早稲田大学大学院情報生産システム研究科

博 士 論 文 概 要

論 文 題 目

Genetic Network Programming Based Rule Accumulation for Agent Control

申 請 者

Lutao WANG

情報生産システム工学専攻
ニューロコンピューティング研究

2012 年 11 月

Agent control is a hot but challenging research topic which covers many research fields, such as evolutionary computation, machine learning, neural networks, etc.. Various approaches have been proposed to guide agents' actions in different environments. Evolutionary Computation (EC) is often chosen to control agents' behaviors since it can generate the best control policy through evolution.

Genetic Network Programming is a newly developed EC method which chooses the directed graph structure as its chromosome. High expression ability of the graph structure, reusability of nodes and realization of partially observable processes enable GNP to deal with many complex problems in dynamic environments. One of the disadvantages of GNP is that its gene structure may become too complicated after generations of the training. In the testing phase, it might not be able to adapt to the new environment easily and its generalization ability is not good. This is because the implicit memory function of GNP is not enough when dealing with dynamic agent control problems. Therefore, explicit memory should be associated with GNP in order to explore its full potential.

Various research has revealed that memory schemes could enhance EC in dynamic environments. This is because storing the useful historical information into the memory could improve the search ability of EC. Inspired by this idea, a GNP-based memory scheme named "Genetic Network Programming with Rule Accumulation" (GNP-RA) is proposed in this thesis. Focusing on this issue, it is studied in the following chapters of this thesis how to create action rules and use them for agent control, how to improve the performance for agent control in Non-Markov environments, how to prune the bad rules to improve the quality of the rule pool, how to build a rule pool adapting to the environment changes and how to improve the robustness of the rule pool by creating more general rules. The organization of this thesis is as follows.

Chapter 1 describes the research background, problems to be solved and outline of the thesis. Some classical methods in the domain of evolutionary computation and reinforcement learning are reviewed.

Chapter 2 designs the general framework of GNP-RA, which contains two stages. In the training stage, the node transitions of GNP are recorded as rules and stored into the rule pool generation by generation. In the testing stage, all the rules in the rule pool are used to determine agents' actions through a unique matching degree calculation. The very different point of GNP-RA from the basic GNP is that GNP-RA uses a great number of rules to determine agents’ actions. However, GNP could use only one rule corresponding to its node transition. Therefore, the generalization ability of GNP-RA is better than that of GNP. Moreover, GNP-RA could make use of the previous experiences in the form of rules to determine agents’ current action, which means that GNP-RA could learn from agents’ past behaviors. This also helps the current agent to take correct actions and improve its performance. Simulations on the tile-world demonstrate that GNP-RA could achieve higher fitness values and better generalization ability.

Chapter 3 aims to solve the perceptual aliasing problem and improve the performance for agent control in non-Markov environments. The perceptual aliasing problem refers to that different situations seem identical to agents, but different optimal actions are required. In order to solve this problem, a new rule-based model, "GNP with multi-order rule accumulation" (GNP-MRA) is proposed in this chapter. Each multi-order rule records not only the current environment information and agent's actions, but also the previous environment information and agent's actions, which helps agents to distinguish the aliasing situations and take proper actions. Simulation results prove the effectiveness of GNP-MRA, and reveal that the higher the rule order is, the more information it can record, and the more easily agents can distinguish different aliasing situations.

Chapter 4 focuses on how to improve the quality of the rule pool. Two improvements are made in order to realize this. One of them is that during the rule generation, reinforcement learning is combined with evolution in order to create more efficient rules. The obtained knowledge during the running of the program could be used to select the proper processing for judgments, i.e., better rules. The second approach is that after the rule generation, a unique rule pruning approach using bad individuals is proposed. The basic idea is that good individuals are used as rule generators and bad individuals are used to filter the generated bad rules. Four pruning methods are proposed and their performances are discussed and compared. After pruning the bad rules, the good rules could stand out and contribute to better performances. Simulation results demonstrate the efficiency and effectiveness of the enhanced rule-based model.

Chapter 5 is devoted to improving the adaptability of GNP-RA depending on the environment changes. GNP-RA has poor adaptability to the dynamic environments since it always uses the old rules in the rule pool for agent control. Generally speaking, the environment keeps changing all the time, while the rules in the rule pool remain the same. Therefore, the old rules in the rule pool become incompetent to guide agents' actions in the new environments. In order to solve this problem, Sarsa-learning is used as a tool to update the rules to cope with the inexperienced situations in the new environments. The basic idea is that when evolution ends, the elite individual of GNP-RA still execute Sarsa-learning to update the Q table. With the changes of the Q-table, the node transitions could be changed in accordance with the environment, bringing some new rules. These rules are used to update the rule pool, so that the rule pool could adapt to the changing environments.

Chapter 6 tries to improve the generalization ability of GNP-RA by pruning the harmful nodes. In order to realize this, "Credit GNP-RA" is proposed in this chapter. Firstly, Credit GNP-RA has a unique structure, where each node has an additional "credit branch" which can be used to skip the harmful nodes. This gene structure has more exploration ability than the conventional GNP-RA. Secondly, Credit GNP-RA combines evolution and reinforcement learning, i.e., off-line evolution and on-line learning to prune the harmful nodes. Which node to prune and how many nodes to prune are determined automatically considering different environments. Thirdly, Credit GNP-RA could select the really useful nodes and prune the harmful ones dynamically and flexibly considering different situations. Therefore, Credit GNP-RA could determine the optimal size of the program along with the changing environments. Simulation results demonstrated that Credit GNP-RA could generate not only more compact programs, but also more general rules. The generalization ability of GNP-RA was improved by Credit GNP-RA.

Chapter 7 makes conclusions of this thesis by describing the achievements of the proposed methods based on the simulation results.