RCS: A Learning Classifier System for Evolutionary Robotics

Noah W. Smith
Colby College
Department of Computer Science
Waterville, ME USA
nwsmith@colby.edu

Clare Bates Congdon
Colby College
Department of Computer Science
Waterville, ME USA
ccongdon@colby.edu

ABSTRACT

This paper introduces RCS, a learning classifier system designed for evolutionary robotics research. In addition to describing the system, it will present the results of RCS applied to a pursuit task. In this test, performance was good and has been improved in ongoing work.

Catagories and Subject Descriptors: I.2 [Computing Methodologies]: Artificial Intelligence— *Learning, Robotics*

General Terms: Performance, Design, Experimentation

Keywords: Learning Classifier Systems, Robotics

1. INTRODUCTION AND BACKGROUND

In a robot system, sensors monitor the environment, a controller chooses actions based at least in part on those sensors, and effectors execute those actions. If a classifier system is chosen as the controller, the input-output chain sends abstracted sensor data to the classifier system, which returns motor actions as output. This process works very well for hand-crafted classifier sets in tasks where no dynamic behavior is required; however, if a learning classifier system (LCS) is desired as a controller, the issue of reinforcement must be addressed.

John Holland's original concept of a LCS[4] received input from the environment, then allowed matching classifiers to bid for the ability to fire. After one was chosen, it entered the "bucket brigade" reinforcement system to adjust its weight. Meanwhile, genetic operators were applied to existing classifiers to gradually change the rule set.

In most non-trivial robotics tasks, success depends on a long series of actions where, even if one classifier determines every individual action, it is very difficult to assign fitness to individual classifiers. Most LCSs based on Holland's original system are constructed with this reinforcement scheme. The most influential of this group include Wilson's ZCS[6], XCS[7], and Stolzmann's ACS[5]. An alternative is to evaluate groups of classifiers as a rule set which evolves as a whole over time. RCS attempts to create such a system specifically for evolutionary robotics research.

2. RCS SYSTEM DESIGN

The RCS algorithm is written in C and may be divided into two parts: the static core, specifying program control, classifier selection, and evolution, and the user-specified functions, which define the task to be completed. There are also a number of parameters that influence the performance of the system.

This element is responsible for running the correct number of trials and generations. A trial is defined as the execution of the task from beginning to end; each classifier set is allocated n trials in each generation. During each trial, the robot running RCS repeatedly updates its sensors, presenting input to RCS. A classifier is chosen and fired, enabling the robot to move through the world. Trials are run one after another until all the rule sets have been evaluated. Once this is accomplished, the average of their fitnesses is stored for use in the evolutionary selection process and the generation is complete. After all the generations have been completed, the final rule sets are written to disk.

Each classifier in a rule set is stored with the last generation it was altered and the number of times it has been fired since that change. The former piece of information is valuable for conflict resolution, while the latter is more for after-the-fact analysis. Also stored with each rule set is a default action which fires if no classifier matches.

The process of evolution begins with a tournament-style selection made on the individuals in the current population. The fitnesses of surviving individuals are then reset to zero. Crossover then occurs in three variations: rule sets swapping default behaviors, classifiers in the same rule set swapping actions, and rearranging the order of two classifiers within a rule set. Finally, random bits are mutated with each bit having the same probability of mutation.

It is possible to adapt RCS to most robotics tasks by merely editing four functions: one specifying what input is presented to the classifier system, one dictating how the output from the system is handled, another defining the condition(s) at which a trial is complete, and finally one indicating what data is to be written to disk for subsequent analysis.

The maximum number of generations, the population and tournament sizes, number of trials per generation, the size of classifier components, and the mutation and crossover rates may all be set by the user. In addition, there are flags that enable elitism, favor more specific rules, maximize or minimize the fitness function, and determine whether the system should start by loading saved classifiers.

3. A SIMPLE PURSUIT TASK

The predator-and-prey task, also known as the pursuit domain, emerged in 1986[2], and was originally designed with four predators and a single prey. The world was divided into a grid, and the agents were restricted to orthogonal movement. The target behavior for the predators was to capture the prey by surrounding it on four sides. The goal for the prey was to evade such capture, and it was found that even random movement sufficed in many cases. In addition, full knowledge of the world was known to all five agents. While originally not a robotics domain, this task has recently been gaining popularity in that community. Recent work[1, 3] has limited the information available to each participant, and also moved away from the rigid grid format, resulting in a partial-knowledge system of robotic agents with a full range of motion. In most cases, capture is now defined as a predator coming into contact with a prey.

A simple pursuit task presented a good first test for RCS because of the qualitative range in behaviors that can culminate in capture. There are very simple classifier sets that can succeed, as well as more complex ones which may continue to be refined as the system evolves. We executed the task on K-Team's Khepera robots, simulated in the Webots environment from Cyberbotics. This combination is common in the evolutionary robotics community, and demonstrates the challenges of the field well.

3.1 Setup

There are four predators and one prey in our system; the prey behavior is static, while the predators utilize RCS. This allows a relatively uniform opponent for the classifier system to base its fitness function on. This prey behavior is very simple: it moves in approximately straight lines, bouncing off the walls around the arena. It has a slight speed edge over the predators to insure that mediocre predator behaviors do not usually result in capture.

The predators employ RCS, feeding camera and infra-red data into the classifier system as input, and interpreting the resulting output as wheel speeds. Each rule set contains eight classifiers. RCS assigns three bits to the camera data, representing the angle to target. The IR data is handled in a slightly different manner: each of the six directions scanned by these sensors is allocated one bit. This is because each sensor is exclusive from all the others. These two elements make up the entire input.

Trial length is capped at 350 classifier selection cycles, although if the prey is captured before that time, the trial is terminated. The fitness value is defined as the sum of the number of cycles until trial termination and the average distance between the predators and the prey. The system was executed for 20 different runs at 100 generations each.

3.2 Results and Observations

Data recorded for this task consisted of minimum, maximum, and average fitness values for each generation. Also, the final populations were available for analysis, and observations were made as evolution progressed in each run. Over time, the average fitness gradually declined relative to the steady maximum and minimum values. This indicates an overall trend toward populations where more of the trials resulted in captures. This should not be interpreted as necessarily showing an increase in the depth of the rule sets, although that phenomenon often accompanied the trend.

The final populations of classifiers also illustrated interesting patterns. Rule sets that scored a fitness value of 275 or less in the final generation were collected into a group of exemplary rule sets. They averaged approximately 3 firing classifiers per set, with a range of 1-6. This demonstrates that even rule sets with only a few firing classifiers were effective, and that some sophistication did develop. Furthermore, examining the corresponding default actions provided a little more insight into the prevailing strategies. In forty percent of the runs, each evolved separately, the same default action could be found: very quick forward movement in a large-radius circle. These rule sets operated by turning in a large circle until the prey is seen, then using three or four classifiers to follow it.

4. CONCLUSIONS AND FUTURE WORK

While RCS completed the task satisfactorily, having only one trial per rule set per generation often produced misleading fitness values, harming the evolutionary process. Certain circumstances tend to create such inacuracies, such as accidental capture as a predator backs into the prey. The easiest solution is to enable more trials per generation and average the values; however, an increase from 1 to 2 doubles the runtime for an experiment, and time is often at a premium in evolutionary robotics. As a proof of concept, we ran RCS on Wilson's Woods1 environment[6] with varying numbers of trials per generation. The accuracy of the fitness function increased polynomially with the number of trials per generation, so that by 10 it was very good and by 25 it was almost perfect.

Unfortunately, raising this value from 1 to 10 is discouraging in general use with robotics tasks as described here, even when run in simulation. To make RCS more useful for those types of applications, an alternative solution would be nice. One possiblility would be to remember some remnant of a rule set's past fitness values and use it in the evolutionary process. Future work could largely consist of investigating this and other possibilities.

5. REFERENCES

[1] G. Bauson and T. Ziemke. Co-evolving task dependent visual morphologies in predator and prey experiments. In *GECCO*, pages 458–469, Chicago, IL, July 2003.

[2] M. Benda, V. Jagannathan, and R. Dodhiwala. On optimal cooperation of knowledge sources - an empirical investigation. Technical Report BCS-G2010-28, Boeing Advanced Technology Center, Boeing Computer Services, Seattle, WA, July 1986.

[3] D. Floreano and S. Nolfi. Co-evolving predator and prey robots: Do 'arm races' arise in artificial evolution? In *Artificial Life*, volume 4, pages 311–335, 1998.

[4] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975. Republished by the MIT press, 1992.

[5] W. Stolzmann. *Antizipative Classifier Systeme.* PhD thesis, Fachbereich Mathematik/Informatik, University of Osnabrück, 1997.

[6] S. W. Wilson. ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2(1):1–30, 1994.

[7] S. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.