nodes in GNP unlike Finite Automata, so GNP can be applied to Partially Observable Markov Decision Processes, which leads to the wide spread use of GNP.

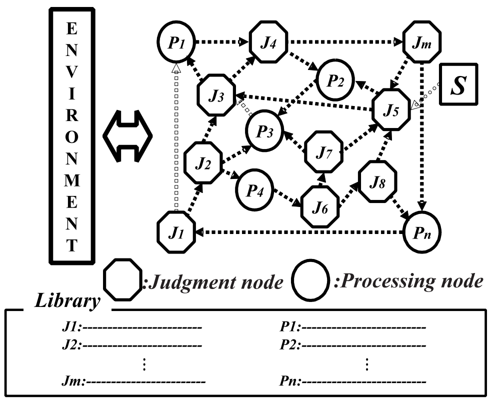

Figure 1: The basic structure of GNP

- The judgment nodes and processing nodes are repeatedly used in GNP, so the gene of it becomes compact, which causes the efficient evolution of GNP.

2.1 Basic Structure of GNP

The basic structure of GNP is shown in Fig.1. As shown in Fig.1, a directed graph structure is used in GNP to represent individuals. GNP is composed of plural nodes which are connected with each other like networks. These nodes are roughly classified into two kinds of nodes: *Judgment node* and *Processing node.*

Judgment nodes correspond nearly to elementary functions of GP and processing nodes correspond almost to terminal symbols of GP. Judgment nodes are the set of $J_1, J_2, \ldots, J_m$, which work as some kinds of judging functions. On the other hand, processing nodes are denoted by the set of $P_1, P_2, \ldots, P_n$, which work as some kinds of processing functions. The practical roles of these nodes are predefined and stored in the library by supervisors. Additionally, one specific node, start node S, is included in GNP. Start node indicates the start point of GNP, which corresponds to GP's root node. GP's elementary functions and terminal symbols are repeatedly used in a tree structure using Automatically Defined Functions (ADFs)[8]. In the same way, there are some judgment nodes and processing nodes used repeatedly in GNP as shown in Fig.1 without ADF. An ADF is a function (i.e., subroutine, subprogram, procedure) that is dynamically evolved during a run of genetic programming and which may be called by a calling program (or subprogram) that is concurrently being evolved. These judgment nodes and processing nodes are the essential elements of GNP. The number of these nodes may be determined as a result of evolution like GP. Actually, GNP can use this strategy, in other words, GNP can adopt evolving the directed graph with varying the number of nodes [9], but in this paper GNP evolves only the directed graph with the prefixed number of nodes. It would be better to say that GNP here evolves the genotype with the fixed number per each kind of nodes.

As there are many GP studies using multiple-choice information, GNP also uses multiple-choice information in judgment nodes, does not use numerical information. Namely, judgment nodes here are $if-then$ type decision making functions.

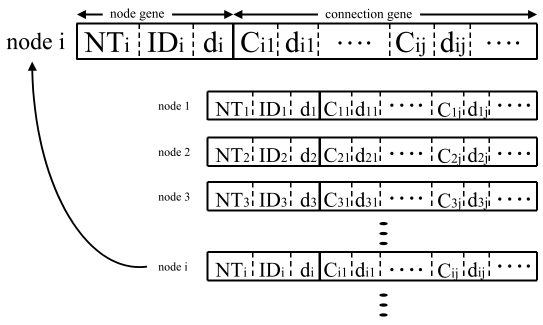

Figure 2: The genotype expression of GNP nodes

Conventional tree based GP starts from their root node, then evaluate leaf nodes one after another. On the other hand, once GNP is booted up, the execution starts from the start node, then the next node to be executed is determined according to the connection from the current activated node. If the activated node is judgment node, the next node is determined by the judgment result at the activated judgment node. When processing node is executed, the next node is uniquely determined by the single connection from processing node.

It is an important point that the activated node isn't compulsorily transferred to the start node in GNP, while other methods do. For instance, GP reinterpret the tree again from the root node after completing the interpretation of the tree, and PADO[4] (Parallel Algorithm Discovery and Orchestration) has also directed graphs with some memories as genes, but it is mainly used for static problems. There is no end node in GNP, therefore, once GNP is booted up, the successive activation of GNP system is carried out according to the network flow until the time limit.

The genotype expression of GNP node is shown in Fig.2. The upper part of Fig.2 describes the gene of node i, then the set of these genes represents the genotype of GNP individuals. All variables in these genes are described by integers. NT_i describes the node type, $NT_i = 0$ denotes START NODE, $NT_i = 1$ when the node i is judgment node and $NT_i = 2$ when the node i is processing node.

ID_i is an identification number, e.g., $NT_i = 1$ and $ID_i = 1$ mean node i is J_1. $C_{i1}, C_{i2}, \ldots,$ denote the nodes which are connected from node i firstly, secondly, ..., and so on depending on the arguments of node i. The total number of the elements of the connection genes depends on the arity of the node's function. d_i and d_{ij} are the delay time. They are the time required to execute the processing of node i and delay time from node i to node C_{ij}, respectively. GNP can become materialized more realistically by setting these delays.

2.2 Genetic Operations of GNP

The following two genetic operators are used in GNP.

- Mutation operator affects one individual. All the connections of each node are changed randomly by mutation rate of P_m.

- Crossover operator affects two parent individuals. GNP evolves the fixed number of nodes as mentioned before, therefore, all the connections of the uniformly selected corresponding nodes in two parents are swapped each other by crossover rate of P_c.

Note that these genetic operators will not change any node functions, they only change the connections among the nodes. Therefore, GNP doesn't evolve the functions of the nodes, but evolves the connections between the nodes.

3. THE PROPOSED METHOD FOR EVOLVING GNP

In conventional GNP, the activated node isn't compulsorily transferred to the start node. The purpose of this is to use repeatedly most of the judgment nodes and the processing nodes in GNP as much as possible. However, there is a possibility that some of the nodes are not used in conventional GNP.

Therefore, in this paper, GNP with plural start nodes is proposed (see Fig.3). In GNP with plural start nodes, the activated node is properly transferred to one of the start nodes. Here, "properly" means that the return to the start nodes is carried out after a certain number of activated processing nodes. The number of nodes used in GNP could be increased by increasing the number of the start nodes. The performance of each individual GNP improves consequently because the increase of the number of the start nodes expands the search space of GNP. The number of the start nodes can be considered breadth of the search, while the number of the activated processing nodes can be considered depth of the search of GNP.

In this paper, the relations of the number of the start nodes and the activated processing nodes are investigated on the problem to solve. The proposed method is studied when there are a lot of connections in each judgment node by which the structure of GNP becomes complex. Actually, the number of connections from the judgment node increases as the problems become complicated ones. In addition, the design on what kinds of branches the judgment nodes should have is so important similarly to the design of the processing nodes in GNP. In this case, the transition or the behavior of GNP is mainly determined by which branch the judgment node selects. Therefore, GNP should have the possibility for selecting every judgment result in GNP. However, the conventional GNP lacks such ability. In order to overcome the above problem, in this paper, GNP with plural start nodes is proposed and the basic study is done. The evolution and performance of GNP are studied by changing the number of start nodes (S) and the number of executable processing nodes per stat node (P) as a basic study.

Concretely, the following is studied.

- The performance of GNP with the combination of S and P is studied, where S*P is total steps of simulations, because carrying out one processing node corresponds to one step.

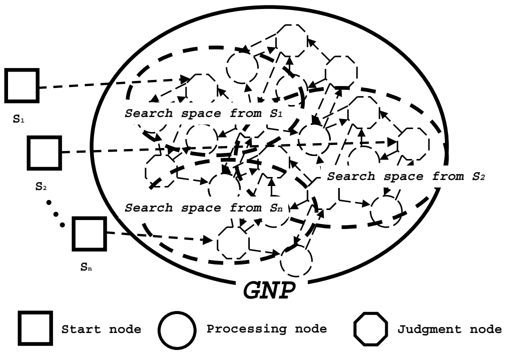

Figure 3: The structure of GNP with plural start nodes

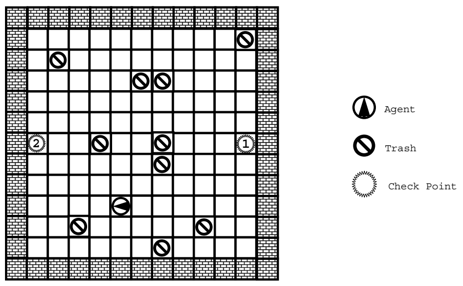

Figure 4: Garbage Collector Problem

- The performance of GNP with the combination of S and P is studied, where S can take values of 1, 2, 5, 10, 25 and P can take values of 10, 5, 2, 1, respectively.

Table 1: Parameter conditions for evolving GNP

Generation	1000
Number of GNP	100
Number of elite GNP	1
Number of crossover individuals	40
Number of mutation individuals	59
Crossover probability P_c	0.1
Mutation probability P_m	0.01
Number per each kind of nodes	5

4. SIMULATION CONDITIONS

4.1 Garbage Collector Ploblem

The garbage collector problem used in this paper is the following.

The garbage collector problem consists of one agent, ten trash, check point 1 and check point 2 on the two dimensional grid like the chessboard (Fig.4). An agent occupies one cell and can move to the forward cell, turn left or right in one step. The agent can collects trash by reaching the cell where trash exists.

Table 1 shows the parameters used in this simulation.

4.2 Fitness Function of GNP

The two dimensional grid is made up of 11×11 cells. The agent can perceive everything in the environment. The agent aims to collect as many trash as possible in 250 steps.

In this paper, the initial position, direction of the agent and the initial position of trash are generated at random at each generation for evolution. The above environment is especially designed for improving the generalization ability of the proposed method.

The fitness for evolving GNP is defined by Eq.(1) and the one for studying generalization is calculated by Eq.(2).

$$Fitness = 100N - D, \tag{1}$$

$$Fitness = 100N. \tag{2}$$

Here, N is the number of collected trash and D is the distance between the agent and the nearest trash after 250 steps.

The total evaluation is done by averaging the above Fitness over ten different environments.

4.3 Judgment Nodes and Processing Nodes

Start nodes, judgment nodes and processing nodes of GNP are described in Table 2. In Table 2, the symbols of 0, 1 and 2 denote a start node, judgment nodes and processing nodes, respectively. The figure in () of the table is the number of connections of each node function. The start node is just a starting position of GNP.

Judgment nodes have some branches corresponding to each judgment result. At the node of "check the distance from the agent to the check point", one of the three branches (0-4, 5-7, 8-) is selected depending on the distance from the agent to the check point. At the node of "check the direction of the agent to check point" and "check the direction of the agent to the nearest trash", there are eight branches (forward, right forward, left forward, right, left, right back, left back, back) depending on the direction of the agent to the check point or the nearest trash. At the node of "check the direction of the agent to the second nearest trash", there is another branch named none, because a case might occur where there is only one trash remained on the two dimensional grid.

The judgment nodes such as "check the distance from the agent to the check point" and "check the direction of the agent to check point" are introduced for the agent to search for the absolute position in the environment.

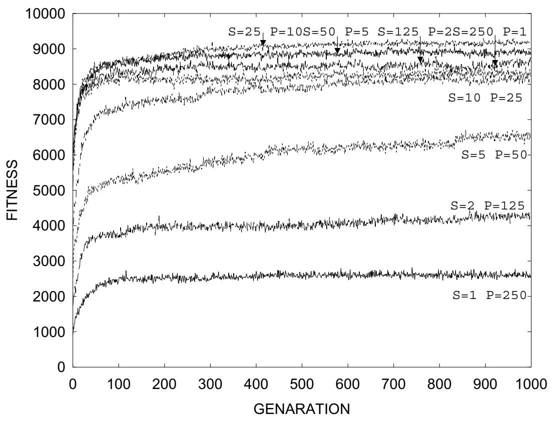

Figure 5: The best trainning fitness values averaged over ten independent trials (in case of S*P=250)

Table 3: Generalization ability of evolved GNPs (Fitness using 1000 environments and 10 best evolved GNPs in case of S*P=250)

	MAX	AVE	MIN	STDEV
S=1 P=250	182.5	169.1	163.2	6.26
S=2 P=125	344.2	314.1	251.8	25.40
S=5 P=50	648.9	524.0	326.3	109.08
S=10 P=25	859.7	864.6	800.6	30.97
S=25 P=10	921.0	864.6	800.6	30.97
S=50 P=5	864.6	814.7	763.2	28.85
S=125 P=2	856.9	785.8	696.5	43.63
S=250 P=1	818.5	753.6	691.5	42.13

5. SIMULATION RESULTS

Simulation results of various cases are compared using the best fitness values averaged over ten independent trials as shown in Fig.5 and Fig.6.

In Table 3 and Table 4, MAX, AVE, MIN and STDEV show the maximum value, average value, minimum value and standard deviation of the fitness of Eq.(2) using 1000 different environments generated at random and ten best evolved GNPs. Therefore, Table 3 and Table 4 show the generalization ability of the evolved GNPs. The generalization ability shows if obtained GNPs can deal with various inexperienced environments or not. High generaligation means that the evolved GNP can acquire the general rules to handle the various environments.

5.1 Results in case of S*P=250

Fig.5 and Table 3 shows that the result of S=25 P=10 is the best and S=1 P=250 (conventional GNP) cannot solve the problem at al. It is clear from Fig.5 and Table 3 that the proposed method is effective even in the problem that cannot be solved by the conventional method. It is generally shown that the problem can be solved when there are more than ten start nodes, while the number of the start nodes is too few, GNP cannot solve the problem. However, the results show that it is not effective to increase the number of start nodes too much.

Table 2: **Functions of a start node, processing nodes and judgment nodes**

	node
0	start node (1)
1	check the distance from the agent to the check point 1 (3)
1	check the distance from the agent to the check point 2 (3)
1	check the direction of the agent to the check point 1 (8)
1	check the direction of the agent to the check point 2 (8)
1	check the direction of the agent to the nearest trash (8)
1	check the direction of the agent to the second nearest trash (9)
2	move forward
2	turn right
2	turn left
2	stay

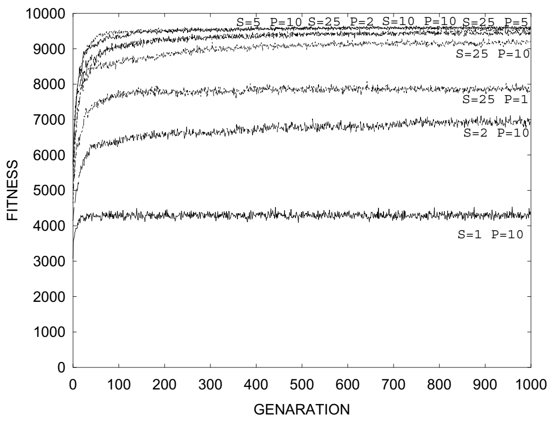

Figure 6: The best training fitness values averaged over ten independent trials (in case of various combinations of S and P)

5.2 Results in case of various combinations of S=25 and P=10

Fig.6 and Table 4 shows that the cases with S=5 P=10 and S=25 P=2 have good performance and the result with few start nodes is bad. These results show that the proposed method becomes effective by selecting an appropriate number of S and P. In other wards, good performance can be obtained as long as an extremely small number of S and P is not chosen. However, it takes a great amount of time to try all cases because there are quite a lot of cases that can be tried. Therefore, it is preferable that the best combination of S and P is acquired by evolution. This is a future work.

6. CONCLUSIONS

In this paper, GNP with plural start nodes is proposed to enhance conventional GNP, and the basic study of it is done. When the number of the start nodes and activated processing nodes is appropriately determined, the proposed GNP can solve the garbage collector problem well. It is proved from simulations that the proposed GNP shows higher performance compared with conventional GNP. However, how to determine an appropriate number of start nodes and activated processing nodes remains to solve in future.

7. ADDITIONAL AUTHORS

Table 4: Generalization ability of evolved GNPs (Fitness using 1000 environments and 10 best evolved GNPs in case of various combination of S and P)

	MAX	AVE	MIN	STDEV
S=1 P=10	326.9	319.9	314.0	5.66
S=2 P=10	642.4	584.7	535.5	33.15
S=5 P=10	971.3	921.1	867.1	27.21
S=10 P=10	946.9	902.6	859.3	26.08
S=25 P=10	921.0	864.6	800.6	30.97
S=25 P=5	928.7	904.2	875.1	13.69
S=25 P=2	956.5	931.4	906.6	15.32
S=25 P=1	710.3	685.1	615.7	27.05

Additional author: Jinglu Hu (Graduate School of Information, Production and Systems, Waseda University, email: `jinglu@waseda.jp`)

8. REFERENCES

[1] R. A. Brooks, "Robust layered control system for a mobile robot", IEEE Journal of Robotics and Automation, Vol.2, No.1, pp. 14-23, 1986.

[2] J. Holland, "Adaptation in Neural and Artificial Systems - An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence -", Ann Arbor: University of Michigan Press, 1975.

[3] J. P. Koza, "Genetic Programming", Cambridge, MA: MIT Press, 1992.

[4] A. Teller and M. Veloso, "PADO, Leaning Tree-structured Algorithms for Orchestration into an Object Recognition System", Carnegie Mellon University, Technical Report Library, 1995.

[5] T. Eguchi, K. Hirasawa, J. Hu and N. Ota, "A study of Evolutionary Multiagent Models Based on Symbiosis", IEEE Trans. on System, Man and Cybernetics -Part B-, Vol.35, No.1, pp. 179-193, 2006.

[6] H. Katagiri, K. Hirasawa and J. Hu, "Genetic Network Programming -Application to Intelligent Agents-", in *Proc. of IEEE International Conference on System, Man and Cybernetics*, pp. 3829-3834, 2000.

[7] K. Hirasawa, M. Okubo, J. Hu and J. Murata, "Comparison between Genetic Network Programming (GNP) and Genetic Programming (GP)", in *Proc. of IEEE CEC International Conference*, pp. 1276–1282, 2001.

[8] John R. Koza, "Genetic Programming II: Automatic Discovery of Reusable Programs", MIT Press, 1994.

[9] H. Katagiri, K. Hirasawa, J. Hu and J. Murata, "Network Structure Oriented Evolutionary Model -Genetic Network Programming and Its Comparison with Genetic Programming-", in *Proc. of GECCO International Conference*, pp. 219-226, 2001.