Genetic Network Programming with Reinforcement Learning and Its Performance Evaluation

Shingo Mabu, Kotaro Hirasawa, and Jinglu Hu

Graduate School of Information, Production and Systems, Waseda University, Hibikino 2-7, Wakamatsu-ku, kitakyushu, Fukuoka, Japan,
mabu@asagi.waseda.jp, {hirasawa, jinglu}@waseda.jp

Abstract. A new graph-based evolutionary algorithm named "Genetic Network Programming, GNP" has been proposed. GNP represents its solutions as directed graph structures, which can improve the expression ability and performance. Since GA, GP and GNP already proposed are based on evolution and they cannot change their solutions until one generation ends, we propose GNP with Reinforcement Learning (GNP with RL) in this paper in order to search solutions quickly. Evolutionary algorithm of GNP makes very compact directed graph structure which contributes to reducing the size of the Q-table and saving memory. Reinforcement Learning of GNP improves search speed for solutions because it can use the information obtained during tasks.

Recently, a new directed graph-based evolutionary algorithm named "Genetic Network Programming (GNP)[1]" has been proposed. Basically, GA and GP represent solutions as string and tree structures, respectively, but GNP represents its solutions as graph structures composed of a number of judgement nodes and processing nodes [Fig. 1]. Judgement nodes are *if-then* type branch decision functions, and Processing node determines an action/processing an agent should do. The GNP we used never causes bloat because of the predefined number of nodes although GNP can evolve the programs having variable sizes. The graph structure of GNP has some distinguished abilities inherently. 1) Since the graph structure of GNP inherently have the ability to re-use nodes unlike GA and tree GP, GNP can use certain Judgement/Processing nodes repeatedly to achieve tasks. Therefore, even if the number of nodes is predefined and smaller than GP programs, GNP can perform well by making effective programs based on re-using nodes. So we do not have to prepare the excessive number of nodes, as a result, the structures of GNP become very compact, which contributes toward saving the calculation time and physical memory. 2) GNP starts its node transition from a start node and continues its node transition according to the node connections without any terminal. So a current node is selected influenced by the past node transition. Therefore, GNP can have the implicit memory function which memorizes the past action sequences of agents in the network flow. The node transition ends when the end condition is satisfied, for example, the time step reaches the maximum one or the GNP program completes the given tasks.

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 710–711, 2004.

However, conventional GNP are based on evolution, i.e., after the programs are carried out to some extent, they are evaluated and evolved based on their fitness values, so many trials must be done again and again. To overcome this problem and search solutions quickly, we have proposed a new algorithm of GNP[2] which combines evolution and reinforcement learning (RL). Since RL is done when agents carry out their task, GNP can search for better solutions every judgement/processing besides the evolution executed every generation. The aim of the combination of RL and evolution is to take advantage of the sophisticated search abilities of evolution and online learning of RL.

In this paper, a new algorithm of GNP with Reinforcement Learning is proposed. This method is a extension of the previous GNP, so it becomes more general framework of GNP with RL in terms that 1) we defined the new state and action pairs used by RL which are different from the previous method, 2) the proposed method can change node functions in addition to node connections and 3) save the number of parameters used by RL, so the size of the Q-table becomes small.

In order to confirm the ability of the proposed method, the simulations using tileworld and maze problem which are the typical benchmark problems of agents are carried out, and the results of the proposed method are compared with those of standard GNP (GNP using evolution only) and standard tree GP. From the simulation results, it is clarified that the proposed method can obtain the best results and learn faster than the previous algorithm [Fig. 2, 3].

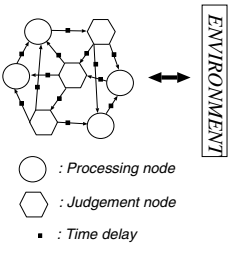

Fig. 1. Basic structure of GNP

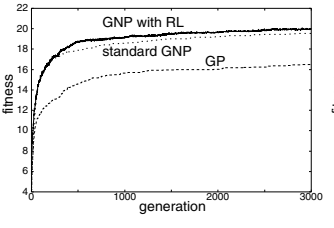

Fig. 2. Tileworld problem

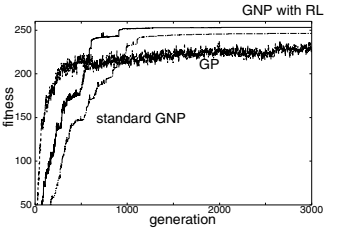

Fig. 3. Maze problem

References

1. H.Katagiri, K.Hirasawa, J.Hu and J.Murata, "Network structure Oriented Evolutionary Model - Genetic Network Programming - and Its comparison with Genetic Programming", *in 2001 Genetic and Evolutionary Computation Conference, Late Breaking Papers*, pp. 219-226, (2001).
2. S. Mabu, K. Hirasawa, and J. Hu, "Genetic Network Programming with Learning and Evolution for Adapting to Dynamical Environments", *in Proc. of 2003 Congress on Evolutionary Computation*, pp. 69-76, (2003).