ROBO-SHEPHERD: LEARNING COMPLEX ROBOTIC BEHAVIORS[†]

Alan C. Schultz, John J. Grefenstette, and William Adams

Naval Research Laboratory, Washington, DC 20375-5337, U.S.A.
schultz@aic.nrl.navy.mil

ABSTRACT

This paper reports on recent results using genetic algorithms to learn decision rules for complex robot behaviors. The method involves evaluating hypothetical rule sets on a simulator and applying simulated evolution to evolve more effective rules. The main contributions of this paper are (1) the task learned is a complex behavior involving multiple mobile robots, and (2) the learned rules are verified through experiments on operational mobile robots. The case study involves a shepherding task in which one mobile robot attempts to guide another robot to a specified area.

KEYWORDS: Genetic Algorithms, Evolutionary Computation, Robot Learning

INTRODUCTION

In this study, a complex robotic behavior, embodied as set of stimulus-response rules, is learned by a robot. Learning initially takes place under simulation. This methodology reflects our belief that many robotic tasks are too expensive or dangerous to learn from experiences on a real robot in its real environment. This work demonstrates that non-trivial behaviors learned under simulation can transfer to an operational mobile robot, with similar behavior and performance levels being achieved in the real environment. The methodology depends on the assumption that important aspects of the real world are captured by the simulator. This assumption should not be viewed as a severe limitation on the approach. Our experience shows that even simulators with limited fidelity are sufficient for learning many interesting tasks for real-world robots.

As an example, we report on learning a shepherding task, in which one mobile robot seeks to guide another to a specified area. The behavior is learned under simulation. The resulting rules are then used to control an operational mobile robot. The resulting behavior and performance level is compared to that achieved under simulation. Adding to the difficulty of this task, all perception of the environment by each operational robot is via the actual sensors, and the robots must learn to avoid collisions with other objects in the world.

THE SHEPHERDING TASK

In this task, one robot is the shepherd and the other robot is the sheep. The performance task is for the shepherd to guide the sheep into a pasture within a limited amount of time. The

[†]published in the Proc. of the International Symposium on Robotics and Automation, 1996, TSI Press, ASME.

sheep reacts to the presence of nearby objects by moving away from them. Otherwise, the sheep moves in a random walk. The shepherd must learn to control his own translation and steering to get the sheep to move into the pasture. Both robots are controlled by a reactive rule set that maps current sensors of each robot into appropriate motion commands. Only the shepherd's rules are learned.

THE LEARNING SYSTEM

The behaviors, which are represented as a collection of stimulus-response rules, are learned in the SAMUEL rule learning system. SAMUEL is a heuristic learning program that uses genetic algorithms and other competition-based heuristics to improve its decision-making rules. The system actively explores alternative behaviors in simulation, and modifies its rules based on this experience. SAMUEL is designed for problems in which payoff is delayed in the sense that payoff occurs only at the end of an episode that may span several decision steps. SAMUEL is particularly well-suited for learning strategies in competitive environments for tasks such as predator-prey problems, tracking problems, and other models involving multiple competing agents.

SAMUEL includes a competition-based production system interpreter, incremental strength updating procedures to measure the utility of rules, and genetic algorithms to modify strategies based on past performance. This system incorporates a convenient language for the expression of tactical decision rules, a graphical interface, and a number of heuristics for rule modification. The rule language in SAMUEL also makes it easier to incorporate existing knowledge, whether acquired from experts or by symbolic learning programs.

In SAMUEL, learning is driven by competition among knowledge structures. Competition is applied at two levels: Within a strategy composed of decision rules, rules compete with one another to influence the behavior of the system. At a higher level of granularity, entire strategies compete with one another using a genetic algorithm.

Previous papers on SAMUEL focused on the operations of the system in purely simulated environments [2][3]. We have also previously reported on using this method to learn simple robot behaviors such as navigation and collision avoidance [7][8].

Representation

Each stimulus-response rule consists of conditions that match against the current sensors of the robot, and an action that suggests a translation or steering velocity command to the robot based on the current situation. The condition and action values are described in more detail below. For example, a rule for the shepherd might be:

IF range = [35, 45] AND bearing = [340, 35] THEN SET turn = -24 (Strength 0.8)

During each decision cycle, all the rules that match the current state are identified. Conflicts are resolved in favor of rules with higher *strength*. Rule strengths are updated based on rewards received after each training episode. See [2] for further details.

LEARNING UNDER SIMULATION

The rule set for the shepherd is learned under simulation while the sheep's rule set is fixed. The learning task requires the shepherd to get the sheep to within a fixed range of the goal within a time limit, without hitting an obstacle (the enclosing walls of the simulated environment or

the sheep). For learning, a differential performance score is given (the fitness) for each rule set in the population. The performance of an individual rule set is calculated by averaging over 20 episodes, in which each episode begins with the sheep and the shepherd placed in random initial positions and orientations, and end when the sheep enters the pasture, time expires, or a collision occurs. The performance of each episode is rated by an automated critic that gives full credit for getting the sheep into the pasture, partial credit for getting it close, and no credit for episodes that end due to collision with the sheep or with the surrounding walls. In these experiments, the pasture remains in a fixed location for all trials.

After each individual in the population is evaluated in simulation, genetic and other operators in the Samuel system are applied to the individuals to generate the next population of behaviors, presumable with higher fitness. This cycle is iterated until the desired fitness is achieved, or until a fixed upper limit of generations have elapsed.

TESTING ON THE ROBOTS

In order to verify the learned behaviors, the learned rules are used to control the actual shepherd robot. The shepherd robot and the sheep robot are placed in random locations and orientations within our laboratory environment, and the resulting performance is recorded.

The two robots used in this study are Nomad 200 systems manufactured by Nomadic Technologies, Inc., and include 16 sonars, 16 infrared sensors (not used in this study), tactile sensors, and a structured light range finder which is described below.

THE SHEPHERD ROBOT

For this task, five abstract sensors are defined for the shepherd: the range and bearing from it to the sheep, the sheep's heading with respect to the sheep's bearing, and the range and bearing to the center of the pasture to which it is herding the sheep. Conceptually, these abstract sensors are shown in figure 1. These abstract sensors are derived from information available through the actual sensors on the robots.

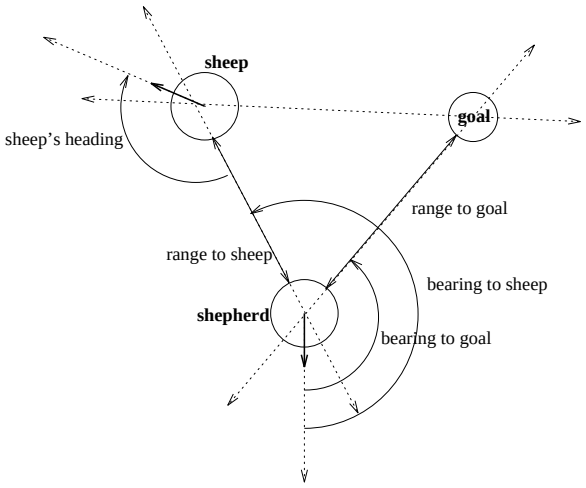

Figure 1: The meaning of the shepherd's sensors

The range, bearing and heading are discretized. The bearing values are partitioned into intervals of five degrees. The range is partitioned into intervals of five inches from 0 to 75

inches. The heading is partitioned into 45 degree segments.

The actual perception system, a structured light rangefinder, has a limited reliable detection range of approximately 8 feet, with the chance of losing the sheep increasing as the range increases. Under simulation, the true range and bearing are determined, and then are discretized as above. To simulate the shepherd's limited perception range its detection of the sheep is restricted to 6 feet.

In addition to the range, bearing and heading of the sheep returned by the shepherd's perception system, the shepherd also knows the coordinates of the pasture to which he wants to herd the sheep. In actual operation, the shepherd uses dead reckoning to determine this location. In the simulation, the true values are determined, and then discretized as above.

The learned actions are the velocity mode commands for controlling the translational rate and the steering rate. The translation is given as -4 to 16 inches/sec in 2 inch/sec intervals. The steering command is given in intervals of 4 degrees/sec from -24 to 24 degrees/sec.

Description of the shepherd's perception system

The range and bearing to the sheep, and the heading of the sheep are determined on the real robot by using a perception system mounted on the shepherd robot. Perception is performed by an on-board triangulation-based rangefinder. A laser on the robot projects a plane of laser light parallel to the floor and the intersection of that plane with any objects is detected by a video camera mounted above the laser. The positions of the illuminated pixels in the image are then mapped into Cartesian coordinates.

To provide both identification and pose information of the sheep robot, an irregular, extruded pentagon is placed on top of the sheep and a mathematical description of it is given to the shepherd. The pentagon vertices have five distinct angles. Software on the shepherd robot's processor continually fits lines to the points returned by its rangefinder, and measures the angles between the adjoining lines. Observation of a single vertex identifies the object as the sheep and, in conjunction with the poses of the adjacent lines, determines the relative orientation and position of the pentagon and therefore the sheep.

When the sheep is in rangefinder view, the shepherd attempts to center the visible vertex in its view. When not in view, the shepherd interpolates the sheep's relative position based on the sheep's last observed position and velocity and the shepherd's known position. An arc is defined encompassing the interpolated position, and the rangefinder is panned through the arc repeatedly. If the sheep is not observed again after a reasonable amount of time, it is considered lost and the shepherd spins the rangefinder in complete circles until the sheep is re-acquired.

THE SHEEP ROBOT

The sheep is controlled by a manually generated reactive behavior (rule-set). In general, its behavior is a random walk when no other objects are within a predetermined distance (150 inches in these experiments). Objects that are within this range tend to make the sheep move away from the object. The sheep makes its decisions based on two abstract sensors, namely, the range and bearing to the closest object within 150 inches. Outside of that distance, these sensors return the value UNKNOWN.

On the actual sheep robot, the 16 sonars are used to determine the range and bearing to the closest object. The minimum value is obtained from the sonars. If the value is less than the specified maximum range of 150 inches, then we determine which sonar returned that value, and then discretize the bearing and range. If the minimum range returned by all sonar sensors is over 150 inches, then we return the value UNKNOWN for the range and the bearing. As in

	total	percent
Successful episodes	19	67%
Failure due to time limit	1	3%
Failure due to collision	6	20%
Failure due to lost perception	3	10%
Failure due to lost communication packets	1	3%

Table 1: Summary of results with real robots

the shepherd, the range is discretized by partitioning into intervals of five inches from 0 to 150 inches. The bearing is partitioned into intervals of five degrees. In the training simulator, the true bearing to the closest object is calculated, and then discretized as above.

The actions of the sheep are limited to a translational rate of -4 to 6 inches/sec and a steering rate of -24 to 24 degrees/sec.

RESULTS FROM LEARNING

In the simulation, the best shepherd behavior reached a success level of 86%. This rule set (from generation 250 in the first SAMUEL run) was then used to control the real shepherd robot. Using the operational robots, 27 episodes were performed. At the start of each episode, the shepherd and sheep robots were placed in random orientations and positions within a restricted portion of the room, similar to the simulation episodes. A success was counted if the shepherd guided the sheep to the pasture area, defined as a three foot radius area in another part of the room, within a fixed amount of time (60 secs). A failure was counted if the time was exceeded, if either the sheep or the shepherd collided with any other object in the environment, if the shepherd's perception system stopped tracking the sheep, or the robot's communication link with the host computer failed. In addition to the actual success rate, observations were subjectively made of the shepherd's behavior to determine if the behaviors were similar to those in the simulator.

The results are summarized in Table 1. The actual robot succeeds in 67% of the episodes. In these results, failures include the shepherd losing track of the sheep, as well as communication failures. However, in the simulation, it is not possible for either of these conditions to occur. If these cases are removed from consideration, then the observed success rate is 73%, much closer to the success rate obtained under simulation.

Subjective observations suggest that the same types of behaviors are exhibited by the real robot as in the simulation. For example, in both cases, the shepherd tends to maneuvers to the outside of the sheep and forces it toward the pasture. Videos of the operational robots will be shown at the Conference.

RELATED WORK

Other approaches to evolutionary robotics have been used before. The work of (Harvey, Cliff and Husbands) [4] has concentrated on bottom-up learning; for example, the perception system is learned as a neural network via evolutionary algorithms. Our approach differs in that we take an engineering view, and do not insist that all components must be learned. We start with

designed parts, and with a specific decomposition of behaviors, and let the system learn the rules for each behavior.

Although similar in that robot behaviors are being learned, Dorigo [1] takes an entirely different architectural approach to evolution, using classifier systems to learn behaviors. Here the entire population of the genetic algorithm is taken as the behavior. In our work, each individual in the population is a behavior, and the population consists of competing behaviors.

In the system GA-Robot by Ram *Et. Al.* [6], a genetic algorithm with a floating point representation is used to optimize parameters that effect the behavior. In Samuel, the entire behavior is learned in a high-level stimulus-response language.

CONCLUSIONS AND FUTURE WORK

This paper reports on an approach for learning behaviors for mobile robots by evaluating hypothetical rule sets on a simulator and applying simulated evolution to evolve more effective rules. The resulting rules are then placed on an actual robotic system for testing. The learning system, SAMUEL, uses genetic algorithms applied to symbolic rules. We have previously reported on using this method to learn simple robot behaviors such as navigation and collision avoidance [7][8]. Here, we have demonstrated that the approach can be used to learn rules to perform a complex herding behavior involving multiple mobile robots, and that the learned rules can be verified through experiments on operational mobile robots.

Future work will aim at scaling these approaches to more complex multi-robot tasks, including cooperative mapping of an area, finding hidden objects and performing cooperative surveillance tasks.

References

[1] Dorigo, M. (1993). "Genetic and Non-Genetic Operators in Alecsys," *Evolutionary Computation*, 1(2): 151-164.

[2] Grefenstette, J. J., Ramsey, Connie L., and Schultz, Alan C., (1990). "Learning sequential decision rules using simulation models and competition," *Machine Learning*, **5(4)**, 355-381

[3] Grefenstette, J.J. (1991). "Lamarckian learning in multi-agent environments," *Proc. Fourth International Conference of Genetic Algorithms*, San Mateo, CA: Morgan Kaufmann, 303-310.

[4] Cliff, D., I. Harvey, and P. Husbands (1991). Cognitive Science Research Paper No. 318, School of Cognitive and Computer Science, University of Sussex.

[5] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* Univ. Michigan Press, Ann Arbor, 1975.

[6] Ram, Ashwin, R. Arkin, G. Boone, and M. Pearce (1994). "Using Genetic Algorithms to Learn Reactive Control Parameters for Autonomous Robotic Navigation," *Adaptive Behavior*, 2(3), 1994

[7] Schultz, Alan C. and Grefenstette, John J. (1992). "Using a genetic algorithm to learn behaviors for autonomous vehicles," *Proceedings of the of the AIAA Guidance, Navigation and Control Conference*, Hilton Head, SC, August 10-12, 1992.

[8] Schultz, Alan C. (1994). "Learning robot behaviors using genetic algorithms," *Intelligent Automation and Soft Computing: Trends in Research, Development, and Applications, v1*, Mohammad Jamshidi and Charles Nguyen, editors, Proceedings of the First World Automation Congress (WAC '94), 607-612, TSI Press: Albuquerque.