An Evolutionary Approach to Automatic Construction of the Structure in Hierarchical Reinforcement Learning

Stefan Elfwing[1,2,3], Eiji Uchibe[2,3], and Kenji Doya[2,3]

[1] KTH, Numerical Analysis and Computer Science department,
KTH, Nada, 100 44 Stockholm, Sweden
[2] ATR, Human Information Science Laboratories, Department 3
[3] CREST, Japan Science and Technology Corporation
2-2-2 Hikaridai, "Keihanna Science City" Kyoto 619-0288, Japan

1 Introduction

Hierarchical reinforcement learning (RL) methods have been developed to cope with large scale problems. However, in most hierarchical RL methods, an appropriate structure of hierarchy has to be hand-coded. This paper presents an evolutionary approach for automatic construction of hierarchical structures in RL.

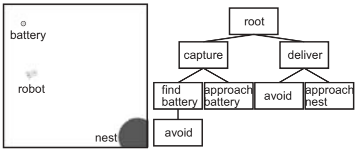
2 Proposed Method

Our method combines the MAXQ method [1] and Genetic Programming (GP). The MAXQ method learns the policy based on the hierarchy obtained by the GP, while GP explores the appropriate hierarchies using the result of the MAXQ method. Leaf nodes and inner nodes of MAXQ representation are regarded as terminals and functions for GP. We use strongly-typed GP [2] that allows the designer to assign specific types to the arguments and the return value of each function.

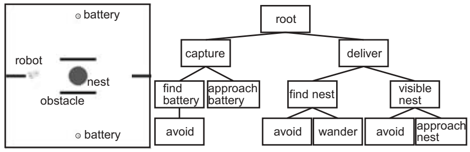
3 Task and Experimental Results

We have performed simulation experiments with a rodent like robot, *Cyber Rodent*. The task is to find, approach and capture a battery pack, and then return the battery pack to the nest. We have prepared three different environments shown in Fig. 1. Cyber Rodent has the distance sensors and the vision system. For all sensor readings in the simulated environment, noise (10 % of input range) is added. In this experiment, we prepared seven composite subtasks (`root`, `capture`, `deliver`, `find battery`, `visible battery`, `find nest`, and `visible nest`), and five primitive subtasks (`avoid`, `wander`, `approach battery`, `approach nest`, and `turn`). Each individual in the population performs a fixed number of trials in each generation. The fitness is calculated as the number of

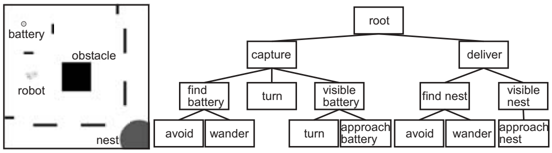
E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2723, pp. 507–509, 2003.



(a) E1: Simple environment



(b) E2: There are two battery packs, and the nest is placed in the center of the environment.



(c) E3: There are many obstacles, and the battery packs can not be observed when the agent is in the nest.

Fig. 1. Tested environments and examples of obtained hierarchies. The small filled light gray circles represent battery packs and the big darker gray circles represent the nests, respectively.

time steps to complete the task. The parent hierarchies for crossover are chosen by tournament selection.

Experimental results showed that GP found suitable hierarchies for all three environments. A remarkable finding was that the complexity of the obtained hierarchical structures was strongly constrained by the complexity of the environment, GP for a simple environment obtained a simple and specialized hierarchy and GP for a complex environment obtained a complex and general task hierarchy. The main difference between E1 and E2 was that the nest in E2 was surrounded by the wall although the battery packs were placed in an open field in both environments. Accordingly, the part of `deliver` in the obtained structure was different, as shown in Fig. 1(a) and (b). In E3, since there were many obstacles and the environment was crowded, the obtained structure was the most complicated.

4 Conclusion

We plan to implement our method to the real hardware. A foreseeable extension of this study is to generalize the method as a model of cooperative and competitive mechanisms of the learning modules in the brain.

Acknowledgments. This research was supported in part by the Telecommunicatios Advancement Organization of Japan.

References

1. T. G. Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.
2. D. J. Montana. Strongly Typed Genetic Programming. *Evolutionary Computation*, 3(2):199–230, 1995.