ICROS-SICE International Joint Conference 2009
August 18-21, 2009, Fukuoka International Congress Center, Japan

# Genetic Network Programming with General Individual Reconstruction

Fengming Ye, Shingo Mabu, Lutao Wang, Kotaro Hirasawa
Graduate School of Information, Production and Systems, Waseda University
yefengming@fuji.waseda.jp, mabu@aoni.waseda.jp, wanglutao@fuji.waseda.jp, hirasawa@waseda.jp

*Abstract*— **Genetic Network Programming (GNP) which has been developed for dealing with problems in dynamic environments is a newly proposed evolutionary approach with the data structure of directed graphs. GNP has been used in many different areas such as data mining, extracting trading rules of stock markets, elevator supervised control systems, etc and has obtained some outstanding results. Focusing on GNP's distinguishing expression ability of the graph structure, this paper proposes a method named Genetic Network Programming with General Individual Reconstruction (GNP with GIR) which reconstructs the gene of randomly selected individuals and then undergoes the special genetic operations by using the transition information of better individuals. The unique individual reconstruction and genetic operations make individuals not only learn the experiences of better individuals but also strengthen exploration and exploration ability. GNP with GIR will be applied to the tile-world which is an excellent benchmark for evaluating the proposed architecture. The performances of GNP with GIR will be compared with conventional GNP demonstrating its superiority.**

## I. Introduction

A lot of research achievements in evolutionary computation have been obtained, such as Genetic Algorithm (GA)[1][2], Genetic Programming (GP)[3][4], Evolutionary Programming (EP)[5][6] and Evolution Strategies (ES)[7]. The essential concept of all of the above approaches is originated from Darwin's Evolution Theory. However the difference between them is about their distinct methodologies. The gene of GA is represented as a string structure. Traditionally, solutions are represented in binary as strings of 0s and 1s. But, for different problems, other encodings such like real number encoding and integer encoding are also available. GP extends GA's expression ability by its tree structure which can be easily evaluated in a recursive way. Each inner node of the tree has an operator function and every terminal node has an operand, as a result, for example, mathematical expressions are very easy to represnt. Fogel used finite state machines for EP as predictors and evolved them. The finite state machine is a model of behavior composed of a finite number of states, transitions between those states, and actions.

Recently, a novel method named Genetic Network Programming (GNP) [8][9][10] has been proposed. GNP is devised to deal with problems in dynamic environments effectively and efficiently [11][12]. In GNP, the structure of directed graph is used to represent the gene. The node transition of the directed graph begins from a start node and transfers based on the judgments on the nodes and node connections, and the agent takes concrete actions according to the functions on the nodes. Compared with the classical evolutionary approaches, GNP has some unique characteristics due to its distinguished directed graph structure: 1) The gene structures of GNP individuals are composed of a number of nodes which execute simple judgment/processing, and these nodes are connected by directed links to each other. 2) The graph structure enables GNP to re-use nodes, thus the structure can be very compact. GP has a tree structure which brings a problem that the size of the tree is uncontrollable, if the problem complexity is unexpectedly high. However, GNP's directed graph structure can carry out some repetitive processes. That is to say, the reusability of nodes makes GNP's structure more compact than that of GP. 3) The node transition of GNP is executed according to its node connections without any terminal nodes. Concretely speaking, the node transition of GNP begins from a start node and transfers based on the judgments on the nodes and node connections, thus it can be said that, for example, agent's actions in the past are implicitly memorized in the network flow of the graph. GNP has been used in many areas such as data mining, forecasting stock markets, elevator supervised control systems, etc. and has obtained outstanding performances in these areas.

However, it is generally found that not all of the GNP nodes and connections but only a part of them is used by agents. So the route of GNP which consists of the used successive GNP transitions from node to node is just the behavior regulation to guide the agent's action upon the environment. Thus, our goal is to extract and accumulate the information of GNP route that agents took and make general use of them to guide the evolutionary process. In the proposed method, the genes of a part of individuals are reconstructed by using route information and also the route information will guide the process of crossover and mutation. In order to verify the effectiveness of the proposed method we report the experimental results using the tile-world.

This paper is organized as follows: Section 2 introduces the basic concept of GNP. Section 3 explains the details of the proposed method. Section 4 describes the experimental environments and reports the simulation results. Section 5 is the conclusion part.

## II. Genetic Network Programming

In this section, the basic concept of GNP including the gene structure and genetic operators is introduced. Generally



PR0002/09/0000-3474 ¥400 

speaking, GNP having a directed graph structure is an extension of GA and GP. The aim of developing GNP is to deal with dynamic environments efficiently by using directed graph structures which have more general representation ability than that of strings in GA and trees in GP. The GNP genetic operators reproduce offspring by exchanging and randomly modifying the gene information of parents and this kind of concept is similar to GA and GP.

### A. Directed Graph Structure of GNP

An individual of GNP contains a fixed number of nodes which are classified into 2 categories: Judgment Node and Processing Node. Judgment Node judges the current state on the environments, and according to the judgement result, the agent selects the following node. In concrete, Judgment Node has multiple branches connecting to different nodes, and after the judgement, a decision should be made to select one branch and move to the next node. Processing Node takes some actions and changes the current state according to some regulations. Different processing nodes take different actions. Therefore, Processing Node has only one branch connecting to the following node. Fig. 1 shows an example of a GNP individual. We can see that in the example, the GNP individual has totally 6 nodes including 1 start node, 3 judgement nodes and 2 processing nodes. And each judgement node has 2 branches connecting to 2 nodes, respectively.

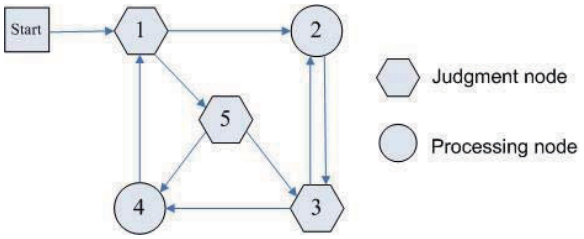


Fig. 1. The directed graph structure of GNP

### B. Genetic Operators of GNP

GNP also has its own genetic operators which are similar to that of GA: mutation and crossover. Mutation is to change the gene of the selected individual. And crossover is to exchange the corresponding part of 2 selected parent individuals and obtain 2 offspring having new genes.

*1) Mutation:* Mutation operator affects only one individual. All the gene information of each node are changed randomly by mutation rate of $P_m$, and one offspring is generated. The mutation refers to the change of genetic information on connections $C_{ij}$ of each node in GNP by the predefined probability $P_{mc}$, and decides the connection. Therefore, the genetic information $C_{ij}$ is modified in the range of the node number, randomly. The example of the mutation is shown in Fig. 2. Before carrying out the mutation, the first branch of node 2 and the second branch of node 7 are connected to node 3 and node 4, respectively. After carrying out the mutation, the first branch of node 2 and the second branch of node 7 are connected to node 5 and node 8, respectively.

*2) Crossover:* Crossover undergoes between two parents and produces two offspring. The connections of the uniformly selected corresponding nodes in two parents are swapped with each other by crossover rate of $P_c$, and two offspring are generated. Fig. 3 shows an example of one point crossover in GNP, where node 4 is selected as a crossover point randomly and the whole genetic information separated by its node is exchanged. So, the node types, node functions and node connections from node 4 to node 9 of the offspring are different from their parents.

### C. The Algorithm of GNP

GNP follows the evolutionary algorithm which is similar to that of GA. First, we initialize the population whose individuals have randomly generated node connections. Then, the evolution process is done iteratively generation by generation until the optimal solution is obtained. Fig.4 shows the flow of GNP.

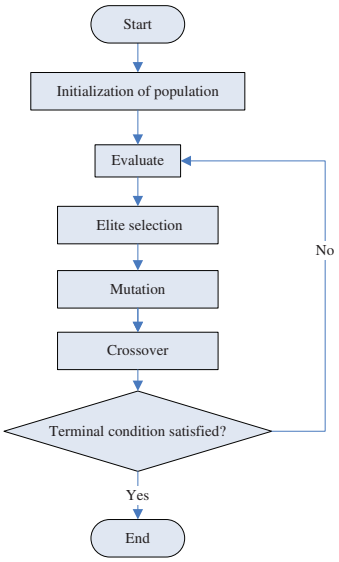


Fig. 4. Flow chart of GNP

## III. GNP with General Individual Reconstruction

Conventional GNP has a kind of feature that some of GNP nodes and connections may not be used by agents during its transition. For example, Fig. 5 shows such a case that the agent follows a transition 1-2-3-5-4. The connections from node 2 to node 6, from node 3 to node 4, from node 5 to node 6 and from node 6 to node 1 are not used.

For further explanation, the definition of GNP route is given as follows.

**Definition 1 (GNP route).**

GNP route is a path on which the agent travels in a GNP individual. It consists of all the nodes and connections that the agent passes by. As shown in Fig. 5, in this example, the

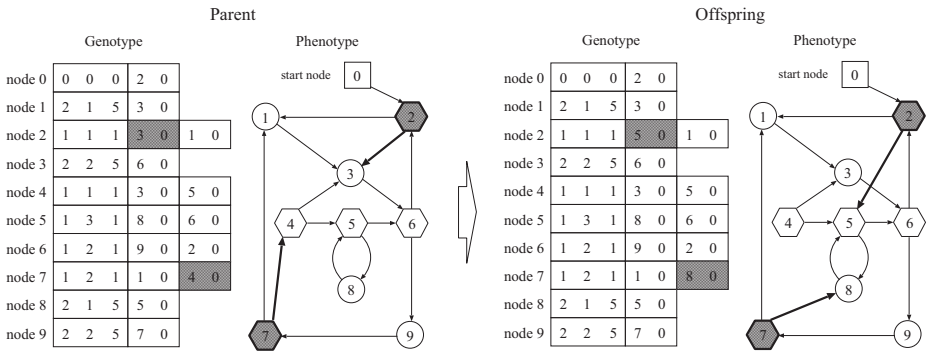


Fig. 2. Mutation of GNP

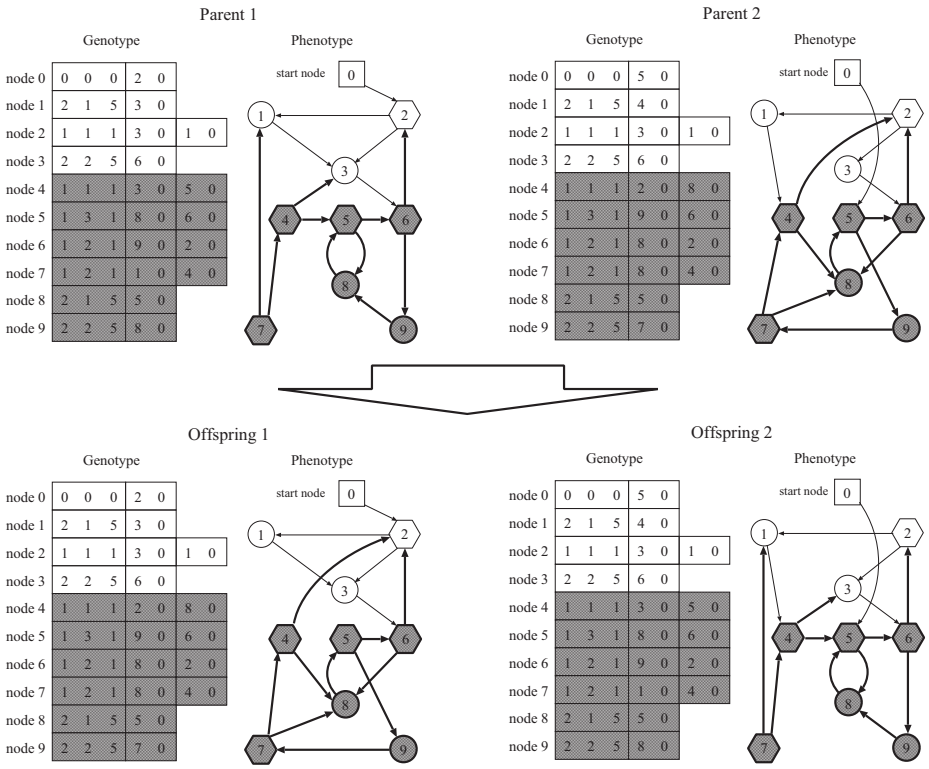


Fig. 3. Crossover of GNP

GNP route is $1-2-3-5-4$ consisting of all the used nodes and connections. In concrete, the GNP route is coded in a string structure, where the node number and the connection index of nodes are coded bit by bit.

Because an individual's fitness value is calculated only by the GNP route. As a result, GNP route could be considered essential to obtain good fitness values. The aim of the proposed method, GNP with GIR, is to collect the information on the routes of GNP individuals and make use of them to guide the evolution process and finally to get better individuals. Compared with conventional GNP, GNP with GIR has a new genetic operator named reconstruction which modifies the gene structures of individuals by using GNP route information. In addition to the reconstruction, the mutation and crossover of GNP with GIR reproduce offspring by using route information, too. In each generation, the elites are directly reserved to the next generation, and then the genetic operators: reconstruction, mutation and crossover undergo the population. Fig.6 shows the flow of GNP with GIR.

### *A. Reconstruction*

In the individual reconstruction, 2 individuals, for example, $Ind_i$ and $Ind_j$ are randomly selected. If

$$fitness_i / fitness_j \geq \alpha, \alpha > 1 \tag{1}$$

where $fitness_i$ and $fitness_j$ are the fitness values of $Ind_i$ and $Ind_j$, respectively and $\alpha$ is a threshold, we use the route

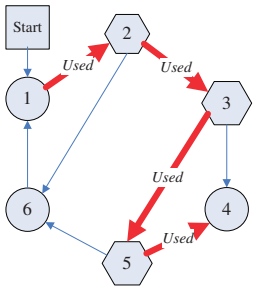


Fig. 5. An example of GNP route

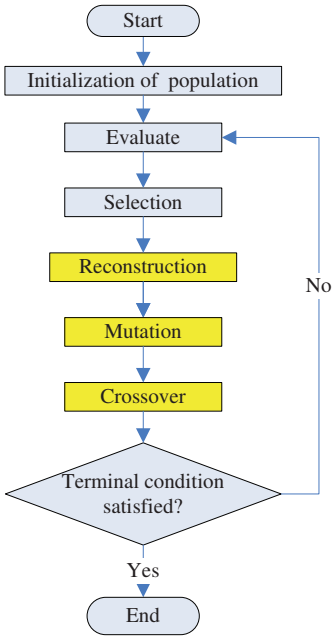


Fig. 6. Flow chart of GNP with GIR

information of $Ind_i$ to reconstruct $Ind_j$. For example, if the GNP route of $Ind_i$ is shown in Fig.7 which indicates a route from Node $I$ to Node $J$ via the $b_i^{th}$ branch, from Node $J$ to Node $K$ via the $b_j^{th}$ branch, etc., the gene of $Ind_j$ will be modified by this route. (i.e: in $Ind_j$, Node $I$ will connect to Node $J$ via the $b_i^{th}$ branch, Node $J$ will connect to Node $K$ via the $b_j^{th}$ branch, etc.)

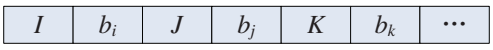


Fig. 7. GNP Route Coding

## B. Mutation

The mutation in GNP with GIR randomly changes the route part of the parent individual. It means the change only occurs on the route of the parent individual. Fig. 8 shows an example of how mutation in GNP with GIR works. Before mutation, the first branch of node 1 connects to node 3 and after mutation, connects to node 4.

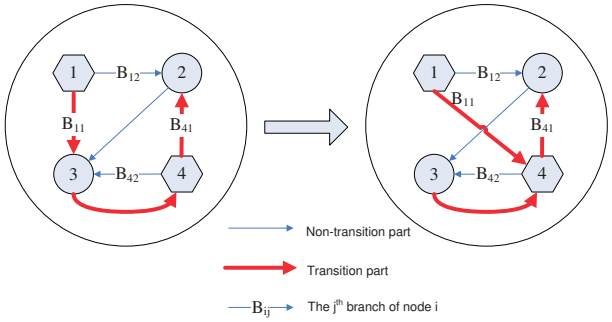


Fig. 8. Mutation in GNP with GIR

## C. Crossover

In crossover, 2 parents $P_i$ and $P_j$ produce 2 offspring $O_i$ and $O_j$. $O_j$ will be reconstructed by the route information of the parents, but $O_j$ will be reconstructed by the non-route information of the parents as much as possible. It means to enhance both the exploitation and exploration abilities. Fig. 9 shows an example of the crossover between $P_i$ and $P_j$. Because $fitness_i > fitness_j$, GNP route and non-route part of $P_i$ has the higher priority to reconstruct $O_i$ and $O_j$, respectively.

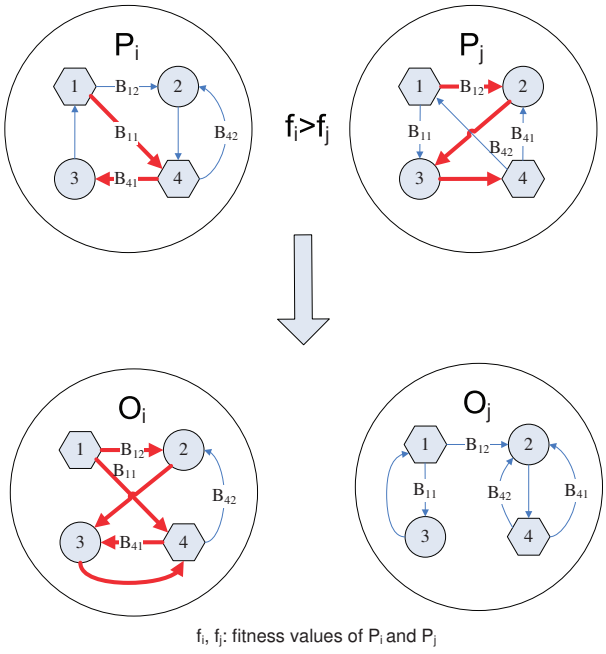


Fig. 9. Crossover in GNP with GIR

# IV. Simulations

## A. Experimental Environments

We use the tile-world, a good architecture to evaluate agent oriented systems, as an experimental environment [13]. A

tile-world contains agents, floors and walls (obstacles). On the floors there are a number of tiles and holes dispersing on different positions, respectively. The agents try to find the positions of tiles and holes. They should pick up tiles and carry them until they reach the holes. Then, they cover the holes with tiles. When agents are moving, they have to avoid the obstacles and uncovered holes. Since each agent can carry only one tile at a time, the agents will continue to move until all the holes are covered by tiles or they use up all allowable steps. So the objective of agents is to cover all the holes with tiles using as few steps as possible.

In our simulations, we trained GNP for the agents in 10 different worlds. Each world has 3 agents, 3 tiles and 3 holes. The positions of holes, obstacles and agents are the same in the 10 worlds. However, the positions of tiles are different from each other. Fig. 10 shows the environments for training.

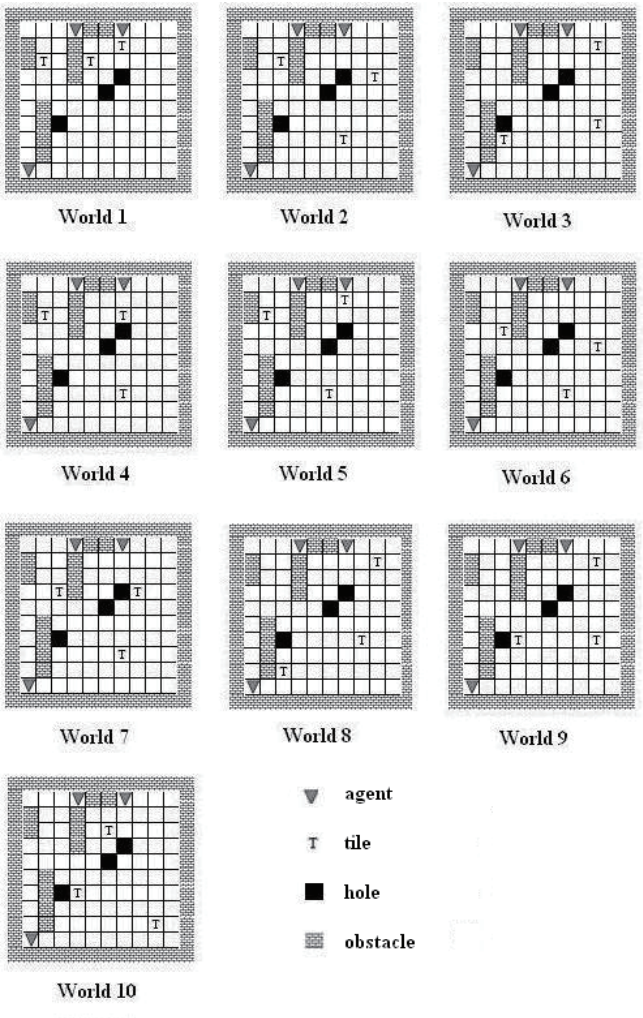


Fig. 10. Training environments

After training, we tested the trained GNP in 8 new different environments, where the positions of tiles, holes and obstacles are totally different. Fig. 11 shows the environments for testing.

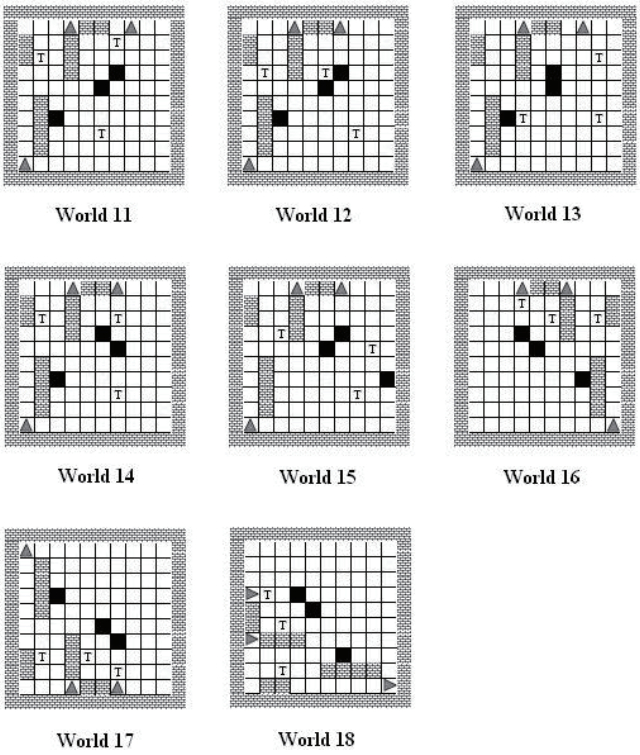


Fig. 11. Testing environments

### *B. Programming Configuration*

In our program, there are 8 kinds of Judgment Nodes. Agents can find out what exists in front of each agent, in the same way, right, left and back of each agent. Agents can also find out the rough direction from the agents to the place where the nearest tile is, where the second nearest tile is and where the nearest hole is. Furthermore, they can find out the rough direction from the nearest tile of the agent to the nearest hole. These different judgements help agents to make a decision in the following step.

And there are 4 kinds of Processing Nodes: to go forward, to turn left, to turn right and to stay. Once the agent takes an action, it consumes one step. In our program, totally, there are 60 allowable steps.

Each individual contains 60 nodes including 40 Judgement Nodes (5 for each kind of Judgement Nodes) and 20 Processing Nodes (5 for each kind of Processing Nodes). Each Judgement Node has 5 branches and each Processing Node has only one branch.

We used population of 201 individuals and tournament selection in the experiments, with *crossover rate* = 0.1, *mutation rate* = 0.01 and the threshold $\alpha = 1.4$ in equation 1. And all cases are carried out for 50 random rounds. Table 1 shows other parameter configurations.

TABLE I

PARAMETER CONFIGURATION

| Reconstruction size | Mutation size | Crossover size | Elite size |
|---|---|---|---|
| 20 | 110 | 70 | 1 |
| Threshold $\alpha$ | 1.4 | Number of generations | 500 |

Fitness function is calculated as follows:

$$\begin{aligned} Fitness &= C_{tile} \times DroppedTile \\ &+ C_{dist} \times \sum_{t \in T} (InitialDist_t - FinalDist_t) \\ &+ (TotalStep - UsedStep), \end{aligned} \tag{2}$$

where, $DroppedTile$ is the number of tiles that the agents have dropped in holes, $InitialDist_t$ is the initial distance of the $t^{th}$ tile from the nearest hole, $FinalDist_t$ is the final distance of the $t^{th}$ tile from the nearest hole, $UsedStep$ is the number of used steps and $T$ is the set of suffixes of tiles. $C_{tile}$, $C_{dist}$ and $TotalStep$ are parameters configured by the designer. In our simulation, $C_{tile}$=100, $C_{dist}$=20 and $TotalStep$ = 60.

### C. Simulation Results

Fig. 12 shows the average best fitness curve of training results of GNP and GNP with GIR over 50 random rounds. The average best performance of GNP and GNP with GIR is 3681 and 3992 after 500 generations, respectively.

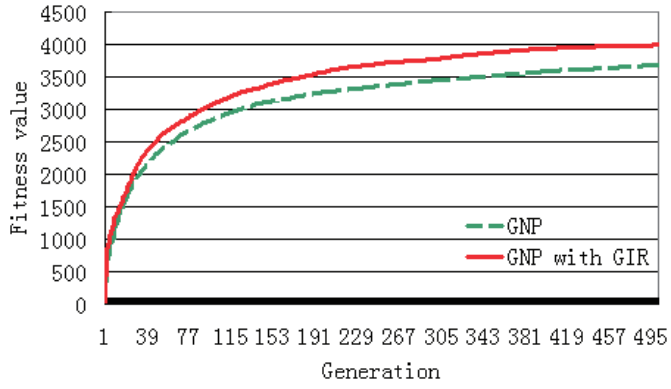


Fig. 12. Fitness curve of training results

Table II shows the average best testing fitness results of GNP and GNP with GIR per tile-world over 50 random rounds in the 8 new worlds, respectively. We can see that in new environments, where the agents have never been trained before, GNP with GIR can obtain better testing results than GNP. Only in the World 15, GNP with GIR performed more poorly than GNP and GNP with GIR performed better than the conventional GNP in all of the other cases. The testing results indicate that GNP with GIR not only can perform better than GNP in the trained environments, but also GNP with GIR has more generalization ability than GNP in the new environments.

TABLE II

TESTING RESULTS OF GNP AND GNP WITH GIR

| Individuals | GNP | GNP with GIR |
|---|---|---|
| World 11 | 153 | 176 |
| World 12 | 149 | 194 |
| World 13 | 215 | 268 |
| World 14 | 78 | 115 |
| World 15 | 134 | 106 |
| World 16 | -36 | 92 |
| World 17 | 198 | 234 |
| World 18 | 112 | 154 |
| Average | 141 | 214 |

## V. CONCLUSION

We proposed a method of GNP with General Individual Reconstruction (GNP with GIR) which shows an improvement of GNP. The genetic operators of the proposed method modify the gene structures of the individuals by using the information of GNP routes in order to enhance the conventional GNP. The simulation results in the tile-world shows the superiority of GNP with GIR over the conventional GNP both in training and testing phase.

## REFERENCES


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

[2] D. E. Goldberg, "Genetic Algorithm in Search Optimization and Machine Learning", Reading, MA: Addison-Wesley, 1989.

[3] J. R. Koza, "Genetic Programming, on the Programming of Computers by Means of Natural Selection", MIT Press, Cambridge, MA, 1992.

[4] J. R. Koza, "Genetic Programming II, Automatic Discovery of Reusable Programs", MIT Press, Cambridge, MA, 1994.

[5] D. B. Fogel, "An introduction to simulated evolutionary optimization", IEEE Transactions on Neural Networks, 5(1) pp. 3-14, 1994.

[6] L. J. Fogel, "Evolutionary Programming in Perspective: The Top-down View", in Computational Intelligence: Imitating Life, J. M. Zurada, R. J. Marks II, and C. Goldberg, Eds. Piscataway, NJ: IEEE Press, 1994.

[7] I. Rechenberg, "Evolution Strategy", in Computational Intelligence: Imitating Life, J. M. Zurada, R. J. Marks II, and C. Goldberg, Eds. Piscataway, NJ: IEEE Press, 1994.

[8] S. Mabu, K. Hirasawa and J. Hu, "A Graph-Based Evolutionary Algorithm: Genetic Network Programming (GNP) and Its extension Using Reinforcement Learning", Evolutionary Computation, Vol. 15, No. 3, pp. 369-398, 2007.

[9] T. Eguchi, K. Hirasawa, J. Hu and N. Ota, "A Study of Evolutionary Multiagent Models Based on Symbiosis", IEEE Trans. on Systems, Man and Cybernetics, Part B, Vol. 36, No. 1, pp. 179-193, 2006.

[10] K. Hirasawa, T. Eguchi, J. Zhou, L. Yu and S. Markon, "A Double-Deck Elevator Group Supervisory Control System Using Genetic Network Programming", IEEE Trans. on Systems, Man and Cybernetics, Part C, Vol. 38, No. 4, pp. 535-550, 2008.

[11] S. Mabu, K. Hirasawa, J. Hu and J. Murata, "Online learning of Genetic Network Programming", In Proc. of the Congress on Evolutionary Computation, pp. 321–326, 2002.

[12] S. Mabu, K. Hirasawa and J. Hu, "Genetic network programming with learning and evolution for adapting to dynamical environments", In Proc. of the Congress on Evolutionary Computation, pp. 69–76, 2003.

[13] M. E. Pollack and M. Ringuette. "Introducing the tile-world: Experimentally evaluating agent architectures", In T. Dietterich, and W. Swartout, editors, In Proceedings of the conference of the American Association for Artificial Intelligence, pp. 183-189, AAAI Press, 1990.