Nested Evolution of an Autonomous Agent Using Descriptive Encoding

Jae-Yoon Jung
School of Computing
Queen's University
Kingston, ON, Canada
jung@cs.queensu.ca

James A. Reggia
Department of Computer Science
University of Maryland
College Park, MD, USA
reggia@cs.umd.edu

ABSTRACT

In this paper, we investigate the use of nested evolution in which each step of one evolutionary process involves running a second evolutionary process. We apply this approach to build a neuroevolution system for reinforcement learning (RL) problems. Genetic programming based on a descriptive encoding is used to evolve the neural architecture, while a nested evolution strategy is used to evolve the needed connection weights. We test this hierarchical evolution on a non-Markovian RL problem involving an autonomous foraging agent, finding that the evolved networks significantly outperform a rule-based agent serving as a control.

Categories and Subject Descriptors: I.2.6 [Artificial Intelligence]: Learning - Connection and neural nets, concept learning

General Terms: Experimentation

Keywords: descriptive encoding, neuroevolution, reinforcement learning, autonomous agent

1. INTRODUCTION

We explore the idea of using a hierarchical, or "nested", evolutionary process to design an effective neurocontroller for a situated agent in a reinforcement learning context. The general idea of *nested evolution* is that an outer/primary evolutionary process "calls" an inner/secondary evolutionary process to take care of a specific problem, much as a program calls a subroutine. We adopt a genetic programming (GP) method called descriptive encoding [2] to represent an agent's genome. During each generation of the primary GP process, an evolution strategy (ES) algorithm [1] is used to evolve connection weights for the specific architectures under consideration.

The specific goal of neuroevolution in this work is to evolve an agent that forages for food and avoids predators in a simulated artificial environment, a task that has seen frequent usage in artificial life research [3]. As a control measure, a rule-based agent that can remember previous environmental states, to compensate for the limited sensory information that agents receive, is defined and tested under varying environmental conditions.

2. METHOD

The initial population of the networks specified (genotypes) is built based upon the description file given by users, and each generation of network architectures is produced by using GP. During each generation of the GP process, for each evolved architecture an ES algorithm is used to identify reasonably optimal sets of connection weights.

The simulation environment consists of a two-dimensional, real-valued space. At the beginning of each simulation, a predefined number of predators and food sites are randomly placed throughout the environment, and a single agent is located in the center of the environment. The agent should evolve to run from predators if the predators find and chase the agent, and should forage for food sites as the initial energy level given to the agent is not sufficient for it to survive the simulation duration. The fitness of the agent is determined by its final energy level when the simulation ends and how long it survived. The agent can perceive predators and food sites within its visibility range, but the range is limited to π radius in its moving direction (i.e., the agent can't see predators chasing it from behind).

A rule-based agent was used as a control and it can be considered as a near-optimal agent in a sense that 1) it performs the required functionality of foraging and escaping; 2) chooses the optimal gait type according to the current state and velocity, thus minimizing energy consumption for the state; and 3) has a memory which has the previous environmental information, compensating for limited sensory input.

3. EXPERIMENTAL RESULTS

We demonstrated that it was possible to use this approach effectively in a reinforcement learning setting. The evolved recurrent networks outperformed a rule-based, pre-designed agent under various environmental conditions, indicating that evolution had discovered a recurrent neural network architecture plus appropriate weight values that compensate for a neural agent's lack of "working memory".

4. ACKNOWLEDGEMENT

This work was supported by NSF award IIS-0325098.

5. REFERENCES

[1] H. G. Beyer. *The Theory of Evolution Strategies*. Springer, Berlin, 2001.

[2] J. Y. Jung and J. A. Reggia. Evolutionary design of neural network architectures using a descriptive encoding language. *IEEE Transactions on Evolutionary Computation*, 10(6):676–688, Dec. 2006.

[3] J. A. Reggia, R. Schulz, G. Wilkinson, and J. Uriagereka. Conditions enabling the evolution of inter-agent signaling in an artificial world. *Artificial Life*, 7(1):3–32, 2001.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.
ACM 978-1-60558-130-9/08/07.