Heterogeneous Multi Agents Learning using Improved Genetic Network Programming

Hirotaka Itoh
Nagoya Institute of Technology
Gokiso-cho, Showa-ku,
Nagoya, Japan
e-mail: ht-itoh@nitech.ac.jp

Naoki Ikeda
Nagoya University
Furo-cho, Tikusa-ku,
Nagoya , Japan
e-mail: icenine84@hotmail.com

Kenji Funahashi
Nagoaya Institute of Technology
Gokiso-cho, Showa-ku,
Nagoya, Japan
e-mail: kenji@nitech.ac.jp

Abstract— **The heterogeneous multi agent system is a system that cooperates intensively in one place by two or more agents and achieves the task. There is Genetic Network Programming as an automatic generation technique of the heterogeneous multi agent system. In this paper, the authors proposes IGNP and GNPIAM as automatic generation technique of the heterogeneous multi agent system. And the authors propose both co-evolution method and non-co-evolution method as agent's evolution method and compare the performance of two methods.**

I. INTRODUCTION

Recently, the multi agent system is widely researched. The multi agent system is a system that solves the problem by using two or more agents. An individual agent perceives the environment and behaves to archive the task. The environment changes from a certain agent's behavior. The change of the environment decides another agent's behavior. In the multi agent system, the entire system is not collectively managed. The individual agent cooperates with other agents. There is an advantage that the cost of the entire system is lower by such distributed management. The individual agent has two abilities of perception and operation. The perception ability is an ability to recognize the environment, and the operation ability is an ability to influence the environment. The action when the task is given to the agent is decided by the procedure of perception and operation. The action of the agent who achieves a simple task can be designed by the hand work. On the other hand, it costs a lot of time and the cost for the complex task. In addition, the design of the behavior of two or more agents becomes a more complex problem, and it is difficult in the hand work.

The Genetic Network Programming (GNP)[1],[2] exists as a technique for generating the agent's behavior automatically. GNP is an improvement technique of Genetic Programming (GP)[1]. GNP has a network structure and can make the agent who can correspond to a dynamic environment that changes by the agent's behavior. The evolution algorithm of GNP is the same as the one of Genetic Algorithm (GA). The initial convergence and poor local search capability are pointed out as the defect of GA. Evolutionary adaptation algorithms focusing on the workings of the immune system have been devised to improve the defect of GA. These are Immune Algorithm (IA)[3]and Genetic Algorithm with Immune Adjustment Mechanism (GAIAM)[4]. Then, the author proposed Immune evolved Genetic Network Programming (IGNP)[5] that had used IA as an evolution algorithm of GNP in the past. In this paper, authors propose Genetic Network Programming with Immune Adjustment Mechanism (GNPIAM) that uses GAIAM.

In the research of the past on the multi agent system that used GNP[6],[7], the one to raise efficiency by using two or more agents for the task that was able to be achieved by a single agent was the main. However, the system that assumes the purpose of this research is not the one that raises efficiency as two or more agents and is the one that achieves the task as two or more agents cooperate and archive the task. The targeting task can't be achieved by single agent and can be archived by multi agents.

The heterogeneous multi agents system is the system by which an individual agent individually has the operation program. Authors propose both co-evolution technique and non-co-evolution technique as agent's evolution technique. Co-evolution technique is the method to which the evaluation value corresponding to the task contribution level is given to each agent, and each agent evolves individually. On the other hand, non-co-evolution technique is the method to which the whole evolves according to the evaluation value to which is given to the entire system.

In this paper, authors compare both co-evolution method and non-co-evolution method using GNP, IGNP and GNPIAM.

Authors describe GNP, IGNP and GNPIAM in chapter II, III and IV, respectively. And propose co-evolution technique and non-co-evololution technique as agent's evolution technique in chapter V. The proposed technique are experimented in chapter VI. In chapter VII, the conclusion is described.

II. GNP: GENETIC NETWORK PROGRAMMING

A. The Structure of GNP

GNP[1],[2] is one of the automatic programming techniques

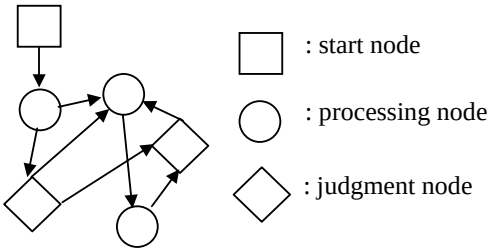

Fig.1 The structure of the GNP

based on GP[2]. GP has the tree structure, it returns to the root in GP after it changes from the root to the terminal. Therefore, GP is suitable for study to a static environment. On the other hand, GNP uses the network structure. Because the state transition is done in the sub network, GNP is suitable for the study of a dynamic environment. Fig.1 shows the structure of GNP. GNP has start node, judgment node and processing node. The start node is displayed by a square in Fig.1 is a point where processing begins. The processing node is displayed by an open circle in Fig.1 is agent's operation part. The judgment node is displayed by a diamond in Fig.1 is agent's sensor.

The network as the individual is composed of many nodes. Fig.2 shows the representation of the node. Each node consists node gene and connection gene.

In the node gene, NT_i is the type of the node, ID_i is the function number of the node and d_i is the delay time takes to process the function of node *i*. If NT_i =0, the node type is the processing node. If NT_i =1, it is the judgment node. The network has the node library. The node library is shown in Fig.3. If the number of processing is *x* and the number of judgment is *y*, *x* processing functions and *y* judgment functions are defined in the node library.

In the connection gene, C_{ij} is the *j*-th connection from the node *i*. N_i^k shows the number of connections from the node *i*. When the node type is the processing node, N_i^k=1, because the processing node has only one connection. When the node type is the judgment node, N_i^k>=2, because the judgment node has several connections according to its condition. d_{ij} is the transition delay time that passes C_{ij} from node *i*.

The start node is defined node 0.

B. Genetic operation of GNP

The genetic operation of GNP is crossover and mutation. Fig.4 shows the crossover. As shown in Fig.4, the replacement of the nodes and changes the connections are done by replacing the specific area. Fig.5 shows the mutation. On the mutation, the connection between nodes is changed.

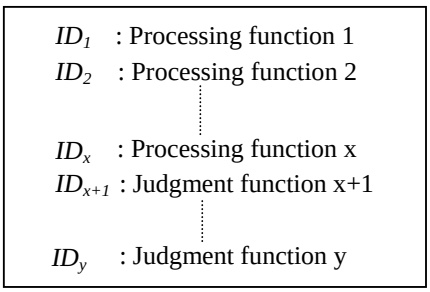

Fig. 3 The node library

C. Algorithm of GNP

In this section authors describe the evolutionary algorithm of GNP.

Step.1 Generation of an initial group of individuals

N individuals are generated for the initial group of individuals.

Step 2 Calculation of fitness

Fitness for each individual is calculated. The fitness is the evaluated value for the solution. And an elitist individual with the best fitness value in the population is found.

Step 3 Selection

Select parents for crossover by the tournament selection from *N* individuals.

Step 4 Crossover and mutation

Apply the crossover based on crossover probability *Pc* to the selected parents in Step 3. Then, these undergo mutation based on mutation probability *Pm*.

Step 5 Replacement

Replace the newly generated population with the previous population. But the elitist individual found in Step 2 or Step 6 is preserved.

Step 6 Calculation of fitness

Fitness for each individual is calculated and an elitist individual with the best fitness value in the population is found.

Step 7 Repetition of Steps 3 to 6 for a determined number of generations.

III. IGNP: IMMUNE EVOLVED GENETIC NETWORK PROGRAMMING

A. The outline of IGNP

GNP uses GA as the evolutionary algorithm, but GA has the defects. IGNP[5] uses IA[3] as the evolutionary algorithm. IA is

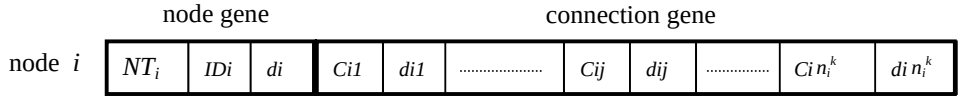

Fig. 2 The representation of the node

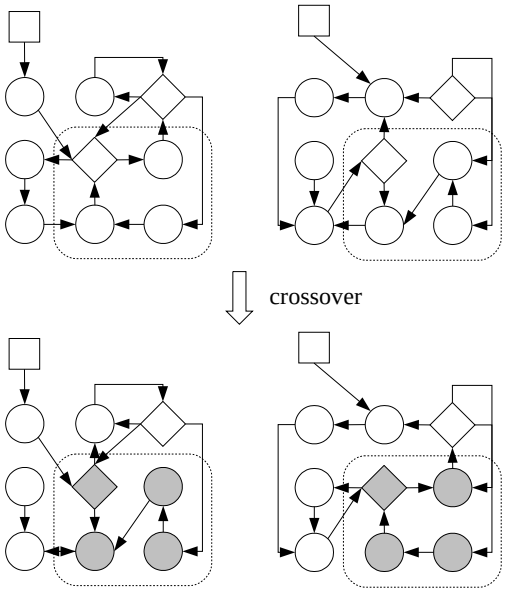

Fig.4 crossover

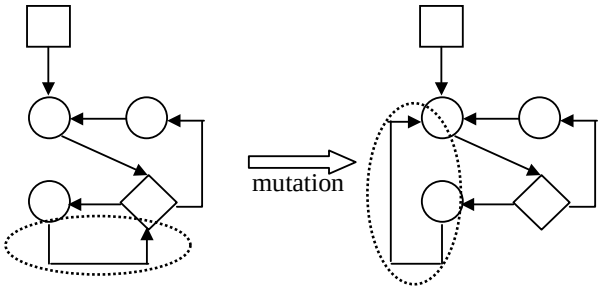

Fig.5 mutation

the evolutionary adaptation algorithms focusing on the workings of the immune system have been devised to improve the defect of GA.

B. *Algorithm of IGNP*

Step 1 Generation of an initial group of antibodies

N antibodies are generated for the initial group of antibodies. The antibodies are the same as individuals for the GA and are solutions to an optimization problem.

Step 2 Calculation of affinities

Affinity ax_i (i= 1,…, N) for antigens are calculated. ax_i is set in accordance with the problem used. Affinity for the antigen is the same as fitness for GNP.

Step 3 Differentiation into memory cells

For details, see reference [3].

Step 4 Calculation of expected values

The expected value e_i (i= 1,…, N) for antibodies surviving into the next generation is calculated.

$$e_i = \frac{ax_i}{C_i} \qquad (1)$$

C_i is the density of antibody i (i= 1,…, N).

$$C_i = \frac{1}{N}\sum_{j=1}^{N} ay_{i,j} \qquad (2)$$

$ay_{i,j}$ is the similarity of antibodies i and j (i = 1,…,N, j =1,…,N) and is set appropriately depending on the problem used. N /2 antibodies with low expected values are eliminated. However, the top 10% of antibodies with high affinities for antigens are excluded from elimination.

Step 5 Antibody production

New antibodies are produced in place of N / 2 antibodies eliminated in Step 4. New antibodies are produced by randomly determining their genes.

Step 6 Crossover and mutation

Antibodies are randomly selected, duplication permitted, from N antibodies and undergo crossover based on crossover probability Pc, producing N /2 antibodies. Then, these undergo mutation based on mutation probability Pm, and affinity for the antigen is calculated.

Step 7 Repetition of Steps 3 to 6 for a determined number of generations.

Because of Step 5, the IA avoided narrowing the search to local solutions.

IV. GNPIAM: GENETIC NETWORK PROGRAMMING WITH IMMUNE ADJUSTMENT MECHANISM

A. *The outline of IGNPIAM*

GNPIAM uses GAIAM[4] as the evolutionary algorithm. GAIAM is an algorithm to take two features of immune system to GA.

There are various antibodies present in the body. As antigens invade the body from the outside, antibodies corresponding to these antigens proliferate and eliminate the antigens. The immune system has the following 2 features:

[Feature 1] Capacity to adapt to mutations in antigens

Preparing matching antibodies for every antigen beforehand is difficult. When antibodies matching an antigen do not exist, the genes of the best matching antibodies respond by mutating. Antibodies adapted to the antigen are produced by repeated mutations of these genes.

[Feature2] Mechanism to adjust antibodies via antibodies

Proliferation of antibodies matching a given antigen is not unlimited; rather, antibodies recognize one another based on their structure. When a given antibody proliferates, inhibiting antibodies recognizing that antibody as an antigen also proliferate and respond. Overall, balance is maintained.

B. *Algorithm of GAIAM*

Step 1 Generation of an initial group of antibodies

Same as Step 1 in the IGNP

Step 2 Calculation of affinities

Same as Step 2 in the IGNP

Step 3 Calculation of expected values

Same as Step 4 in the IGNP

Step 4 Antibody production

New antibodies are produced in place of N /2 antibodies eliminated in (3). N /2 are selected from surviving antibodies in accordance with expected values. Then, the N/2 antibodies selected are mutated, after which affinities for antigen are calculated.

Step 5 Crossover and mutation

Same as Step 6 in the IGNP

Step 6 Adjustment of antibodies

With respect to each antibody i of the N /2 antibodies newly produced in Step 5, antibody j with the greatest affinity for i are sought from among existing N antibodies. Of antibodies i and j, those with a high affinity for the antigen survives to the next generation while those with a low affinity are removed.

Step 7 Repetition of Steps 3 to 6 for a determined number of generations.

Step 4 models [Feature 1] of the immune system and Step 6 models [Feature 2] of the immune system. With GNPIAM, antibodies with a high affinity for the antigen and low density tend to remain to maintain diversity. Moreover, such antibodies proliferate with GNPIAM; effective antibodies with a high affinity for the antigen are produced by mutation. That is, local search capability is improved. Moreover, narrowing of the search range is avoided by Step 6. Thus, narrowing of the search to the vicinity of a single local solution is avoided.

V. LEARNING OF THE HETEROGENEOUS MULTI AGENTS SYSTEM

A. Heterogeneous multi agents system

In the multi agents system, the entire system is composed by using two or more agents who move individually and autonomous. If the agent can cooperate mutually, the big task that won't be done in a single agent can be achieved. The multi agents system is divided into the heterogeneous multi agents[6] and homogeneous multi agents. In the heterogeneous multi agents, an individual agent has an individual operation program. On the other hand, all agents in the system have the same operation program in the homogeneous multi agents. The multi agents system that this research targets is the heterogeneous multi agents. In the text, the learning method of the heterogeneous multi agents is examined by using GNP, IGNP and GAIAM. The author proposes both co-evolution technique and non-co-evolution technique as the learning method of the heterogeneous multi agents.

B. Co-evolution heterogeneous multi agents

In the heterogeneous multi agents, an individual agent has a directed graph individually. The directed graph here is the agent's operation program, and it is shown in Fig.1. By giving the evaluation value to an individual agent according to the contribution level to the task, the agent is individually evolved. It is thought that the agent with low contribution level can be studied to contribute to the task by giving the evaluation value to the individual agent. The co-evolution here is to mean, that is, the co-evolution in the agent group, and a certain agent's evolution influences another agent's evolution. Each agent learns own behavior by being influenced by other agents' behavior and co-evolving.

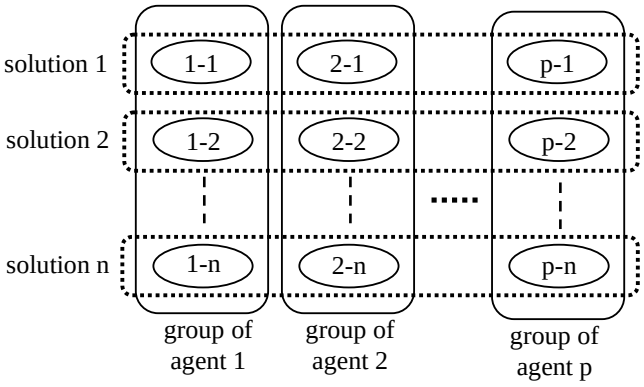

Fig.6. The agent's grouping

The process of co-evolution is as follows.

Step.1 The initial group is generated.

Step.2 The initial group is divided into groups of the same number as agents, one individual is selected from each group, and the agent group is made (Fig.6). This agent group is not changed by the evolution processing.

Step.3 The task is executed in the agent group.

Step.4 The fitness of the individual agent is calculated according to the contribution level to the task.

Step.5 Each agent group does the evolution processing.

Step.6 Repetition of Steps 3 to 6 until it meets the end requirement.

C. Non-co-evolution heterogeneous multi agents

In non-co-evolution heterogeneous multi agents, but the individual agent individually has a directed graph, but the agent doesn't have the distinction according to the evaluation value. As shown in Fig.7, one directed graph is divided into two or more sub directed graphs. The number of sub graphs is the same as the number of agents. There is the start node in each sub graph, and the connection is not between sub graphs, and is sub graphs are independent. It is possible to think this heterogeneous multi agent to be one directed graph that includes sub graphs. One directed graph becomes one of the solution candidates of the entire multi

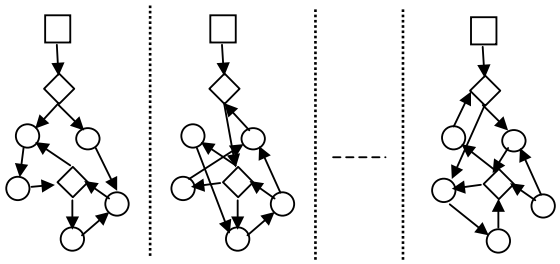

Fig.7 agent group

agents system, and the evaluation value is given to the solution. Therefore, the evolution processing can be done just like usual GNP.

VI. Experiment of Proposed Method

A. Tile World

Authors experimented by using the tile world to evaluate the proposed methods. The tile world is known well as an example of simulating a dynamic environment. As shown in Fig.8, the tile world is two-dimension lattice plane where agent, tile, floor, hole, and obstacle are arranged. An individual division in the lattice is called a cell, and the agent can move by one cell at one unit time. In Fig.8, the agent, the tile, the hole and the obstacle are denoted by "A", "T2", an open circle and a full square, respectively. A blank cell in the tile world is a floor. In this experiment, weights are set to the tile for the cooperation of the agents. In Fig.8, the number after "T" shows the weights, and "T2" is a tile with two weights. Two agents are necessary to carry the tile with two weights. In this tile world, the agents who corporate with others and drop the tile to the hole are made. Two agents gather in the cell of the tile, two agents grip the tile, and these move to the cell of the hole together. The tile falls into the hole in two agents' releasing the tile.

B. Experiment Environment

The experiment was executed by initial arrangement of Fig.8. There are one tile with two weights, one hole and two agents. Two agents' initial positions are the same.

Table.1 shows the nodes used to the experiment. In Table.1, *NT* is the node type, *d* is delay time, and *Nn* is the number of nodes. On the judgment nodes, there are condition branching. Table.2 shows the condition branching of the judgment nodes. The delay time between nodes is assumed to be 0.

The evaluation value of the agent in the tile world, that is, the fitness is calculated by the following equation.

$$fitness = 45 \times DropTile \times TileWeight$$
$$+ 5 \times \frac{InitialDist - LastDist}{InitialDist} \times TileWeight$$
$$+ 10 \times GrabTile + 0.01 \times MoveArea$$

DropTile is number of tiles that agent dropped. *TileWeight* is the weight of the tile. *InitialDist* is the shortest distance between the tile and the hole in the initial state. *LastDist* is the shortest distance between the tile and the hole after time limits passes. *GrabTile* is whether the tile is gripped after the time limit or not? *MoveArea* is the number of cells that the agent moved. 45,5,10 and 0.01 are the weight of each paragraph. The agents could drop the tile to the hole if the fitness exceeds 100. The agents were able to grip the tile if the fitness exceeds 10. Agents could take the tile to the hole if the fitness exceeds 20. Table.3 shows parameters of the experiment.

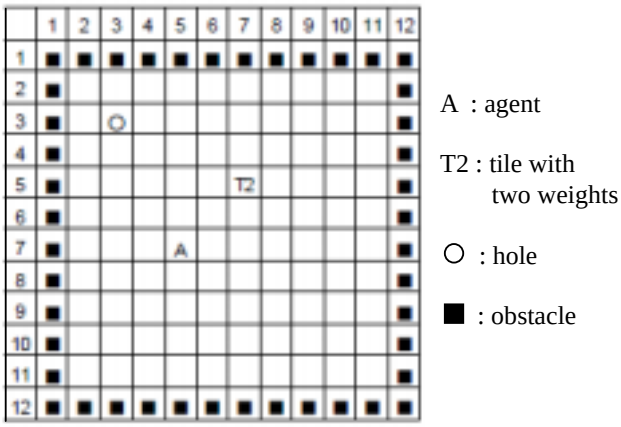

Fig.8 Tile world

Table.1 The nodes used to the experiment

node name	meaning	*NT*	*d*	*Nn*
GO_FORWARD	Go ahead by one cell.	Processing	1	2
GRAB	Grab the tile	Processing	1	3
RELEASE	Release the title	Processing	1	3
TURN_LEFT	Turn left	Processing	1	2
TURN_RIGHT	Turn right	Processing	1	2
NOP	Nothing is done	Processing	1	2
HAVE_TILE	Does the agent has the title?	Judgment	2	1
FIND_TILE	Where is the tile?	Judgment	6	2
FIND_HOLE	Where is the hole?	Judgment	5	2
FIND_AGENT	Where is other agent?	Judgment	5	1
TRANS_FAIL	Did the previous action fail?	Judgment	2	2

Table.2 The condition branching of the judgment nodes

node name	*Nd*	answer
HAVE_TILE	2	Yes, No
FIND_TILE	5	upper, lower, left, right, prevent place
FIND_HOLE	5	upper, lower, left, right, prevent place
FIND_AGENT	5	upper, lower, left, right, prevent place
TRANS_FAIL	2	Yes, No

Table.3 Parameters of experiment

generation number	500
individual number	100
limits time	2000
mutation probability	0.01
crossover probability	0.65
trial number	10

C. Experiment results

The learning result of the co-evolution heterogeneous multi agents using GNP, IGNP and GNPIAM is shown in Fig.9. And, each progress of ten trials is shown in Table.4 how the learning is advanced.

The learning result of the non-co-evolution heterogeneous multi agents using GNP, IGNP and GNPIAM is shown in Fig.10. And, each progress of ten trials is shown in Table.5 how the learning is advanced.

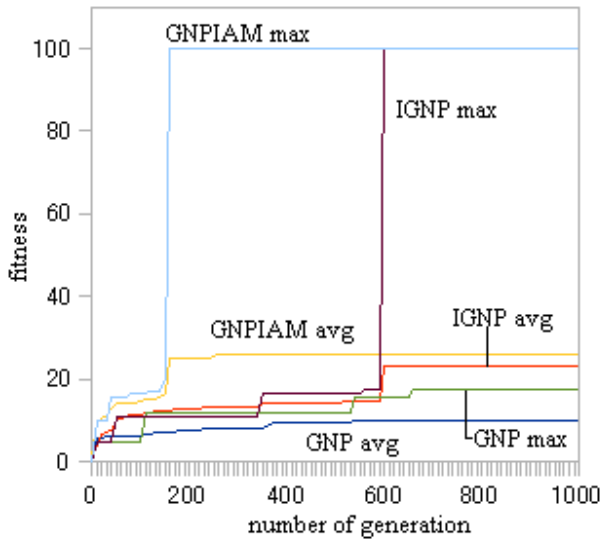

Fig. 9 The Learning result of co-evolution method

Table.4 Progress of the learning

	GNP	IGNP	GNPIAM
All agents grip the tile.	7	10	10
Agents move gripping the tile.	3	8	10
Agents reache the cell of the hole.	0	2	5
Agents drop the tile into the hole	0	1	1

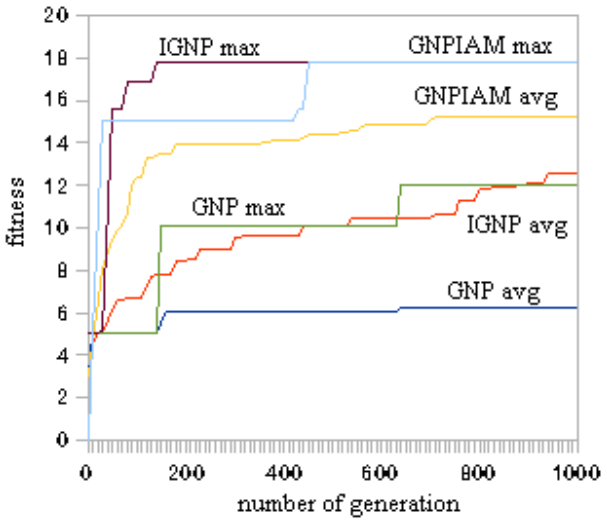

Fig. 10 The Learning result of non-co-evolution method

Table.5 Progress of the learning

	GNP	IGNP	GNPIAM
All agents grip the tile.	3	9	10
Agents move gripping the tile.	1	6	9
Agents reache the cell of the hole.	0	0	0
Agents drop the tile into the hole	0	0	0

In Fig.9 and Fig.10 "max" shows the transition of the evaluation value of times that obtain the maximum fitness of ten trials. "avg" is an average of the fitness in ten trials. When considering it from Fig.9 and Fig.11, in the both methods of co-evolution and non-co-evolution, IGNP and GNPIAM are more excellent than GNP. GNP is thought that it settles to the local solution, and the evaluation value did not go up. IGNP and GNPIAM is thought that these obtained better fitness because these search area is wide. IGNP and GNPIAM are compared by using Table.4 and Table.5. It is understood that the learning of GNPIAM is better than that of IGNP from Table.4 and Table.5. This is because the local search ability of GNPIAM is higher (describe in Chapter 3).

When co-evolution and non-co-evolution are compared, co-evolution is more successful. In the non-co-evolution, the fitness of a certain agent becomes the fitness of the entire agent. Therefore, evolution settles if the agent with high fitness appears.

VII. Conclusion

The heterogeneous multi agents was learned by using GNP, IGNP and GNPIAM. The targeting task can't be achieved by single agent and can be archived by multi agents. As a result, the learning of GNPIAM is the best. GNPIAM has the high local search ability and can search for wide area. The authors proposed the co-evolution and the non-co-evolution as the learning method of the heterogeneous multi agents. The co-evolution method is better than the non-co-evolution method.

The future work is that the method is improved and the agent who achieves the task by the higher percentage is made because of the task cannot be achieved in all the trials.

References

[1] K.Hirasawa, M.Okubo, H.Katagiri, J.Hu and J.Murata, "Comparison between Genetic Network Programming and Genetic Programming Using Evolution of Ant's Behaviors (in Japanese)", *The Transactions of the Institute of Electrical Engineers of Japan*, Vol.121-C, No.6, pp.1001-1009, 2001

[2] T.Murata, T.Nakamura, "Genetic Network Programming with Automatically Defined Groups for Assigning Proper Roles to Multiple Agents", Proceedings of Genetic and Evolutionary Computation (GECCO 2005), pp.25-29, pp.1705-1712, 2005

[3] K.Mori, M.Tsukiyama and T.Fukuda, "Immune Algorithm with Searching Diversity and its Application to Resource Allocation Problem", *The Transactions of the Institute of Electrical Engineers of Japan*, Vo1.113-C No.10, pp. 872-878, 1997

[4] H.Itoh, "Genetic Algorithm with Immune Adjustment Mechanism", *Proceedings of the 3rd IASTED Conference Computational Intelligence*, pp.79-84,2007

[5] H.Itoh, T.Mase, Y.Iwahori, "Agent Learning using Immune Evolved Genetic Network Programming", *Transactions of the Institute of Electrical Engineers of Japan* , Vol.125-C, No.4, pp.537-5442005

[6] K.Hirasawa, M.Okubo, J.Hu, J.Murata and Y.Matsuya, "Co-evolution of Heteo-Multiagent Systems Using Genetic Network Programming", *Transactions of the Institute of Electrical Engineers of Japan,*, Vol.123-C, No.3, pp.544-551, 2003.

[7] T.Eguchi, K.Hirasawa and T.Furuzuki, "Construction of Symbiotic Evolutional Model in Multiagent Systems", *IPSJ TOM* , Vol.45, No.SIG2, pp144-156, 2004