Online Learning of Genetic Network Programming (GNP)

Shingo Mabu, Kotaro Hirasawa, Jinglu Hu and Junichi Murata

Graduate School of Information Science and Electrical Engineering, Kyushu University
6-10-1, Hakozaki, Higashi-ku, Fukuoka, 812-8581, Japan
E-mail: {mabu, hirasawa}@cig.ees.kyushu-u.ac.jp

Abstract - A new evolutionary computation method named Genetic Network Programming (GNP) was proposed recently. In this paper, an online learning method for GNP is proposed. This method uses Q learning to improve its state transition rules so that it can make GNP adapt to the dynamic environments efficiently.

I. INTRODUCTION

Evolutional processes of organisms are very complicated and sophisticated. They are based on the mechanisms such as learning, selection and evolution, so that they can harmonize well with environments.

"Genetic Algorithm (GA)[1]" and "Genetic Programming (GP)[2]" are typical methods that are based on the evolutional processes of organisms. Because the conventional control theory should obey the rules that are predefined in advance and it cannot be adapted to the dynamical environment rapidly, the evolutionary computation overcoming the above problems attracts the attention.

GA and GP which have been mainly applied to optimization problems represent solutions as a string and a tree structure, respectively and evolve them. GP was devised later in order to expand the representation ability of GA and to solve more complex problems. But, GP might be difficult to search for a solution because of the bloat of its tree structure, which expands the depth of the tree unnecessarily although it is sometimes useful for expanding a search space and find a solution.

Recently, a new evolutionary computation method named "Genetic Network Programming (GNP)" was proposed[3]-[4]. GNP that is an expansion of GA and GP represents solutions as a network structure. Because GNP can memorize the past action sequences in the network flow and can deal with Partially Observable Markov Decision Process (POMDP) well, GNP was applied to complicated agent systems involving uncertainty such as decision-making problems. But, conventional GNP are based on "offline learning", that is, after GNP is carried out to some extent, it is evaluated and evolved according to rewards given by an environment. However, the adaptation to dynamical environments might be difficult because if the changes of an environment occurred, offline learning must do many trials to evaluate GNP and evolve it again and again, therefore, it cannot keep up with the changes of environments quickly.

In this paper, Q learning[5] which is one of the famous online learning method in reinforcement learning field is introduced for the online learning of GNP. GNP changes its node transition rules considering the rewards given one after another, so that it can change its solution (behavior sequences) immediately after environmental changes that cause a bad result. The reason for applying Q learning to online learning of GNP is described in section 2.

In this paper, Prisoner's dilemma game is used for simulations and the performance of online learning is studied. Prisoner's dilemma game needs two players and they compete with each other to get high scores. First, GNP having a game strategy competed with the famous strategies of Prisoner's dilemma game and showed the good performances (scores). Then, two GNPs competed with each other and showed the online adjustment on their strategies considering the opponent's strategy in order to get higher scores.

This paper is organized as follows. In the next section, the details of Genetic Network Programming is described. Section 3 explains Prisoner's dilemma game and shows the results of the simulations. Section 4 is devoted to conclusions.

II. GENETIC NETWORK PROGRAMMING (GNP)

In this section, Genetic Network programming is explained in detail. GNP is an expansion of GP in terms of gene structures. The original motivation of developing GNP is based on the more general representation ability of graphs than that of trees.

A. Basic structure of GNP

First, GP is explained in order to compare it with GNP. Fig.1 shows a basic structure of GP. GP can be used as a decision making tree when non-terminal nodes are *if-then* type functions and all terminal nodes are some concrete action functions. A tree is executed from the root node to a certain terminal node in each iteration, therefore, it might fall into the deadlocks because the behaviors of agents made by GP are determined only by the environments at the current time. Furthermore, GP tree might cause the severe bloat that makes search for solutions difficult because of the unnecessary expansion of depth.

Next, the characteristics and abilities of GNP are explained using Fig.2 which shows the basic structure of GNP. Now, it is supposed that agent behavior sequences are created by GNP.

(1) GNP has a number of Judgement nodes and Processing nodes connected by directed links with each other like

0-7803-7282-4/02/$10.00 ©2002 IEEE

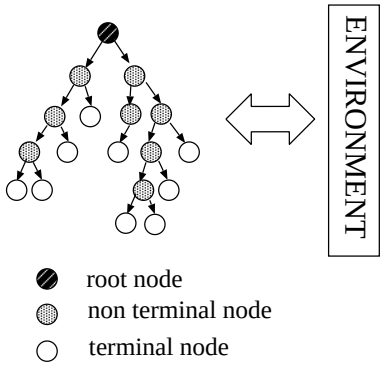

Fig.1 basic structure of GP

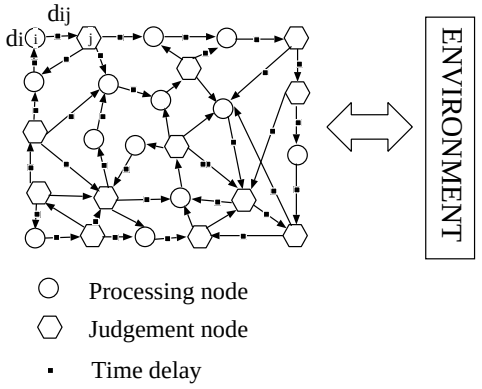

Fig.2 basic structure of GNP

networks. Judgement nodes, that are *if-then* type decision functions or conditional branch decision functions, return judgement results for assigned inputs and determine the next node GNP should take. Processing node determines an action/processing an agent should take. Contrary to judgement nodes, processing nodes have no conditional branch. The GNP we used never causes bloat because of the predefined number of nodes, although GNP can evolve phenotype/genetypes of variable length.

(2) GNP is booted up from the start node that is predefined in advance arbitrarily and there are no terminal nodes. After the start node, the next node is determined according to the connections of the nodes and judging results of judgement nodes. In this way, GNP system is carried out according to the network flow without any terminal, so the network structure itself has a memory function of the past actions of an agent. Therefore, the determination of the next node is influenced by the node transitions of the past.

(3) GNP can evolve appropriate connections so that it can judge environments and process functions efficiently. GNP can determine what kind of judgement and processing should be done now by evolving the network structure.

(4) GNP can have time delays. d_i is the time delay GNP spends on judgement or processing and d_{ij} is the one GNP spends on transitions from node i to j. In the real world problems, some time is spent when judging or processing and also transferring from one node to the next node. For example, when a man is walking and there is a puddle before him, he will avoid it. At that time, it takes some time to judge the puddle (d_i of judgement node), to put judgement into action (d_{ij} of transition from judgement to processing) and to avoid the puddle (d_i of processing node). However, d_i and d_{ij} are not used in this paper because the purpose of the simulations using Prisoner's dilemma game is to make and change strategies adapting to the opponent behavior and the time spent for judgement or processing does not need to be considered. Time delay is necessary in the case of practical applications. When various judgements and processes are done, GNP should change its structure considering the time it spends. Time delay is listed in each node gene which will be described later because it is the unique attribute of a node.

There have been developed some graph based evolutional methods such as PADO (Parallel Algorithm Discovery and Orchestration)[6] and EP (Evolutionary Programming)[7]. PADO has both start node and end node in the network and it mainly represents a static program. EP is used for the automatic synthesis of Finite State Machines and in all states, state transitions for all inputs have to be determined, therefore, the structure becomes complicated if the number of nodes increase. On the other hand, GNP uses necessary information only for judging the environments, i.e., it can deal with POMDP. Therefore, GNP could be compact even if the system to solve is large enough.

B. Genotype expression of GNP node

The whole structure of GNP is determined by the combination of the following node genes. A genetic code of node i is represented as Fig.3.

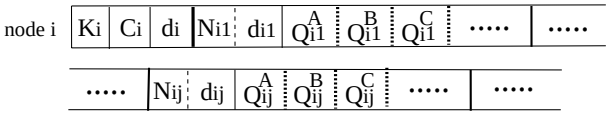

Fig.3 genotype expression of node i

where, K_i : Judgement/Processing classification
0 : Processing node
1 : Judgement node

C_i : content of Judgement/Processing

d_i : time delay spent for judgement/processing

N_{ij} : transition from node i to node j
$N_{ij} = 0$: infeasible (no connection)
$N_{ij} = 1$: feasible

d_{ij} : time delay spent for the transition from node i to j

$Q_{ij}^A, Q_{ij}^B, Q_{ij}^C \ldots$: transition Q values from node i to j

K_i represents the node type, K_i=0 means Processing node, K_i=1 means Judgement node. C_i means the content GNP judges or processes and they are represented as unique numbers. d_i is the time delay spent for judgement or processing. N_{ij} shows whether the transition from node i to j $(1 \leq j \leq n$ (each node has a unique number from 1 to n, respectively when the sum of nodes is n)) is feasible or not. N_{ij}=0 means infeasible, N_{ij}=1 means feasible. d_{ij} means time delay spent for the transition from node i to j. $Q_{ij}^A, Q_{ij}^B, Q_{ij}^C \ldots$ means transition Q values from node i to j. Judgement node has plural judging results. For example, if a result of judgement is "A", GNP uses Q_{ij}^A to select a transition, if the result is "B", it uses Q_{ij}^B. Therefore, the number of $Q_{ij}^A, Q_{ij}^B, Q_{ij}^C, \ldots$ are the same as the number of the results at judgement nodes. On the other hand, processing node has only Q_{ij}^A because there are no conditional branches when processing.

C. Online learning of GNP

Because the conventional learning method for GNP is "offline learning", it is evaluated by the rewards given by the environment after GNP is carried out to some extent. Offline learning for GNP includes "selection", "mutation", and "crossover". Online learning for GNP is based on Q learning and changes its node transition rules considering the rewards given one after another. That is, each node has a probability of plural transitions to the other nodes and transition Q values are calculated by using Q learning. GNP selects the transition according to the Q values. Agents can learn and adapt to changes of the environments through this processes.

1) Q learning (An off-policy TD control algorithm) : Q learning calculates Q values which are functions of the state s and the action a. Q values means the sum of the rewards an agent gets in the future, and the update processes of them are implemented as follows. When an agent selects an action a_t in state s_t at time t, the reward r_t is given and the state is changed from s_t to s_{t+1}. As a result, $Q(s_t, a_t)$ is updated as the following equation.

$$\begin{aligned} &Q(s_t, a_t) = Q(s_t, a_t) \\ &+ \alpha\Big[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\Big] \end{aligned} \tag{1}$$

If the step size parameter α decreases according to a certain schedule, Q values converge on an optimal values after enough trials. The action selection which has the maximum Q value becomes the optimal policy. γ is a discount rate which shows how long an agent considers the future rewards.

If an agent continues to select the actions having maximum Q value when the learning is not enough, the policy improvement of taking the action is not implemented even though there are still better action selections. One of the method for overcoming the above problem is "ε-greedy". This method makes the agent select the random action by the probability of ε, or select the action having maximum Q value by the probability of 1-ε. ε-greedy is a general method for keeping a balance between exploitation of experience and exploration. In this paper, ε-greedy is adopted for the action selections.

2) Relation between GNP and Q learning : In Q learning, state s is determined by the information an agent can get, and action a means the actual action it takes. On the other hand, GNP regards the state of the nodes after judging and processing as a state and a node transition as an action. The state s and action a of online learning of GNP are different from the ones determined by simple Q learning.

3) The reason for applying Q learning :

(1) Once GNP is booted up from the start node, the current node is transferred one after another without any terminal nodes. Therefore, the framework of reinforcement learning that uses the sum of the discounted future rewards are suitable for online learning of GNP.

(2) TD control needs only a maximum Q value in the next state, therefore, much memory is not needed and Q value is updated easily.

(3) Because GNP should search for an optimal solution independently of the policy (ε-greedy), off-policy is adopted.

4) Node transition and Learning : The following is the outline of online learning. In this paper, the transitions from each node to all the other nodes are possible, namely all N_{ij}=1 in this paper. The transition begins from the start node that can be selected from all the nodes arbitrarily and GNP executes the content of the start node, then GNP selects a transition having a maximum Q value in all transitions which connect the start node to the next one. However, by the probability of ε, it selects the random transitions. After a reward is given, Q value is updated according to the eq.(1). After that, the transitions continue following the network flow.

The concrete processes are described using Fig.4 that shows an example of node transitions. There are n nodes including judgement nodes and processing nodes, and each node has the number from 1 to n, respectively. At time t, it is supposed that the current node is node $i(1 \leq i \leq n)$. Because node i is the judgement node, it judges the current environment. In this example, the result of the judgement could be A, B or C and now B is supposed at time t. Then, the state s_t becomes S_i^B, and the transition is selected according to ε-greedy.

- case *1 : next node is a processing node*

If the transition which has $Q_{ij_1}^B$ is selected, the next node is processing node j_1. GNP processes what is assigned to it and gets the reward r_t. Then, the time is changed to $t+1$ and the state s_{t+1} becomes $s_{j_1}^A$ decisively because the processing nodes have no judgement unlike judgement nodes. If $Q_{j_1k_1}^A$ is the maximum value among $Q_{j_11}^A, \ldots, Q_{j_1n}^A$, this value is used to update $Q_{ij_1}^B$. GNP selects the transition to node k_1 which has the maximum Q value as an action a_{t+1} by the probability

of 1-ε, or selects the random transition by the probability of ε. In this case, Q value is updated accordingding to the eq.(2).

$$Q^B_{ij_1} = Q^B_{ij_1} + \alpha\Big[r_t + \gamma Q^A_{j_1k_1} - Q^B_{ij_1}\Big] \tag{2}$$

- case *2 : next node is a judgement node*

If the transition which has $Q^B_{ij_2}$ is selected, the next node is judgement node j_2 and GNP judges what is assigned to it. However, because the actual action is not done at node j_2, the reward r_t is not given. The time is changed to t+1 and the state s_{t+1} could be $S^A_{j_2}$, $S^B_{j_2}$ or $S^C_{j_2}$ according to the judgement at node j_2. Because it is supposed that the result of the judgement is A in this example, s_{t+1} becomes $s^A_{j_2}$. If $Q^A_{j_2k_2}$ is the maximum value among $Q^A_{j_21}, \ldots, Q^A_{j_2n}$, it is used for updating $Q^B_{ij_2}$. GNP selects the transition to node k_2 which has the maximum Q value as an action a_{t+1} by the probability of 1-ε, or selects the random transition by the probability of ε. The update of Q value is implemented according to eq.(3).

$$Q^B_{ij_2} = Q^B_{ij_2} + \alpha\Big[Q^A_{j_2k_2} - Q^B_{ij_2}\Big] \tag{3}$$

When updating Q values of the transitions to the judgement nodes, the discount rate γ is fixed to 1, that is, the discount of the rewards is not done. Online learning of GNP aims to maximize the sum of discounted rewards when actual actions are implemented.

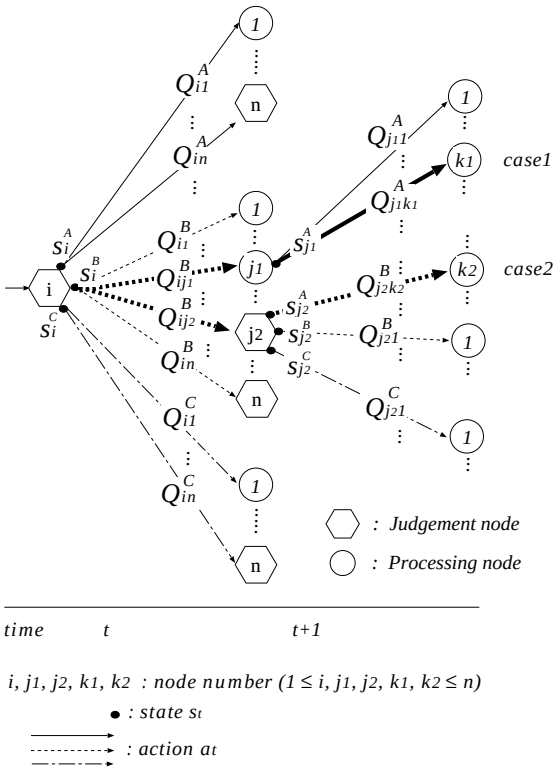

Fig.4 example of node transition

III. SIMULATIONS

In this section, GNP is used as a player of "Prisoner's dilemma game". The aim of this simulation is to confirm the effectiveness of the proposed online learning of GNP.

A. Prisoner's dilemma game

- story : The police don't have enough evidence to indict two suspects for complicity in a crime. The police approach each suspect separately. "If either of you remains silent, the sentence for your crime becomes a two-year prison in spite of the insufficient evidence. If you confess and your pal remains silent, you are released and the crime of your pal becomes a five-year prison. But, either of you confess, your sentence becomes a four-year prison."

TABLE. I
Relation between Sentence and Confession/Silence

Suspect1	Suspect2	Sentence for suspect1
Confession	Silence	released
Silence	Silence	2-year prison
Confession	Confession	4-year prison
Silence	Confession	5-year prison

From TABLE. I, Profits can be 0, -2, -4 and -5, respectively. In order to make them be positive numbers, 5 is added to each profit, then the profit matrix shown in Fig.5 is obtained. Silence is called "Cooperate(C)" because it shortens its prison term. Confession is called "Defect(D)" because it is done in order for only one suspect to be released. If the profits described by the symbols R, S, T and P in the frame follow inequality (4), the dilemma arises. This matrix is the famous one in many research on Prisoner's dilemma game. Therefore, the results in this paper are calculated using Fig.5. In this simulation, players repeat selecting Cooperation or Defect.

$$\begin{aligned} T > R > P > S \\ 2R > S + T \end{aligned} \tag{4}$$

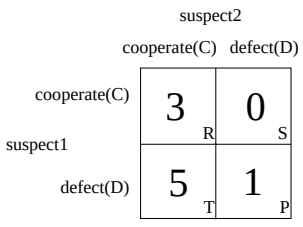

Fig.5 profit matrix (profit for suspect1)

If the players have no information about their opponents and they select the action only once, Defect is to be taken. In either case where the opponent takes Cooperation or Defect, Defect gets a higher profit than Cooperation. However, as the competition is repeated, and the characteristic

of the opponent comes out, GNP can develop its strategy considering the strategies of the opponent and itself of the past.

B. GNP for Prisoner's dilemma game

The player of Prisoner's dilemma game is regarded as an agent using GNP. The nodes of GNP are

- Self-judgement node (judge the action taken by itself)
- Opponent-judgement node (judge the action taken by an opponent)
- Cooperation processing node (C) (take Cooperation)
- Defect processing node (D) (take Defect)

Fig.6 shows an example of GNP structure for Prisoner's dilemma game. In this simulation, although each node is connected to all the other nodes, only a few connections are drawn in order to see them easily in Fig.6. The thick arrows show an example of node transitions.

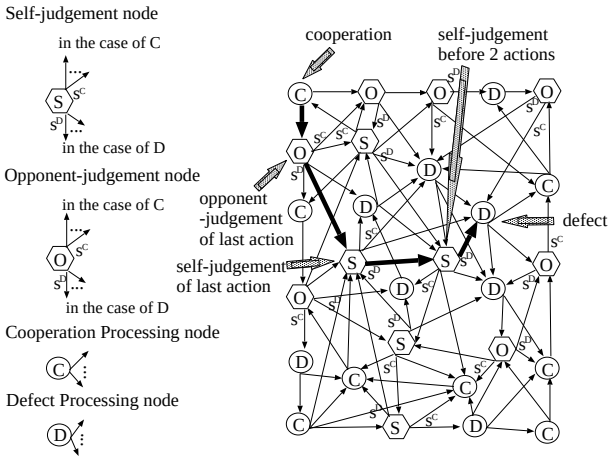

Fig.6 example of GNP structure

Each transition from judgement node has two Q values, Q^C and Q^D. Q^C is used when the past action of itself or the opponent is "cooperation" (judgement result C and the state becomes s^C). Q^D is used when the past action of them is "defect" (judgement result D and the state becomes s^D). On the other hand, each transition from the processing node has one Q value, which differs from a judgement node. After processing, if the next node is a self-judgement node, GNP judges the last action of itself, if an opponent-judgement node, it judges the one of the opponent. If the same kind of judgement node is executed again, GNP carries out the judgement of the action taken two steps before. In case the same kind of judgement nodes are continued to be selected, GNP judges previous action one after another. After processing, if the next node is a processing one, an agent competes using the corresponding processing against the opponent.

C. Simulation results

In this simulation, firstly GNP competes against Tit for Tat and Pavlov strategy that are the fixed strategies because they are predefined in advance and not changed. The purpose of these simulations is to show that GNP can learn the characteristics of the opponents and can change its strategies to get high scores. Next, two GNPs compete against each other. This situation can be regarded as a game in a dynamic environment because GNPs can change their strategies each other. Therefore, GNPs should adapt to the change of the strategies of the opponents.

GNP competes under the following conditions.

the number of nodes(N)	:	20
		Processing nodes
		: 5 per each Processing
		Judgement nodes
		: 5 per each judgement
discount rate(γ)	:	0.99
step size parameter(α)	:	0.5
ε	:	0.05
reward(r)	:	obtained by Fig.5
Q values	:	zero in the initial state

The competition is carried out for the predifined iterations after Q values are initialized (all Q values are zero at first). This process is called a trial.

1) Competition between GNP and Tit for Tat :

Tit for Tat :

- take Cooperation at the first iteration
- take the last opponent action from the second iteration

This strategy never loses a game by a wide margin and gets almost the same average scores as the opponent. Therefore, Tit for Tat is an admitted strong strategy. Even though the other strategies lose their scores against the opponent, Tit for Tat ends its competition with the same average as an opponent. Therefore, Tit for Tat becomes the strongest strategy as a whole. In this simulation, the competition between GNP and Tit for Tat is done. The competition is carried out for 20000 iterations and this trial repeated 100 times. In each trial, the moving averages over 10 iterations are calculated. Fig.7 shows the averages of these moving averages over 100 trials. Because Tit for Tat never loses by a wide margin, the average scores of GNP and Tit for Tat are almost the same and the lines of them are overlapped. As the competition proceeds, GNP gradually gets the high scores. If GNP takes Defect, Tit for Tat takes it in the next step and the score is diminished.

If GNP continues to take cooperation, Tit for Tat also do it, therefore, we can infer that GNP learned the cooperative

0-7803-7282-4/02/$10.00 ©2002 IEEE

strategy in order to get high scores. In the simulation, the more use of Cooperation node than Defect node is confirmed.

2) Competition between GNP and Pavlov strategy : TABLE. II shows Pavlov strategy. Pavlov strategy decides the next action according to the combination of the last actions of both strategies.

TABLE. II

Pavlov strategy

Pavlov	Opponent	Next action of Pavlov
Cooperation	Cooperation	Cooperation
Cooperation	Defect	Defect
Defect	Cooperation	Defect
Defect	Defect	Cooperation

Fig.8 shows the result which is calculated by the same procedure as simulation 1. GNP could win by a wide margin. GNP realized that it could defeat Pavlov strategy by taking Defect strategy. If both GNP and Pavlov strategy take Defect, Pavlov strategy takes Cooperation in the next step. Therefore, GNP learnd that it should take Defect next in order to get 5 point.

3) Competition between GNPs : Two GNPs which have the same parameters compete with each other in this simulation. Fig.9 shows a typical trial in order to study the progress of scores in detail. The average scores mean the moving averages over 10 iterations. The characteristic of the result is as follows. while GNP1 increases its scores, GNP2 decreases them and while GNP1 decreases its scores, GNP2 increases them. If GNP1 can win for some time by taking Defect more than Cooperation, GNP2 intends to take Defect. Then, the scores of GNP1 decrease. Therefore, GNP1 intends to change its strategy from Defect to Cooperation in order to avoid continuing to take Defect each other. However, GNP2 becomes to win thanks to the strategy change of GNP1. GNP1 intends to take Defect in turn and thus GNP2 intends to take Cooperation in order to avoid taking Defect each other. This process is repeated. From the viewpoint of online learning, this result is natural.

IV. CONCLUSIONS

In this paper, in order to adapt to dynamic environments quickly, online learning of GNP is proposed and applied to players for Prisoner's dilemma game and confirmed its learning ability. GNP can make its node transition rules according to Q values learned by Q learning. As the result of the simulations, GNP increased its score when competing against fixed strategy (Tit for tat and Pavlov strategy). In the competition between GNPs where strategies are dynamic, both GNPs can keep up with the changes of their strategies.

Henceforth, we will integrate online learning and offline learning so that the performance will be improved much more and GNP can model the learning mechanisms of organisms more realistically.

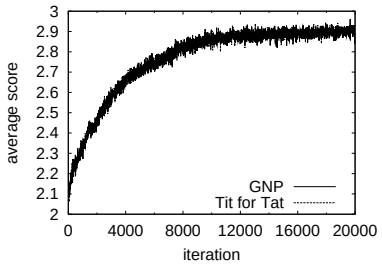

Fig.7 Competition between GNP and Tit for Tat

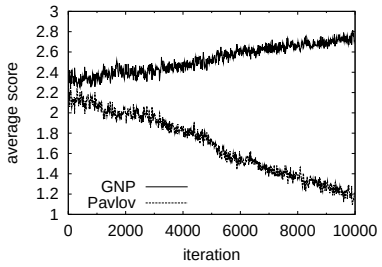

Fig.8 Competition between GNP and Pavlov strategy

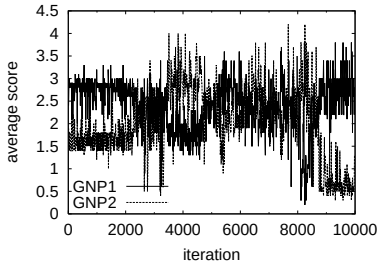

Fig.9 Competition between GNPs

References

[1] Holland, J. H., "Adaptation in Natural and Artificial Systems," University of Michigan Press, 1975, MIT Press, 1992

[2] Koza, John R., "Genetic Programming, on the programming of computers by means of natural selection," Cambridge, Mass., MIT Press, 1994

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

[4] K.Hirasawa, M.Okubo, H.Katagiri, J.Hu and J.Murata, "Comparison between Genetic Network Programming (GNP) and Genetic Programming (GP)," in Proc. of Congress on Evolutionary Computation, pp. 1276-1282, 2001

[5] Richard S. Sutton and Andrew G. Barto, "Reinforcement Learning - An Introduction," MIT Press Cambridge, Massachusetts, London, England, 1998

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

[7] L.J.Fogel, A.J.Owens, and M.J.Walsh, "Artificial Intelligence through Simulated Evolution," Wiley, 1966

0-7803-7282-4/02/$10.00 ©2002 IEEE