Genetic Network Programming with Automatically Defined Groups for Assigning Proper Roles to Multiple Agents

Tadahiko Murata
Faculty of Informatics,
Kansai University
2-1-1 Ryozenji, Takatsuki, Osaka 569-1095, Japan
+81-72-690-2429
murata@res.kutc.kansai-u.ac.jp

Takashi Nakamura
Computer Science Major,
Kansai University Graduate School
2-1-1 Ryozenji, Takatsuki, Osaka 569-1095, Japan
+81-72-690-2429
fb4m110@edu.kutc.kansai-u.ac.jp

ABSTRACT

In this paper, we apply a Genetic Network Programming (GNP) Architecture using Automatically Defined Groups (ADG) to a multi-agent problem where cooperation of agents are required. GNP is a kind of evolutionary methods inspired from Genetic Programming (GP). While GP has a tree architecture, GNP has a network architecture with which an agent works in the virtual world. In GNP with ADG, each agent is assigned to a group according to its role to complete some task of a cooperative problem. We consider two types of problems in this paper: one problem is to assign an appropriate role to each agent according to its ability, and the other is to assign a proper role to each agent with the same ability. While the first problem has the specific conditions as for the ability of an agent, the latter is a general problem. We show the effectiveness of GNP with ADG through computer simulations on the two types of load transportation problems.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *Multiagent systems*

General Terms

Algorithms, Performance.

Keywords

Genetic Network Programming, cooperation, role assignment.

1. INTRODUCTION

Recently many researchers have investigated on automatic design of complex systems using evolutionary computation (EC) techniques. Genetic Programming [9] is one of well-known EC techniques that uses a tree structure to describe solutions of numeral problems, combination optimization problems, and action control problems for agents. However, GP is known to have difficulty to converge in one near-optimal solution or to search an optimal structure efficiently due to its large solution space. When it is applied to dynamic problems such as action control problems, some actions should be selected in a chain of actions. That is, an action should be selected not only by the current situation but also by the actions already taken. If a designer of GP notice the necessity of that issue, s/he can try to define appropriate nodes to utilize previous information. However, it is difficult to define appropriate nodes in advance even if s/he knows its importance.

In order to cope with such problems, a new architecture called Genetic Network Programming (GNP) has been proposed [7]. It is inspired from GP, but it does not have a tree architecture, but has a network architecture. While Evolutionary Programming (EP) [1] also has a network architecture, it may be difficult to predefine all transitions in advance. GNP can work with only problem-dependent nodes like GP so that the designer predefine only a small number of types of nodes. PDGP [13] was also proposed to represent graph-like structures, however, it can not represent the repetition of actions since connections among nodes are restricted only to upwards or the adjacent.

In GNP [7], every agent takes actions according to an identical set of rules. It is called a homogeneous model. In GP research works, however, various methods have been proposed for heterogeneous model [10,4,6]. Luke and Spector [10] showed that the heterogeneous model performs better than the homogeneous model because the ability of agents becomes higher and agents perform more complex cooperative behaviors. However, the search efficiency for GP becomes worse in the heterogeneous model as its searching space increases. While these previous studies [10,6] considered the heterogeneous model, the number of agents was restricted to two to four. We consider 20 agents in this paper.

In order to obtain multiple roles for a number of agents, Hara and Nagao [2,3] have proposed a combined model of the homogeneous and the heterogeneous model. That is, they assigned a role to a class of agents. They developed several trees for classes of agents by GP, and the agents with the same role refer the same tree. Their model is called Automatically Defined Groups (ADG) [2]. In their ADG model, the number of roles does not have to be predefined, but it is obtained automatically through evolutionary process. They showed their effectiveness for a load transportation problem [2,3] and the tile world problem [14].

We have combined GNP and ADG for developing control rules for multiple agents [11] (for details, see [12]). In this paper, we evaluate the performance of GNP with ADG by applying it to two types of problems: one problem is to assign an appropriate role to each agent according to its ability, and the other is to assign a proper role to each agent where all agents have the same ability. While the first problem has the specific conditions as for the ability of an agent, the latter has few conditions as a general problem, so that the problem has larger solutions space to be searched. We show the effectiveness of GNP with ADG through computer simulations on these two problems.

2. GENETIC NETWORK PROGRAMMING

2.1 Basic Architecture of GNP

In this section, we describe the basic architecture of GNP [5,7,8]. GNP uses a network architecture instead of using a tree architecture. Fig. 1 shows the basic structure of GNP. In GNP, we have three types of nodes: start node, judgment node and processing node. In Fig. 1, the start node, the judgment node and the processing node are denoted by a square, a diamond and an open circle, respectively. The start node is like a root node in GP. The other two nodes, the judgment node and the processing node, correspond to the function node and the terminal node, respectively. Therefore we can employ the same function node and the terminal node used in GP as the judgment node and the processing node in GNP. Main difference between GP and GNP lies in the terminal node or the processing node. As shown in Fig. 2, the terminal node of GP can not have a further connection. That is the reason why it is called as "terminal" node. On the other hand, the processing node in GNP can connect other nodes as shown in Fig. 1.

The difference between GNP and GP lies only in their architectures, but it brings great difference between them. For example, because the action of an agent is determined only in the terminal node in GP, the agent should return to the root node to take a next action. On the other hand, an agent in GNP acts when it finds a processing node, and it does not have to return to the start node after making decision.

As for another disadvantage of GP, the performance of the tree may change greatly if the sub tree is exchanged by genetic operations at the node near to the root node in GP. On the other hand, the node exchange in GNP does not have such a great influence on the performance of the network.

2.2 Chromosome Representation

In order to generate a network as an individual in GNP, we assign $N+1$ nodes for one network randomly. The representation of each node is shown in Fig. 3. Each node has its ID number i ($i = 0,1,2,...,N$). As shown in Fig. 3, each node consists of two parts: node gene and connection gene. In the node gene, NT_i shows the type of the node i: "0" denotes the processing node and "1" denotes the judgment node. ID_i indicates the function of the node. If we assume that there are P types of the processing node, ID_i varies from 0 to $P-1$. When there are J types of the judgment node, ID_i varies from 0 to $J-1$. The function of the node i is defined according to the values of NT_i and ID_i. As for the start node, we do not assign NT_0 and ID_0 in the node gene.

In the connection gene, C_{ij} indicates the j-th connection from the node i, and n_i^{ark} shows the number of connections from the node i. If $NT_i = 0$, $n_i^{ark} = 1$ because the processing node has only one connection. When $NT_i = 1$, $n_i^{ark} \geq 2$ because a judgment node has several connections according to its condition. The value of the C_{ij} indicates the ID number of the node connected from the node i.

In the initial generation, we first generate N nodes by assigning NT_i and ID_i for $i = 1,...,N$ randomly. After that we copy the set of generated N nodes for N_{pop} individuals. To the N nodes in each individual, we randomly assign the connection gene C_{ij} for each individual to form a population with the specified number of individuals. Therefore, the node gene of each individual has the same NT_i and ID_i, but the connection gene has different connections among N nodes.

2.3 Genetic Operations for GNP

GNP has the two types of genetic operations: crossover and mutation. Fig. 4 shows an example of the crossover operation for GNP. The procedure of the crossover is as follows:

[Crossover in GNP]

Step 1: Using the tournament selection, select two networks for crossover (Parents 1 and 2 in Fig. 4).

Step 2: Select nodes randomly in one network (closed and slashed circles and diamonds).

Step 3: Exchange the selected nodes between two networks (right side of Fig. 4).

Step 4: Repeat Step 1 through 4 until the prespecified number of offspring is generated.

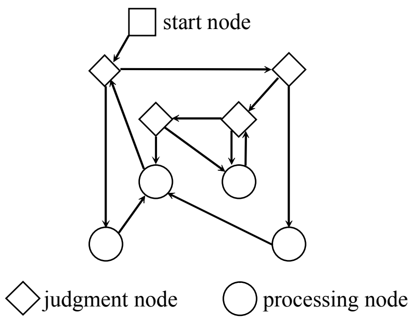

Figure 1. Basic Structure of GNP.

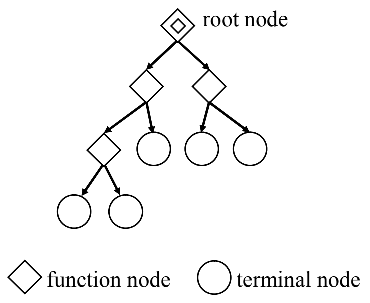

Figure 2. Basic Structure of GP.

	Node Gene		Connection Gene				
node i	NT_i	ID_i	C_{i1}	C_{i2}	----	----	$C_{in_i^{ark}}$

Figure 3. Representation for the node.

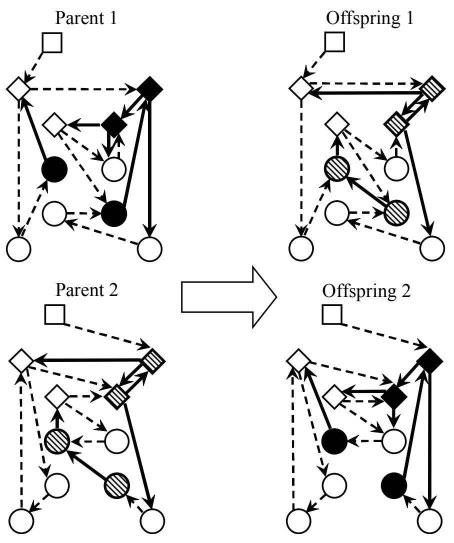

Figure 4. Crossover in GNP.

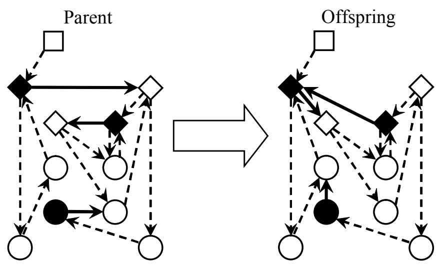

Figure 5. Mutation for connection gene in GNP.

As shown in Fig. 4, the connections of the randomly selected four nodes are exchanged by this crossover.

Fig. 5 shows a mutation operation for connection gene. The procedure of the mutation for the connection gene is as follows:

[Mutation in GNP]

Step 1: Select a network randomly.

Step 2: According to the mutation probability P_m, select a connection (Closed circle and diamonds).

Step 3: Turn the value of the selected connection to another node ID.

Since these genetic operations modify the network among the nodes, all network generated by them have the same number of nodes.

2.4 Algorithm of GNP

Using the initialization process and the genetic operations in Subsections 2.2 and 2.3, we form the following algorithm for GNP.

[GNP Algorithm]

Step 1 (Initialization)
  Initialize the population with N_{pop} individuals.

Step 2 (Fitness evaluation)
  Calculate the fitness value of each individual, and find an elitist individual with the best fitness value in the population.

Step 3 (Genetic operations)

Step 3-1 (Selection):	Select parents for crossover by the tournament selection.
Step 3-2 (Crossover):	Apply the crossover operator to the selected parents.
Step 3-3 (Mutation):	Apply the mutation operator to the connection genes.
Step 3-4 (Elite strategy):	Preserve the elitist individual found in Step 2 or Step 5.

Step 4 (Replacement)
  Replace the newly generated population with the previous population.

Step 5 (Fitness evaluation)
  Calculate the fitness value of each individual, and find an elitist individual with the best fitness value in the population.

Step 6 (Termination Condition)
  Terminate the algorithm if the specified condition is satisfied. Otherwise return to Step 3.

3. GNP with the ADG MODEL

3.1 Automatically Defined Groups

We employ the Automatically Defined Groups (ADG) model [2,3] for developing several roles of agents and assigning them to one of the roles. According to its role, each agent refers to the action control rules described by GNP. In the concept of the ADG model, each agent has its own role, and the agents with the same role take actions under the identical set of rules. This is likened to social insects such as bees and ants. They have several specialized classes among them such as queens, drones, workers and so on. It is considered that a number of workers take action under the identical rules. While these rules of a class are developed by GP in [2,3], we have introduced this model into GNP [11,12]. In the ADG model, a set of several classes is considered as one individual to be governed by genetic operations. That is, a set of several networks of GNP is considered as an individual. We refer this individual including several networks as an ADG individual in this paper.

In our ADG model, each agent belongs to one of the classes of the network. Therefore each ADG individual has three pieces of information: the class structure, a network structure for each class, and the class IDs for all agents.

3.2 Genetic Operations for the ADG Model

Genetic operations for the ADG model aim to acquire several classes according to the number of roles of agents. There are two tasks to attain this aim:

1) Acquire the number of roles, and develop a network of GNP for each role.
2) Assign an appropriate role to each agent.

In order to attain these tasks, Hara and Nagao [2,3] proposed a group mutation and a crossover for ADG individuals. They proposed a group mutation to change the members of a class (or a group) as follows:

[Group Mutation in ADG]

Step 1: According to the group mutation probability P_{gm}, select an ADG individual.

Step 2: For the selected ADG individual, select an agent, and identify the class (or the group) of the selected agent.
Step 3: Change the class ID of the selected agent randomly.
Step 4: If the class ID specified in Step 3 is the same class ID to which the agent belonged, generate a new network only for the selected agent. The structure of the newly generated network is the same one to which the agent belonged. Then go to Step 5. If the class ID chosen in Step 3 is different, go to Step 5.
Step 5: Return to Step 1 until the mutation is applied to all the ADG individuals.

By this mutation, the class structure (i.e., the number of classes and assigned agents to each class) is modified before the crossover operation.

After applying the group mutation to each ADG individual, the following crossover is applied to the ADG population.

[Crossover in ADG]
Step 1: According to the crossover probability P_{gc}, select two ADG individuals among the ADG population.
Step 2: Select one agent randomly.
Step 3: Identify the network of GNP to which the selected agent refers in each of the selected ADG individuals. Let Net and Net' denote each network of the selected ADG individuals, respectively.
Step 4: Identify the set of agents $A(Net)$ that refer to the network Net as action control rules. Identify $A(Net')$, too.
Step 5: Apply the crossover operator for GNP (see Subsection 2.3) to the selected networks Net and Net'.

In Step 5, the following three relations between $A(Net)$ and $A(Net')$ are examined and the merger or the division of the classes is implemented before applying the crossover operator.

(Type a) $A(Net) = A(Net')$,
(Type b) $A(Net) \subset A(Net')$ or $A(Net) \supset A(Net')$,
(Type c) Otherwise.

In the case of Type a, the crossover for GNP is applied to Net and Net' without the merger or the division of the classes. If $A(Net)$ and $A(Net')$ are in the relation of Type b, the division of the class is occurred in the larger set. On the other hand in Type c, the merger of the classes is taken place when $A(Net)$ and $A(Net')$ are neither the same set nor in the inclusion relation. Using Figs. 6 - 8, we explain these three cases.

In Figs. 6 - 8, two ADG individuals are already selected for the crossover operator, and Agent 2 is selected in Step 2 of the crossover operation for the ADG model.

(Type a)
Fig. 6 shows an example of this case. In this case, there is no change among the class structure in an ADG individual since Agent 2 is a member of the set {1, 2} in both the ADG individuals in Fig. 6. Apply the crossover for GNP in Subsection 2.3 to the selected networks.

(Type b)
When Agent 2 is selected in Step 2, it is a member of the set {2} in the left parent, and a member of the set {1, 2, 3, 4} in the right one in Fig. 7. If the network for the set {1, 2, 3, 4} in the right parent is modified with the network in the left by the crossover, Agents 1, 3, 4 are also influenced by the modification. In order to restrict the influence of the network for Agent 2 in the left parent, the same network of {1, 2, 3, 4} is newly generated in the right and Agent 2 is assigned to the newly generated network. After that, apply the crossover to the network of the left parent and the newly generated network in the right parent.

(Type c)
As shown in Fig. 8, Agent 2 is included in {1, 2} in the left parent, and in {2, 3} in the right one. Since these two sets are not included each other, the union of these two sets is assigned to the networks generated by the crossover.

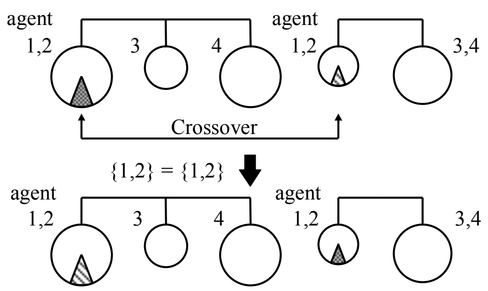

Figure 6. Crossover for ADG in the case of $A(Net) = A(Net')$ (Type a).

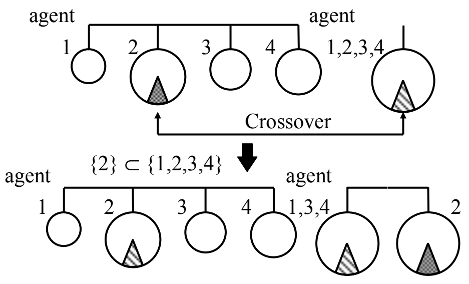

Figure 7. Crossover for ADG in the case of $A(Net) \subset A(Net')$ or $A(Net) \supset A(Net')$ (Type b).

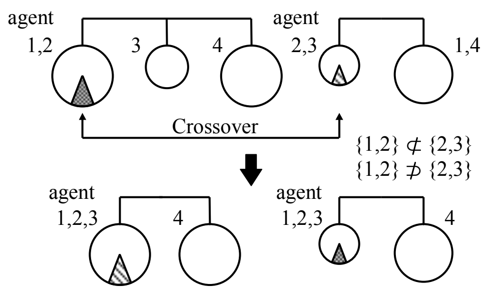

Figure 8. Crossover for ADG in the case of no inclusion of $A(Net)$ and $A(Net')$ (Type c).

3.3 Overall Algorithm of GNP with ADG

Using the ADG model, we develop networks of GNP by the following algorithm:

[Algorithm for GNP with ADG]

```
Step 1 (Initialization)
  Initialize the population with $N_{pop}^{ADG}$ ADG individuals.
Step 2 (Fitness evaluation)
  Calculate the fitness value of each ADG individual, and find an
  elitist individual with the best fitness value in the population.
Step 3 (Genetic operations)
  Step 3-1 (Group Mutation): Select $N_{pop}^{ADG}-1$ ADG individuals
by the tournament selection, and apply the group mutation to
ADG individuals according to the group mutation probability.
  Step 3-2 (Crossover):       Apply the crossover operator in
    Subsection 3.2 to the selected parents.
  Step 3-3 (Mutation):        Apply the mutation operator for
    GNP in Subsection 2.3 to the connection genes.
  Step 3-4 (Elite strategy):  Preserve the elitist individual
    found in Step 2 or Step 5.
Step 4 (Replacement)
  Replace the newly generated population with the previous
  population.
Step 5 (Fitness evaluation)
  Calculate the fitness value of each ADG individual, and find an
  elitist individual with the best fitness value in the population.
Step 6 (Termination Condition)
  Terminate the algorithm if the specified condition is satisfied.
  Otherwise return to Step 3.
```

4. LOAD TRANSPORTATION PROBLEM

In this section, we employ a simple load transportation problem [2,3] to show the effectiveness of GNP comparing to GP. Later, the effectiveness of the ADG model is shown by comparing it with the heterogeneous model using the same problem. We consider two types of problems: one problem is to assign an appropriate role to each agent according to its ability, and the other is to assign a proper role to each agent with the same ability.

4.1 Problem Settings

In the load transportation problem [2,3], there are two types of loads that differ in weight. Each of the two types loads is placed in a point in a two-dimensional grid world (11×10). That is, the heavy loads are placed at one point (10, 9) in the environment, and the light ones are placed at the other point (0, 9). These loads are inexhaustible in this problem settings. The number of agents is 20. The aim of this transportation problem is to carry as many loads as possible to the goal point (5, 0) within a limited duration. The fitness of a team of 20 agents is measured by how many loads are transported to the goal point within the allotted time. The score of a heavy load is 5 and that of light one is 1. We allow 100 time steps for each agent. During that time steps each agent can bring a load back to the goal point three times if it moves along with a short way between the goal and load places.

We consider two types of settings for this transportation problem. In the first problem, there are only five agents that can bring either load among them. The other 15 agents can carry only a light load. The aim of this problem can be said that to find the five agents with the ability to carry heavy loads first and to assign them that task. And the task to carry light loads is assigned to the other agents.

In the other problem settings, we have 20 agents that have the same ability. But still we have two places where there are loads. In order to carry all the loads in two places, the agents should be divided to be assigned to one of load places. This problem has many solutions to assign role to each agent, but it may become a bit difficult since the search space becomes larger than the first one. We call the first problem as Ability-Based Role Assignment Problem, and the second one as General Role Assignment Problem.

4.2 Ability-Based Role Assignment Problem

4.2.1 Comparison Between GNP and GP

In this section, we compare the performance of GNP with GP using the load transportation problem, that is the role assignment problem according to agents' ability. We employed the homogeneous model in this section. That is, the algorithm in Subsection 2.4 is used for GNP. In order to compare the performance of GNP with GP, we employ the same judgment (function) and processing (terminal) nodes in both the algorithms. Table 1 shows two judgment nodes and four processing nodes. These nodes are commonly used in GNP and GP. We did not employ "Prog N" node to concatenate processing nodes in GP. If we employ "Prog N" nodes in GP, there is a possibility to enhance the performance of GP.

Since we prepare five nodes for each node type, each initial individual for GNP consists of 30 nodes with a start node. In order to use the same number of nodes in GP, we specified the maximum depth of the tree was five.

Table 1. Nodes used for GNP and GP.

Name	Description
if_carrying_load	Carry load or not
if_load_here	There is a load or not
Pick_up	Pick up load at the current position
Move_goal	Move to the goal point
Move_heavy_load	Move to the heavy load point
Move_light_load	Move to the light load point

Table 2. Parameter Specifications in GNP and GP.

Parameter	Value
Population Size (Common)	200
Tournament Size (Common)	5
Elite Size (Common)	1
Crossover Rate (Common)	0.9
Mutation Rate (Common)	0.01
Group Mutation Rate (GNP with ADG)	0.9

Table 3. Results of GNP and GP.

Fitness	GNP	GP
Average	105.8	75.0
Standard Deviation	7.19	0.00
Max	120	75
Min.	75	75

Table 2 shows the parameter specifications specified for all the algorithms by preliminary experiments. We applied the genetic operations with 500 generations in each trial of GNP and GP. In this problem, the best fitness of a team, $120 = (5\times5+1\times15)\times3$, is attained by assigning appropriate roles to five agents to carry the heavy loads and the other 15 agents to carry light loads. Figs 9 and 10 show the maximum, minimum and average fitness over 100 trials by GNP and GP. From these figures, we can see that GNP can find the network that enables agents to perform with the best fitness. On the other hand, GP could not find the best fitness 120 from the 100 trials.

Table 3 shows the statistical result of 100 trials at the final generation. According to the results, we can see that GP could not find the tree structure that enables agents to get the highest fitness value while GNP could obtain the network to attain the maximum fitness value. Figs 11 and 12 show the tree and the network structure obtained by GP and GNP, respectively. In these figures, the nodes that have no effect in the decision making of an agent are omitted. From Fig. 11, we can see that GP could produce the tree which enables only five agents to carry heavy loads. This tree is the optimal tree generated by GP with the five types of nodes shown in Table 1. On the other hand, GNP could produce the network if an agent can not carry the heavy load, it will move to the place with light loads. This result clearly shows the effectiveness of GNP.

4.2.2 *Performance of GNP with ADG*

In this section, we compare the performance of GNP with ADG against that of GNP using the heterogeneous and homogeneous model. GNP with the homogeneous model is the same algorithm we have already examined in Subsection 4.2.1. In the heterogeneous model, every agent has its own network for its action control rule.

Fig. 13 shows that the average results obtained by these three models. We can see that GNP with ADG could obtain the best fitness faster than the other two models. As for GNP with the heterogeneous model, it could obtain better results than the homogeneous model at the final generation, but the convergence speed was slower than the homogeneous model. This deterioration of the heterogeneous model can be also seen in Table 4. Table 4 shows the statistical result of 100 trials at the final generation by the three models. We can see in this table that GNP using the ADG model obtained the highest fitness value in all the trial. On the other hand, GNP using the heterogeneous model could not find the highest. Furthermore, the best result obtained by the heterogeneous model was worse than that obtained by the homogeneous model while the average of the heterogeneous model is higher than the homogeneous. These results clearly show that the difficulty of the heterogeneous model due to its large solution space.

Fig. 14 shows the number of classes in GNP with ADG. We can see that the number of classes is converged to two by GNP with ADG.

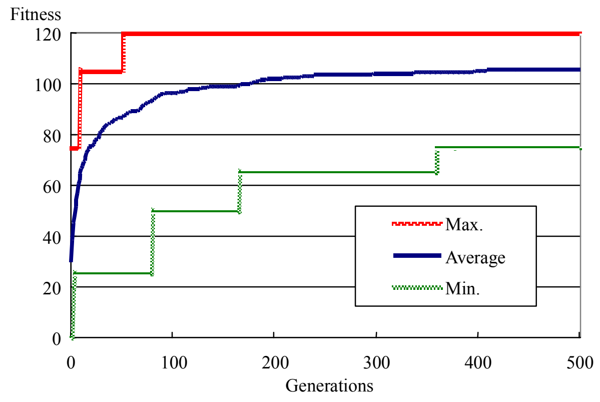

Figure 9. Maximum, minimum and average performance over 100 trials by GNP.

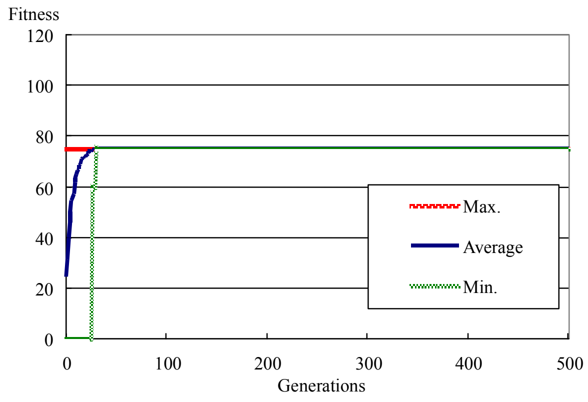

Figure 10. Maximum, minimum and average performance over 100 trials by GP.

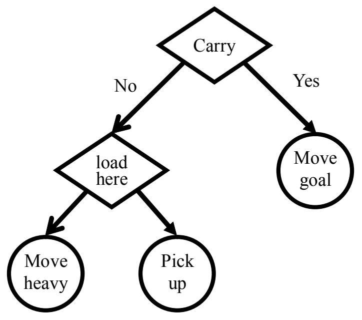

Figure 11. The best tree obtained by GP.

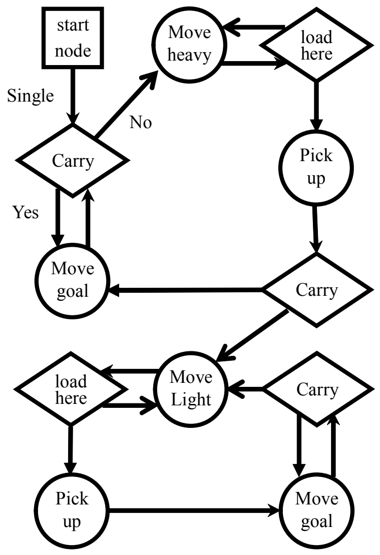

Figure 12. The best network obtained by GNP.

Table 4. Results of the ADG and the heterogeneous model (Ability-Based Role Assignment Problem).

Fitness	ADG	Hetero	Homo
Average	120.0	110.9	105.8
Standard Deviation	0.00	3.50	7.19
Max	120	118	120
Min	120	99	75

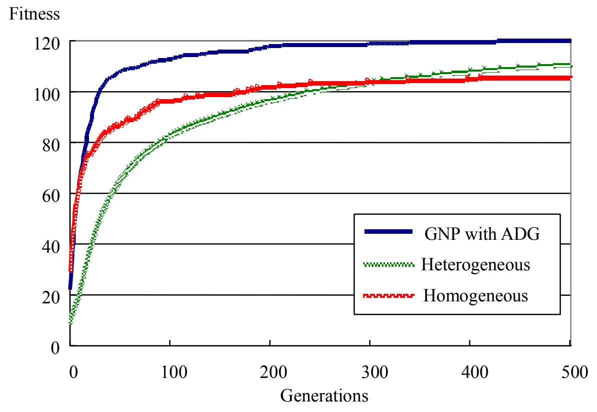

Figure 13. Average performance over 100 trials by GNP with ADG, Heterogeneous, and Homogeneous models (Ability-Based Role Assignment Problem).

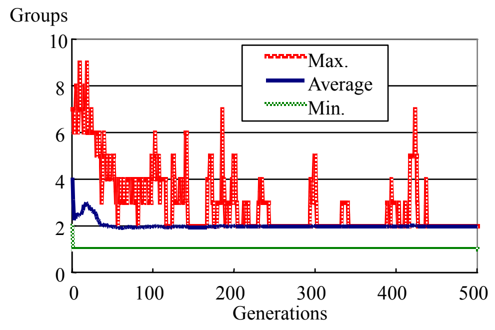

Figure 14. The number of classes in GNP with ADG.

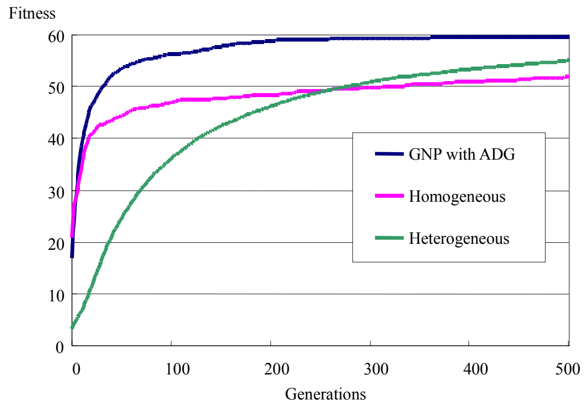

Figure 15. Average performance over 100 trials by GNP with ADG, Heterogeneous, and Homogeneous models (General Role Assignment Problem).

Table 5. Results of the ADG and the heterogeneous model (General Role Assignment Problem).

Fitness	ADG	Hetero	Homo
Average	59.60	55.03	51.80
Standard Deviation	2.81	1.84	6.23
Max	60	59	60
Min	40	50	40

4.3 General Role Assignment Problem

We also apply the GNP with ADG to a different settings of the load transportation problem: General Role Assignment Problem. In this problem settings, we consider only the light loads, and every agent has the same ability to carry a load. We place 30 light loads in each of two load positions. Therefore there are 60 light loads in total, so that the highest fitness value becomes 60 in this problem. Since each agent can go and return three times within the limited duration, 20 agents should be divided into two classes to attain the highest score.

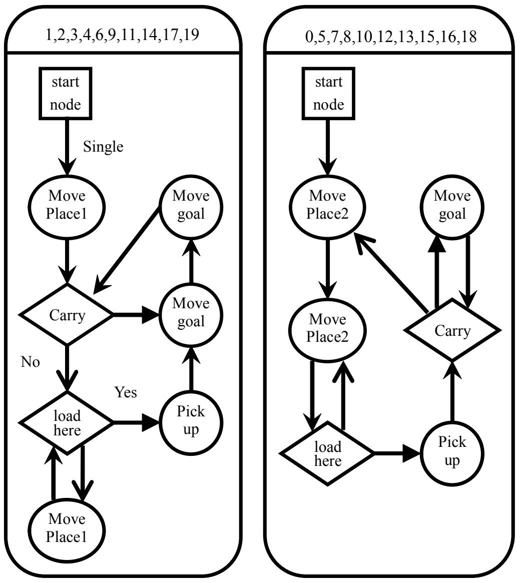

Figure 16. An individual with the best fitness for General Role Assignment Problem.

Fig. 15 shows the average results obtained by GNP with ADG model, the heterogeneous model, and the homogeneous model. From this figure, we can see that the similar results to Fig. 13. Table 5 shows the results obtained at the final generation. From these figure and table, we can see that GNP with ADG could not find the highest fitness value in all the trials. However, the performance of GNP with ADG was the best comparing to the other two models.

Fig. 16 shows an example of the individual with the best fitness value for the General Role Assignment Problem. In this individual, we can see that 20 agents are divided into two classes (The head part of each class indicates the IDs of agents which are assigned to that class). And agents in one class move to the first load place, and the other move to the other load place. From this figure, we can see that the 20 agents could develop their work sharing in order to attain the best score.

5. CONCLUSION AND FUTURE WORKS

In this paper, we examine the performance of the GNP architecture with the ADG model. By computer simulations, we clearly show the better representation ability of GNP than that of GP. We also showed that GNP with the ADG model could find appropriate roles of agents according to their ability by the load transportation problems. Through the computer simulations, we could see the difficulty of the heterogeneous model. That model allows to generate complicated systems but it may be time consuming because the search space of that problem is so huge.

As for further research topics, we need to investigate a way to reduce the redundant nodes in the networks developed in GNP. As shown in Fig. 16, the network obtained for each class was not optimized with respect to the number of nodes. We can continue to work to refine the structure of obtained networks.

6. ACKNOWLEDGMENTS

This work was supported by Ministry of Education, Culture, Sports, Science and Technology, “Open Research Center” Project (2003).

7. REFERENCES

[1] Fogel, L. J., Owens, A. J., and Walsh, M. J. *Artificial Intelligence through Simulated Evolution*, Wiley, 1996.

[2] Hara, A., and Nagao, T. ADG: Automatically Defined Groups for multi-agent cooperation, In *Proc. of 2nd Japan-Australia Joint Workshop on Intelligent and Evolutionary Systems*, 1998, 91-98.

[3] Hara, A., and Nagao, T. Emergence of cooperative behavior using ADG; Automatically defined groups, In *Proc. of 1999 GECCO,* 1999, 1039-1046.

[4] Haynes, T., and Sen, S. Crossover operation for evolving a team, In *Proc. of Genetic Programming 1997*, 1997, 162-167.

[5] Hirasawa, K., Okubo, M., Katagiri, H., Hu, J., and Murata, J. Comparison between genetic network programming and genetic programming using evolution of ant’s behaviors, *Trans. on Institute of Electrical Engineers of Japan*, Vol. 121-C, No. 6, 2001, 1001-1009 (in Japanese).

[6] Iba, H. Multiple agent learning for a robot navigation task by genetic programming, In *Proc. of Genetic Programming 1997*, 1997, 195-200.

[7] Katagiri, H., Hirasawa, K., Hu, J., and Murata, J. Network structure oriented evolutionary model -Genetic Network Programming- and its comparison with Genetic Programming, In *Proc. of GECCO 2001*, 2001, page 219.

[8] Katagiri, H., Hirasawa, K., Hu, J., and Murata, J. Variable size genetic network programming, *Trans. on Institute of Electrical Engineers of Japan*, Vol. 123, No. 1, 2003, 57-66 (in Japanese).

[9] Koza, J. R. *Genetic Programming On the Programming of Computers by means of Natural Selection*, MIT Press, 1992.

[10] Luke, S., and Spector, L. Evolving teamwork and coordination with genetic programming, In *Proc. of Genetic Programming 1996*, 1996, 141-149.

[11] Murata, T., and Nakamura, T. Multi-Agent Cooperation Using Genetic Network Programming with Automatically Defined Groups, In *Proc. of GECCO 2004*, Vol. 2, 2004, 712-714.

[12] Murata, T., and Nakamura, T. Developing Cooperation of Multiple Agents Using Genetic Network Programming with Automatically Defined Groups, In *Proc. of LBP in GECCO 2004*, 2004, 12 pages (CD-ROM).

[13] Poli, R. Evolution of graph-like programs with parallel distributed genetic programming, In *Proc. of 7th ICGA*, pp. 346-353.

[14] Pollack, M.E., and Ringuette, M. Introducing the tile world: Experimentally evaluating agent architectures, In *Proc. of 8th National Conf. on Artificial Intelligence*, 1990, 183-189.