A Memory Scheme for Genetic Network Programming with Adaptive Mutation

[Extended Abstract]

Fengming Ye
Graduate School of Information, Production and Systems
Waseda University, Japan
yefengming@fuji.waseda.jp

Shingo Mabu
Graduate School of Information, Production and Systems
Waseda University, Japan
mabu@aoni.waseda.jp

Kotaro Hirasawa
Graduate School of Information, Production and Systems
Waseda University, Japan
hirasawa@waseda.jp

ABSTRACT

Recently, a new approach named Genetic Network Programming (GNP) has been proposed for especially solving complex problems in dynamic environments. In this paper, we propose a memory scheme for GNP to enhance the performance of GNP and use SARSA learning based adaptive mutation mechanism to guide the GNP evolution process.

Categories and Subject Descriptors

I.6.3 [**Computer Applications**]: Simulation and Modeling - Applications

General Terms

Algorithms

Keywords

genetic network programming, adaptive stock selection, stock markets, risk management, portfolio selection

1. INTRODUCTION

In this paper, we focus on a recently proposed approach named Genetic Network Programming (GNP)[1], a variation of GP, which adopts directed graphs. The effectiveness of GNP has been demonstrated by previous research on various complex applications. Considering the advantages of GA having the memory scheme [4, 2], we think GNP will also be enhanced by employing the memory scheme in dynamic problems. The memory scheme stores the best solutions of each generation in this paper and accumulates the Q value information measured by SARSA learning[5].

2. PROPOSED METHOD: GNP-RISLAM

The proposed approach is named Genetic Network Programming with reconstructed individuals[6] and SARSA learning-based adaptive mutation(GNP-RISLAM). In the reconstruction phase, worse individuals learn experiences from the elites and the learned Q values are used to measure the utilities of the branches. In GNP-RISLAM, after individual reconstruction, the genetic operators will be conducted and specifically, the traditional uniform mutation is replaced with an adaptive mutation.

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

2.1 Definitions

For further explanation, some definitions are given as follows.
Trial: A trial refers to the process for an agent to execute a task being supervised by GNP.
Route: A route refers to the sequence of nodes and branches occurring in a trial.
State: A state refers to a branch of a node.
Action: An action refers to a node.

2.2 SARSA Learning Model

The proposed learning approach mainly includes four steps, summarized as follows:
1. Establish a Q table that contains all the possible state-action pairs.
2. After each trial, obtain a route, a score and some instant rewards.
3. Use the score and rewards to update the Q value with the following update equation, following a backwards order.

$$Q(s,a) = Q(s,a) + \alpha \cdot (r + \gamma \cdot Q(s',a') - Q(s,a)), \quad (1)$$

where, $Q(s,a)$ is the Q value of the current state-action pair, $Q(s',a')$ is the Q value of the next state-action pair. r is the reward. α denotes the learning rate, while γ denotes the discount factor.
4. For different trials, repeat step 2 and step 3 to update the Q table iteratively until the end of the evolution.

2.3 Adaptive mutation

If the Q value of the branch-node pair passes a threshold T, we still adopt the predefined mutation rate to perform mutation, otherwise a monotonically decreasing function is utilized to calculate the mutation rate, to be specific, the proposed adaptive mutation.

3. EXPERIMENTAL REPORT

In this paper, the proposed method is evaluated in the tile-world problems[3]. We conducted 2 simulations. In simulation 1, we trained the agents in the experimental environments of 10 tile-worlds and compared the performances of GNP-RISLAM, GNP-RI, GNP-SLAM and GNP. In simulation 2, we trained the agents in other 6 tile-worlds which are much more complicated for the agents.

3.1 Programming Configuration

The fitness is calculated by accumulating the scores obtained from each tile-world. The score function is closely related to the objective of the tile-world problem, represented by

$$Score = 100 \cdot DT + 20 \cdot \sum_{p=1}^{P} d(p) + (M_t - U_t), \qquad (2)$$

where,DT is the number of tiles dropped into the holes, p is the ID of the relatively nearest tile-hole pair at every time step in the trials, P is the maximum number of the relatively nearest tile-hole pairs, $d(p)$ is the decrease of the distances between the tiles and holes in the pairs, M_t is the maximum time step, and U_t is the used time step.

Then, the fitness function is defined by

$$Fitness = \sum_{w=1}^{W} Score(w), \qquad (3)$$

where, w is the ID of the tile-world, W is the maximum number of training tile-worlds, and $Score(w)$ is the score obtained in the wth tile-world.

3.2 Simulation Results

Fig. 1 shows the average best fitness curves over 30 random rounds in the training of GNP-RISLAM, GNP-RI, GNP-SLAM and GNP, which shows that GNP-RISLAM obtained the best results among 4 methods in simulation 1. Moreover, GNP-RI and GNP-SLAM also perform better than standard GNP. The result suggests both GNP-RI and GNP-SLAM can enhance the architecture of GNP and the combination of these two approaches can make the performance of it even better.

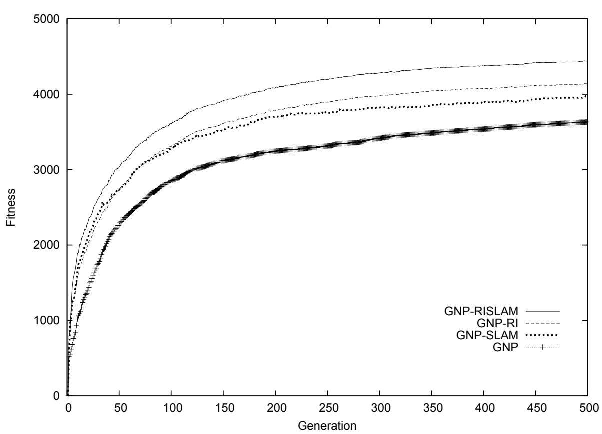

Figure 1: Average best fitness curves over 30 random rounds in simulation 1

Fig. 2 shows the average best fitness curves over 30 random rounds in the training of the four architectures in simulation 2. The experimental result also shows that GNP-RISLAM obtained the best results among 4 methods.

4. CONCLUSION

This paper introduces an approach employing a memory scheme in GNP to improve its performance. The proposed approach is named Genetic network programming with reconstructed individuals and SARSA learning based adaptive mutation (GNP-RISLAM). Based on the learned knowledge, it replaces the traditional uniform mutation with adaptive mutation. The experiments conducted on the tile-world problem reveal several advantages of GNP-RISLAM.

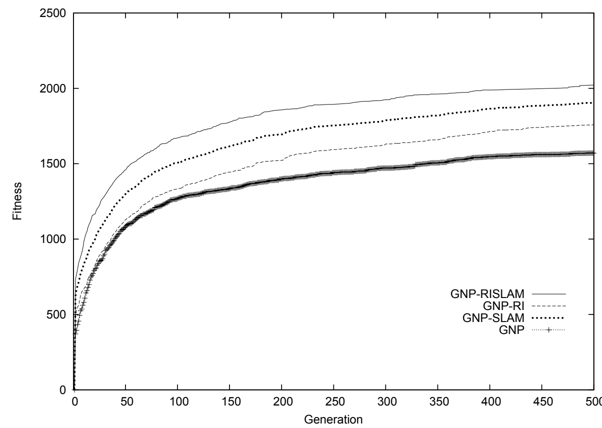

Figure 2: Average best fitness curves over 30 random rounds in simulation 2

5. REFERENCES

[1] H. Katagiri, K. Hirasawa, J. Hu, J. Murata, and M. Kosaka. Network structure oriented evolutionary model: Genetic network programming. *Trans. of the Society of Instrument and Control Engineers*, 38(5):485–494, 2002.

[2] S. J. Louis and Z. Xu. Genetic algorithms for open shop scheduling and re-scheduling. In *Proc. of the 11th ISCA International Conference on Computation and Their Application*, pages 99–102, 1996.

[3] M. E. Pollack and M. Ringuette. Introducing the tile-world: experimentally evaluating agent architectures. In *Proc. of the conference of the American Association for Artificial Intelligence*, pages 183–189, 1990.

[4] C. L. Ramsey and J. J. Grefenstette. Case-based initialization of genetic algorithms. In *Proc. of the 5th International Conference on Genetic Algorithms*, pages 84–91, 1993.

[5] G. A. Rummery and M. Niranjan. *On-Line Q-Learning Using Connectionist Systems*. Technical Note, 1994.

[6] F. Ye, S. Mabu, L. Wang, S. Eto, and K. Hirasawa. Genetic network programming with reconstructed individuals. *SICE Journal of Control, Measurement, and System Integration*, 2(2):121–129, 2010.