Comparison between Genetic Network Programming and Genetic Programming Using Evolution of Ant's Behaviors

Hirasawa, Kotaro
Department of Electrical and Electronic Systems Engineering, Faculty of Information Science and Electrical Engineering, Kyushu University

Okubo, Masafumi
Department of Electrical and Electronic Systems Engineering, Graduate School of Information Science and Electrical Engineering, Kyushu University : Graduate Student

Hu, Jinglu
Department of Electrical and Electronic Systems Engineering, Faculty of Information Science and Electrical Engineering, Kyushu University

https://doi.org/10.15017/1515713

出版情報：九州大学大学院システム情報科学紀要. 6 (1), pp.31-37, 2001-03-26. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

九州大学大学院
システム情報科学紀要
第6巻 第1号 平成13年3月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.6, No.1, March 2001

Comparison between Genetic Network Programming and Genetic Programming Using Evolution of Ant's Behaviors

Kotaro HIRASAWA* , Masafumi OKUBO** and Jinglu HU*

(Received December 20, 2000)

Abstract: Recently, many methods of evolutionary computation such as Genetic Algorithm(GA) and Genetic Programming(GP) have been developed as a basic tool for modeling and optimizing the complex systems. Generally speaking, GA has the genome of string structure, while the genome in GP is the tree structure. Therefore, GP is suitable to construct the complicated programs, which can be applied to many real world problems. But, GP is sometimes difficult to search for a solution because of its bloat and introns. In this paper, a new evolutionary method named Genetic Network Programming(GNP), whose genome is network structure is proposed to overcome the low searching efficiency of GP and is applied to the problem on the evolution of behaviors of ants in order to study the effectiveness of GNP. In addition, the comparison of the performances between GNP and GP is carried out in simulations on ants behaviors.

Keywords: Genetic algorithm, Genetic programming, Evolutionary computation, Artificial life

1. Introduction

There have been many difficult problems to solve in our society when we want to deal with complex systems, such that the number of parameters of the problems is large, the environments defining the problems are changing, systems can't be identified accurately and also the space for searching a solution is enormous. In theses cases, we can rely on the leaning and evolutionary mechanisms of living systems in order to solve the above problems, because living systems have found the abundant ecosystems where they interact with each other by symbiotic relations and co-evolve into intelligent creatures using the mechanisms of development, learning and evolution.

Typical computational methods based on the adaptive evolutionary mechanisms of living systems are Genetic Algorithm(GA)[1] and Genetic Programming(GP)[2]. GA has the genome of string structure, while the genome in GP is the tree structure. Therefore GP can be applied successfully to many real problems where complicated programs are to be construced to solve the problems. But, it is generally said that GP is sometimes difficult to search for an optimum solution because the searching space of solutions becomes enormous due to its bloat and introns, that is, searching efficiency of GP is not so high in some cases. In addition, although it is possible to set the past information in the tree structure of GP, a priori knowledge is needed about to what extent the past information should be given.

In this paper, a new evolutionary computational method named Genetic Network Programming(GNP), whose genome structure is networks, is proposed to overcome the problems of GP. In addition to that, the comparison of the performances between GNP and GP is given by applying GNP to the problems on the evolution of behaviors of ants.

There have been also proposed Evolutionary Programming(EP)[3] and Parallel Algorithm Discovery and Orchestration(PADO)[4] which use network structures as their genome. EP has been developed to solve sequence prediction problems defined on the finite alphabet, and its structure is fundamentally finite automaton. Therefore, transition functions for all inputs to all states of the automaton should be defined in advance, in other words, the structure of EP becomes complicated when the number of its inputs and states is large.On the other hand, GNP is possible to construct problem oriented compact genome networks, because in GNP, too many transition functions need not be installed due to the fact that an appropriate number of problem oriented judgement nodes and processing nodes are set in the network.

The programs of PADO are also regarded as N nodes in a derected graph, with as many as N arcs are going from each node. E each node consists of

* Department of Electrical and Electronic Systems Engineering

** Department of Electrical and Electronic Systems Engineering, Graduate Student

an action part and a branch-decision part. There are also the following special nodes in a program such as start node, stop node, and subprogram calling nodes. Although the network of PADO is similar to that of GNP, PADO is oririnally designed to construct the same static programs as GP, which can be seen from the fact that PADO has a stop node, while GNP is mainly used to model dynamic systems. In other words, GNP is a new evolutionary method to construct a generalized descrete event systems by combining program models, e.g., judgement and processing modules using evolutionary computation. In this paper, the main parts are devoted to the comparison between GNP and GP.

The paper is organized as follows. In Section 2, GA and GP are briefly described. GNP is presented in Section 3. As a numerical example, simmulations of the evolution of behaviors of ants are carried out in Section 4. Section 5 is devoted to the conclusions.

2. Conventional Evolutionary Computation

2.1 Genetic Algorithm

Genetic Algorithm(GA) was originally developed from the middle of 1960's to the beginning of 1970's by J. Holland in order to study the adaptive mechanisms of nature and develop an artificial model of evolution. GA is an important predecessor of genetic programming, from which the latter derived its name. GAs have proved useful in a wide variety of real world problems.

The original GA has two main characteristics : it uses a fixed length binary representation and makes heavy use of crossover. The simple representation of individuals as fixed length strings of zeros and ones (**Fig. 1**) is used as the encoding of the problem. The commonest form of crossover is called one point crossover. Two parents individuals of the same length are aligned with each other and a crossover point is chosen at random between any of their component positions. The tails of the two individuals from this point onward are switched, resulting in two new offspring. Another key ingredient to GAs is selection mechanism. This mechanism contains one of the basic principles of evolutionary computation selection -more individuals than the best one have a chance to reproduce.

But, recently it has become known that as the genome of GA is fixed length string, it is hard for GA to obtain appropriate computer programs usually changing their size and form depending on the problems.

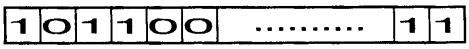


Fig. 1 : Problem representation in the binary string of a GA.

2.2 Genetic Programming

Two researchers, Cramer[5] and Koza[2] suggested that a tree structure should be used as the program representation in a genome to overcome the problems of GA. Koza, however, was the first to recognize the importance of the method and demonstrate its feasibility for automatic programming in general. In his 1989 paper, he provided evidence in the form of several problems from five different areas.

Today there exists a large set of different genetic programming techniques which can be classified by many criteria, such as abstracting mechanisms, use of memory, genetic operators employed, and more. The tree type genome of typical GP is shown in **Fig. 2**. GP consists of one root node and plural number of non-terminal nodes and ternimal nodes, where non terminal nodes are used as functions such as arithmetic function, Boolean function, conditional statements and so on. On the other hand, terminal nodes are comprised of the inputs to the GP program or used for the particular processing which depends on the problems concerned.

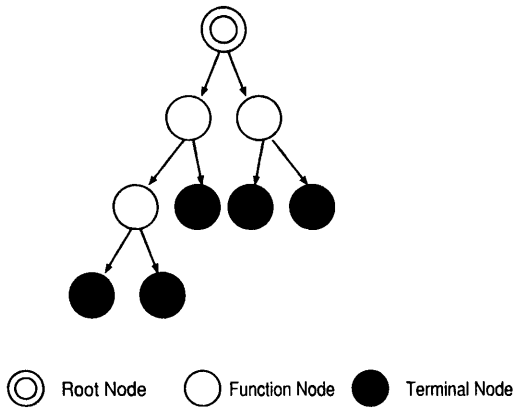
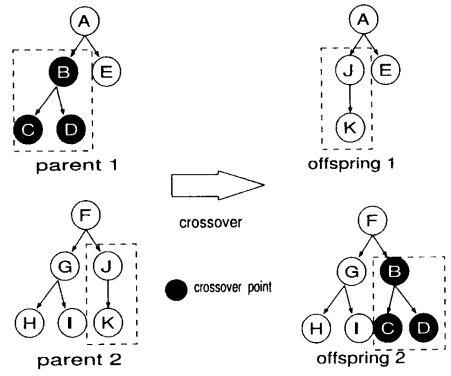


Fig. 2 : Problem representation in the tree of a GP.

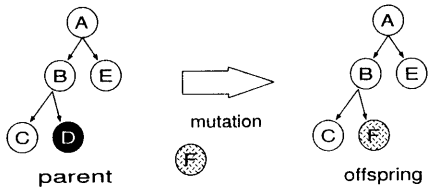
When GP is used to generate the behavior sequences of the dynamic agents, their sequences can be obtained by processing each node starting from the root node.

The following genetic operators shown in **Fig. 3** are usually used in GP. The first one is to swap the selected subtrees between the two parents, the sec-

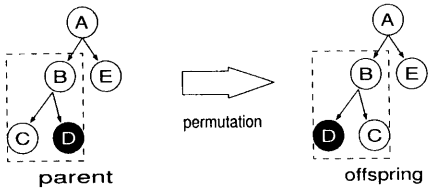
ond one is to select a point in the tree randomly and replacing the existing subtree at that point with a new randomly generated subtree, and the last one is to permute arguments of a node.



(Offspring resulting from crossover)



(Offspring resulting from mutation)



(Offspring resulting from permutation)

Fig. 3 : Various genetic operators in GP

GP evolve a population of programs in parallel. The driving force of this simulated evolution is some form of fitness-based selection. Fitness -based selection determines which programs are selected for further improvements. The GP problem representation is, theoretically, a superset of the representations of all other machine learning systems. This systems from both its variable length structure and its freedom of choice of functions and terminals. G-P's enormous freedom of representation is a mixed blessing. With such a huge search space, an algorithm might have to search for a long time.

3. Genetic Network Programming

3.1 Basic Structure of GNP

As was stated before, the genome of GNP has network structure which consists of two kinds of nodes, i.e., judgement nodes and processing nodes as shown in **Fig. 4**.

Judgement nodes and processing nodes correspond to non terminal nodes and terminal nodes in GP, respectively. In addition, time delays can be set in and between the nodes. Time delay d_i denotes the processing time in node i, while time delay d_{ij} means the time period from the end of the processing of node i to the beginning of the processing of node j. Time delays are installed in GNP so that it can model the dynamic discrete event systems more easily than GP.

In other words, GNP is proposed to model the generalized discrete event systems by evolving the network genome. Furthermore, the genome network of GNP has more problem oriented architectures than that of general automatons in a sense that judgement nodes in GNP are designed to suit to the problems concerned more specically.

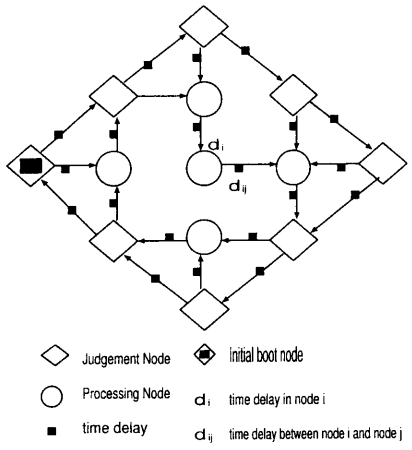
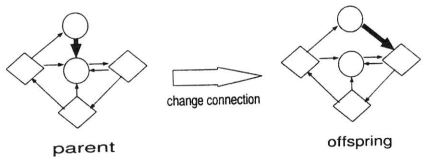


Fig. 4 : The genome structure of GNP

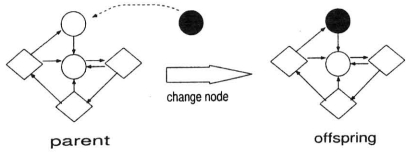
3.2 Genetic Operators of GNP

The following genetic operators shown in **Fig. 5** are used in GNP. Mutation operators on only one individual. One type of mutation operator in network GNP selects a point in the network randomly and changes the connection of the node at that point (offspring resulting from changing connection), or replaces the existing node at that point with a new randomly generated node (offspring resulting from

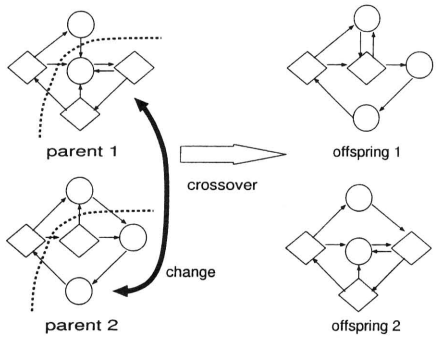
changing node). Network-based crossover proceeds by the following. Choose two individuals as parents, divide and select a subnetwork in each parent, and swap the selected subnetworks between the two parents. The resulting individuals are the children (offspring resulting from crossover).



(Offspring resulting from changing connection)



(Offspring resulting from changing node)



(Offspring resulting from crossover)

Fig. 5 : Various genetic operators in GNP

After genetic operations, fitness-based selection determines which network genomes are selected for further improvements.

4. Simulations

Simulations are carried out in order to compare the evolutionary performances between GNP and GP and to show the effectiveness of GNP.

4.1 Simulation Conditions

Simulations of collecting food by ants are explained as follws. When an ant find food, the ant goes back to the nest dropping pheromone. Ants are likely to be attracted to the places where pheromone is. Pheromone is volatile. As a result as long as food exists, the pheromone road grows and finally it disappears according to the run out of food. This shows that the efficiency of collecting food by ants is increased by using the medium of pheromone.

The above behaviors of ants are modeled by GNP and GP and compared in order to demonstrate that GNP can model the ant's behavior more efficiently and effectively than GP.

The exact simulation model is as follows : There are two spots where food is placed and one nest in an artificial field which is made of 32×32 torus boxes as shown in **Fig. 6**. The size of the food mountain is $3 \times 3 \times 8$, therefore there exist a total of 144 pieces of food and the size of the nest is 3×3. The total number of ants is 20.

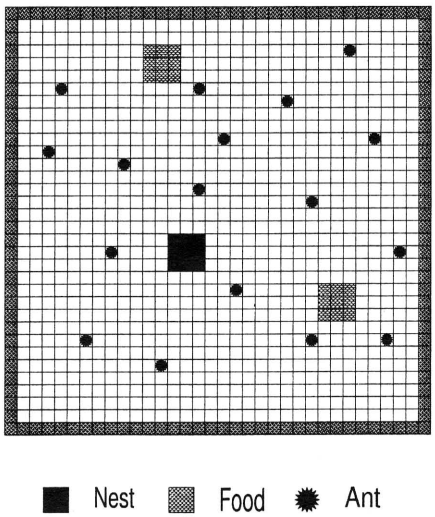


Fig. 6 : The ant world

The mission of ants is to bring as much food as possible to the nest within a fixed time period. The ant moves one box at one time step. Furthermore ants can get the following information : (1) the position of an ant itself in the field, (2) the direction to which an ant is moving (4 directions are possible), (3) the information about whether an ant has food or not.

The following four kinds of respective judgement nodes and processing nodes shown in **Table 1** are used in GNP.

When pheromone is dropped, it is spreaded to 3×3 boxes and disappears 10 time steps later, because of its volatility.

Fitness function is defined as the number of remaining food, that is, the number of food which could not be brought to the nest within 1000 time steps. Therefore the optimal value of fitness func-

tion is zero.

Table 1 : Functions of judgement nodes and processing nodes

J/P	function
J	whether an ant is on a food box or not
J	whether an ant has food or not
J	if there is food on the next going box, then an ant steps to the box
J	if there is pheromone on the next going box, then an ant steps to the box
P	step to the next box randomly
P	step to the nest
P	pick up food
P	drop pheromone

J : judgement node , P : processing node

4.2 Simulation Results

The following three kinds of evolutionary computations are studied, i.e., GNP, GNP with GP type and GP. Here GNP with GP type is a kind of GNP that starts from the start node and returns to the start node again when it reaches the stop node. Therefore GNP with GP type is similar to PADO system.

4.2.1 Study of changing the maximum depth of the tree in GP

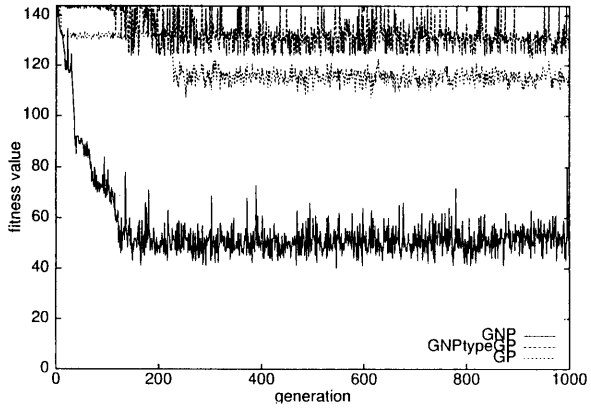
In this subsection, we studied the effect of changing the maximum depth of the tree in GP. Simulation conditions are shown in **Table 2**.

Table 2 : Simulation conditions of changing the maximam depth of the tree in GP

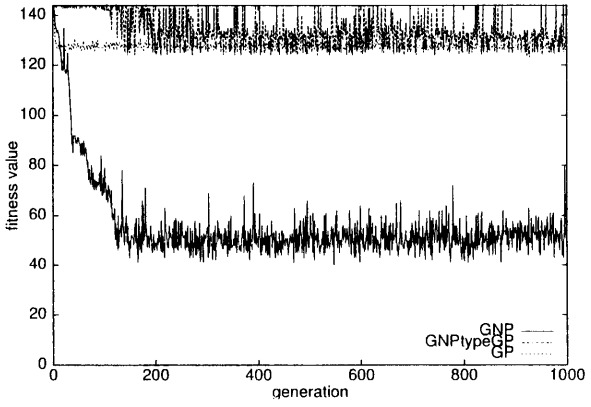
item		number
the number of individuals		100
the number of nodes		24
GNP	connection changing rate	0.7
	node changing rate	0.1
	crossover rate	0.1
GP	crossover rate	0.5
	mutation rate	0.05
	permutation tate	0.1

The mutation rate and crossover rate etc. in **Table 2** is the optimum rate by which the minimum fitnesses of GNP and GP are obtained, respectively. For simplicity all the delay are set at one time step in GNP.

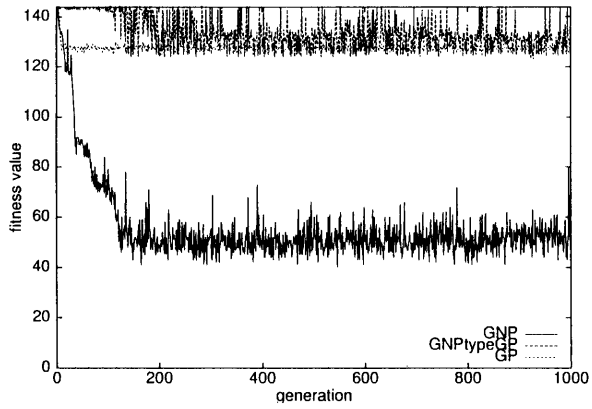
Average fitness values of the three kinds of methods mentioned before are shown in **Fig. 7**, changing the maximum depth of the tree in GP.



(Maximum depth of tree is 5)



(Maximum depth of tree is 8)



(Maximum depth of tree is 10)

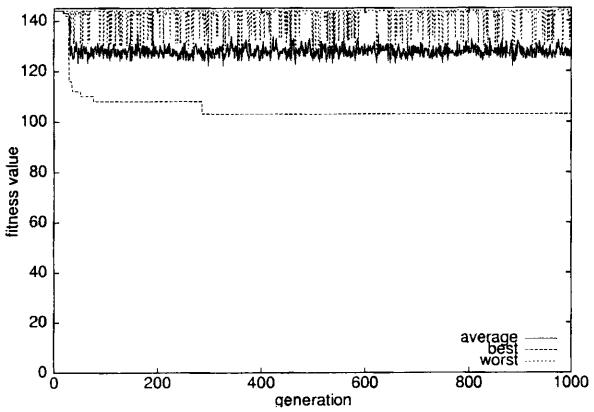
Fig. 7 : Learning curves of ant's behavior changing the maximum depth of tree

It is clarified from **Fig. 7** that GNP is the best, followed by GP, and GNP with GP type is the worst, when the maximum depth of the tree in GP is five. On the other hand, when the maximum depth is eight or ten, the fitnesses of GNP with GP type and GP are almost the same.

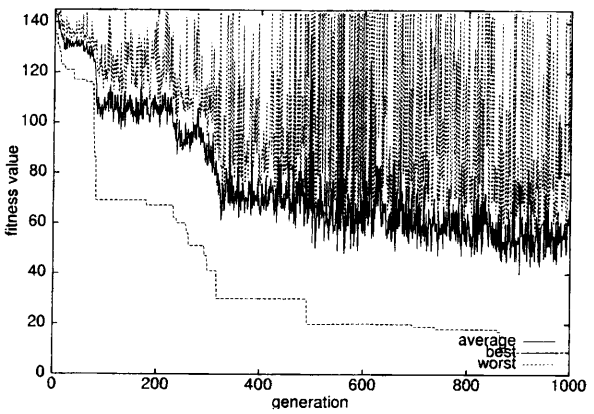
From the above it is estimated that the efficiency of searching for solutions in GP degenerates according to the increase of the maximum depth of the tree, because the searching space becomes huge due to the bloat of GP.

Therefore it is concluded that GNP can be evolved more efficiently and more quickly than GP irrespective of the maximum depth of the tree in GP. This is because the number of nodes in GNP does not change even after its genetic operations, while in GP the tree has the probability to grow to the maximum depth as mentioned before.

4.2.2 Study of changing the number of individuals



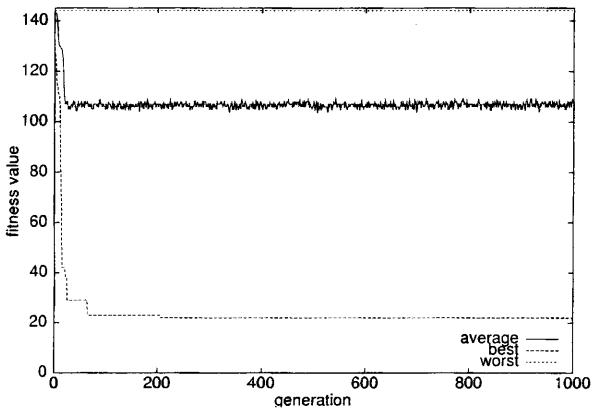
(In the case of GP)



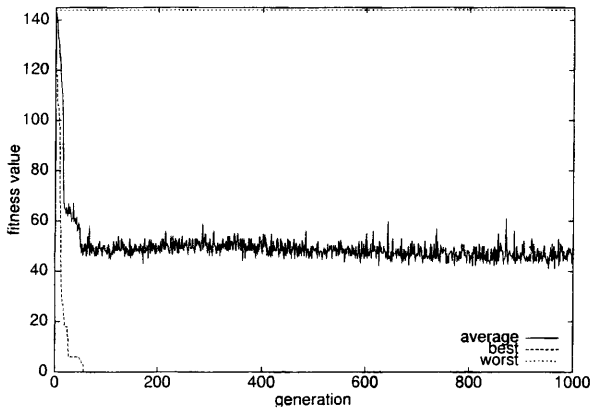
(In the case of GNP)

Fig. 8 : Learning curves of ant's behavior using 20 individuals

In this subsection, the effect of changing the number of individuals was studied, where the maximum depth of the tree in GP is set at eight. Simulation conditions are the same as **Table 2**.



(In the case of GP)



(In the case of GP)

Fig. 9 : Learning curves of ant's behavior using 500 individuals

Figures, 8,9 show the maximum, minimum and average fitnesses of GNP and GP with the number of individuals being 20 and 500, respectively. The upper part of the figures is about GP, while the results of GNP are shown in the lower part of the figures.

From **Fig. 8**, we can see that the searching for the solutions using GNP is carried out more efficiently than GP when the number of individuals is relatively small 20. On the other hand, enen when the individuals increase to relatively large 500, still the seaching of GNP is carried out more efficiently than GP, although the efficiency gap between GNP and GP becomes small.

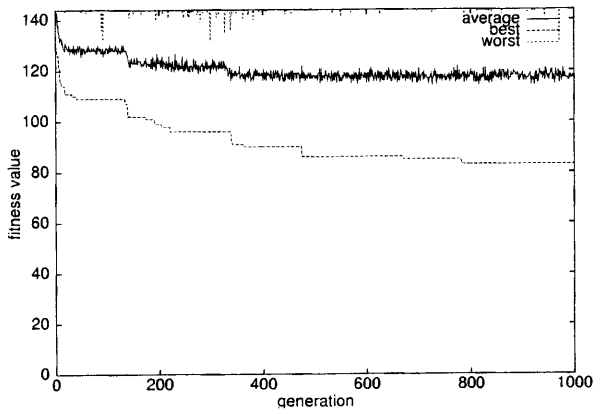
Agreat number of individuals are generally used in GP in order to obtain the optimal solution. But

it is clarified from **Fig. 8** and **Fig. 9** that GNP can find the optimal solution easily in spite of the small number of individuals.

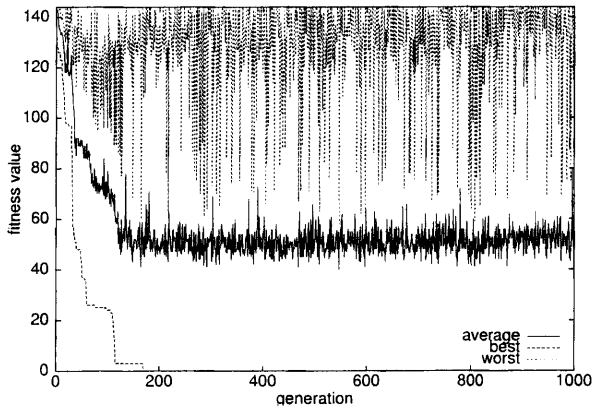
4.2.3 Study of changing the number of the nodes

In this subsection, simulations for studying the effect of changing the number of nodes in GNP were carried out with the number of individuals being fixed at 100. Simulation conditions are the same as **Table 2**.

Fig. 10 shows the maximum, minimum and average fitnesses of GNP when the total number of nodes is 8, i.e., one node per each function, and a total of 24 nodes are used, i.e., three nodes per each function, respectively.



(In the case of 8 nodes)



(In the case of 24 nodes)

Fig. 10 : Learning curves of ant's behavior when changing the number of nodes

From **Fig. 10**, it is clarified that at least three nodes per function are needed to obtain the optimal solution. But, it is supposed two many nodes may cause the low searching efficiency because of a huge search space as in the case of GP bloat.

5. Conclusion

We proposed a new evolutionary computation algorithm named Genetic Network Programming(GNP), where the genome employs network structures in stead of tree structures in Genetic Programming(GP), and compared the performance between GNP and GP using the evolution of behaviors of ants.

In addition, it was studied how GNP, Evolutionary Programming(EP) and Parallel Algorithm Discovery and Orchestration(PADO) are differently constructed and applied to real problems, although they all have the same network genome structures.

From simulations on ants behavios, it is clarified that the average fitness of GNP is always better than that of GP under various conditions, even when there is not so much difference between the two with respect to the optimum fitness.This is because the efficiency of searching for solutions in GNP is higher than that of GP due to the network genome structures of GNP, which essentially prohibit GNP from growing without limit.

References

1) J. H. Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975, MIT Press 1992
2) J. Koza, "Genetic Programming, On the Programming of Computers by means of Natural Selection", MIT Press, 1992
3) L. J. Fogel, A. J. Owens, and M. J. Walsh, "Artificial Intelligence through Simulated Evolution", Wiley, 1966
4) A. Teller and M. Ueloso, "PADO : Learning tree structured algorithms for orchestration into an object recognition|system", Technical Report CMU-CS-95-101, 1995, Department of Computer Science, Carnegie Mellon University, Pittsburg, PA.
5) N. L. Cramer, "A representation for the adaptive generation of simple sequential programs", Proceedings of an International Conference on Genetic Algorithms and the Applications, pp. 183-187, 1985
6) J. R. Koza, "Hirerarchical genetic algorithms operating on populations of computer programs", Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, IJCAI-89, Vol. 1, pp. 768-774, 1989
7) H. Katagiri, K. Hirasawa and J. Hu, "Genetic Network Programming-Apprication to Intelligent Agents-", Proc.|of IEEE International Conference on|System, Man and Cybernetics, pp. 3829-3834, 2000