Evolving Cooperation Strategies for Multiple Mobile Robots

Yeon-Jun Hong
Dept. of Computer Engineering
Konkuk University
Seoul 143-701, Korea
yjhong@ai.konkuk.ac.kr

Byoung-Tak Zhang
Dept. of Computer Engineering
Seoul National University
Seoul 151-742, Korea
btzhang@comp.snu.ac.kr

ABSTRACT

This paper addresses the evolution of collective behavior of autonomous mobile robots for transporting a large table in teamwork. Several design decisions are discussed to solve this problem by evolutionary computation. These include team diversity, breeding policy, coordination mechanisms, and granularity of primitive behaviors. We present a method for evolving cooperation strategies of a homogeneous team in which the individuals are cloned to form a team which uses deictic sensing. Asynchronous recurrent neural networks are used to control the motions of robots. Simulation results are provided to demonstrate the feasibility of this approach.

1. Introduction

Some tasks can be done faster or more easily by dividing it up among many agents. Other tasks may not only be solved better by using multiple agents, but can only be effectively solved, by using teams of agents working together. Teams can be characterized in terms of three attributes: team diversity, breeding policy, and coordination mechanisms [4].

In terms of diversity, teams could consist of clones of single individuals (homogeneous team) or could also consist of distinct individuals (heterogeneous team). In breeding policy, individuals may be cloned to form teams or interbreeding could be allowed between members of different teams. Coordination mechanisms in a team can use either name-based sensing, deictic sensing, or no sensing.

In name-based sensing, each agent has access to the positions of each distinct fellow agent, referenced by agent name. In the deictic sensing approach, sensing is allowed only relative to the agent, for example, "nearest neighbor agent," or "front of the pack." In the no sensing approach, team members evolve to coordinate with one another in a hard-coded way.

In this paper we suggest a method for evolving cooperation strategies of a homogeneous team in which the individuals are cloned to form a team which uses a deictic sensing approach. The application domain is the transport of a large table. Cooperation strategies for a team of four mobile robots are evolved in the form of recurrent neural networks which are activated asynchronously. To evolve a complex behavior we have used an increased-complexity learning method in which the entire task is decomposed into several subtasks and learning starts with a simple one, with more complex ones (along with the simpler ones) learned after completing previous subtasks.

The paper is organized as follows. In Section 2, the table-transport task is described in more detail. Section 3 illustrates the architecture of the neural network and the method of increased-complexity learning. Secion 4 reports the simulation results and Section 5 contains the conclusion.

2. The Task: Table Transport

The application we chose involves transporting a table. This task cannot be performed by a single small robot; it requires multiple robots to cooperate. The workspace of the robot is a plane of 20m $\times$ 20m on which the robots move around. Figure 1 shows the workspace occupied by the table and robots to transport it. We assume cylinder robots of 20cm in diameter. It has four sensors positioned 45 degree upward. One move action of the robot brings its body 5cm to the direction it heads at the time. The sight of robots is limited to 2m. An additional sensor detects the target object within 10m. Sensors are connected to the input nodes of a neural network that determines the effector values for move actions. The initial environment contains three obstacles which are located randomly between the robots and the table.

Before we describe our approach, we first review related work. In particular, we focuse on approaches to

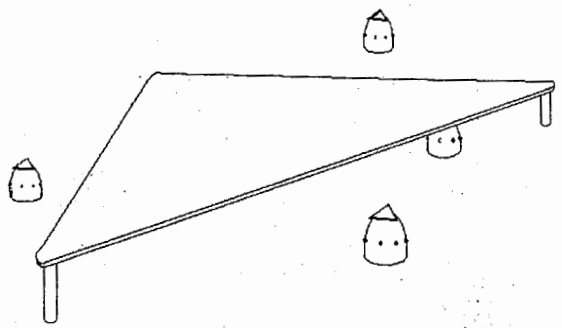

Figure 1: The robots in motion to transport the table.

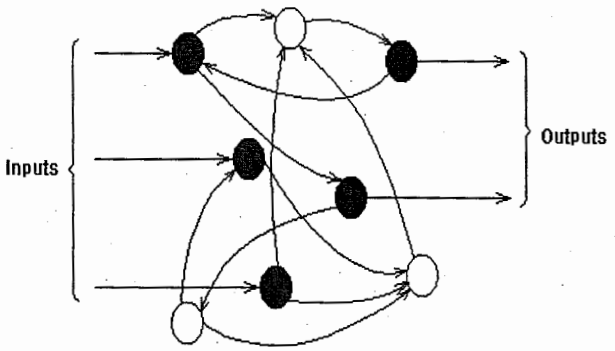

Figure 2: Recurrent neural network architecture.

the evolution of cooperation and communication between multiple mobile agents. Reynolds [5] used genetic programming to evolve "critters" which reacted with a herd instinct to outside preditors. He evolved a single controller which moved each critter based on its position and information about its neighbors and preditors. Each critter used this same controller algorithms to make movement decisions.

Haynes *et al.* [2] proposed a genetic programming method to generate a team by considering the whole team as one individual. They considered the subtrees of a genetic program as subindividuals within a main individual, each of which may serve a distinct specialized purpose. Luke and Spector [4] implemented heterogeneous breeding strategies which is functionally very similar to the clear method of producing homogeneous individuals. In both cases standard genetic programs are grown, and only one "individual" is tested at a given time. For homogeneous teams, individuals are tested by cloning them to form teams, and the resulting teams are tested in the environment. Heterogeneous teams, however, are formed from the individual's collection of subtreess.

3. The Method

3.1. Increased-Complexity Learning

The entire task of table transport is a combination of subtasks of finding, lifting, and moving the table. Learning the entire task using just one fitness function seems too complex and we use the increased-complexity learning strategy; it is a relatively well-known fact in psychology that students learn more effectively when they are given simple tasks and later more complex tasks. Similarly, we first evolve robots that show herding, then evolve robots that can do both herding and lifting, and finally evolve ones that perform the entire task. Each of the subtasks are briefly described in the following.

In finding the table, the robots should move in group to find the table. Each of them should position within a short distance from others and, at the same time, avoid collisions with other colleagures. Lifting the table requires synchronization which in turn assumes an explicit communication among the robots. The lifting behavior also requires a proper distribution of labor among the member robots. If a lift trial fails, the robots need resynchronization to trigger lifting behavior again. Transporting the table involves moving the table to the direction which is determined by the majority of the force in the group.

3.2. Asynchronous Recurrent Neural Networks

The behavior of robots are controlled by recurrent neural networks, like as shown in Figure 2. The neuron type we adopted is a linear threshold unit. It first computes the weighted sum of inputs from the environment or other units. If the sum s exceeds the threshold θ, then it produces an output in proportion to the sum. More formally, the activation of each unit is determined as:

$$y(s) = \begin{cases} 0 & \text{if } s < \theta \\ c \cdot s & \text{if } \theta \leq s \leq s_{max} \\ y_{max} & \text{if } s > s_{max} \end{cases}$$

where c, s_{max}, and y_{max} are constants.

The activations of units are updated asynchronously. The final output of the network is determined after a fixed number of activation updates. In the experiments the number of updates was varied between 1 to 100. The asynchronous update of activations increases the memory capacity of the network, which is an advantage over the more conventional feed-forward networks.

Networks also learn using a Hebbian rule; the connection weight of two adjacent units is increased when they simultaneously activate. The Hebbian learning rule has proved efficient in single-layer networks, but its capability remains unexplored in recurrent networks. According to the Hebbian rule the weights may increase without bound. To solve this problem, we used a weight-decay method in which the connection weights are decayed a fixed rate at every weight update.

inputs	usage
pressure	direction of move
table	finding, lifting
obstacle	collision avoidance
comm. signal	synchronization
stagnation	direction of movement

Table 1: Sensor inputs to neural networks.

outputs	usage
left/right motor position	motor drive
lift action	lifting
infrared signal	communication

Table 2: Outputs from the neural network for robot control.

The combination of asynchronous activation and Hebbian learning has a special meaning from the viewpoint of agent theory. Each unit in the network is considered as autonomous agents and the final output of the network can be viewed as the result of collective behavior of the individual agents. This aspect is especially interesting from artificial life.

4. Simulation Results

Table 1 lists the sensory inputs needed to control the robots. They include the pressure for measuring the unbalance of weight of the table among the member roobts, the detectors for the table and obstacles, a communication signal for synchronization of group behavior, and a stagnation detector for the determination of the direction to move to. From these inputs the network produces the outputs listed in Table 2. They represent the values for driving the left/right motors of the wheels and the signal to send for communication necessary for finding and lifting behavior.

In the simulations described below, we have used as input to the network the detectors for the table, colleagues, and obstacles. The neural network has 9 output units, 8 for determining the direction and the rest for non-moving.

The fitness of each network is evaluated by implanting it into the four robots. Each robot is given a bucket of points at the outset. The robots are allowed to move a fixed maximum number of steps. At each movement, the badness of their behavior evaluated and penalized. When it collides, for example, with its colleagues, it gets K_{coll} points subtracted from the bucket. An amount of K_{away} points is also subtracted when it moves too far away from its colleagues. This factor encourages herding behavior. When the table is out of sight of the robot on each move action, the robot gets K_{sight} points decreased. In effect, the sum of the penalties on each movement is

$$\begin{aligned} A &= K_{away} \cdot \text{NumAways} \\ &\quad + K_{coll} \cdot \text{NumCollisions} \\ &\quad + K_{sight} \cdot \text{NumOutsights} \end{aligned} \tag{1}$$

where NumAways is a count for the number of being far away from the colleagues, and NumCollisions and NumOutsights are counts for collisions and being too far away from the table.

To encourage the movements of the robots, the resulting bucket of points is multiplied by the following factor

$$S = K_S \cdot \text{NumStepsMoved} \tag{2}$$

where K_S is a constant amd NumStepsMoved is the total number of steps moved. This term penalizes the robots that stay at the same location or move seldom.

The bucket of remaining points is subtracted again by a fractional amount of the distance from the table:

$$D = K_D \cdot \text{FinalDisplacement} \tag{3}$$

where K_D is a constant. This is to promote moving toward the table.

Overall, the fitness of a robot or its neural network is:

$$F_i = (\text{Bucket} - A) \times S - D \tag{4}$$

where A, S, and D are defined as above.

After being evaluated their fitness, the individuals are selected to be parents for recombination. The selection probability of each individual is given as:

$$R_i = \frac{F_i - F_{min}}{F_{max} - F_{min}} \tag{5}$$

where F_{max} and F_{min} are, respectively, the maximum and minimum fitness values of the individuals in the current population. We used steady-state selection, i.e. 5% of the population are replaced after each evaluation. Thus the population evolves more slowly but continuously than in generational selection.

Simulations have been performed distributed on a number of stand-alone machines. We performed preliminary experiments to determine the algorithm parameters and then started main experiments.

Figure 3 depicts the network architecture evolved after 200 generations. Analysis of the structure evolved in the network is being performed. Figure 4 shows the best and average fitness at each generation during evolution. Figure 5 demonstrates the behavior of robots whose neural network was evolved after 200 generations. As generation goes on, we could observe improvement in robots' herding behavior compared to that of the robots at early generations.

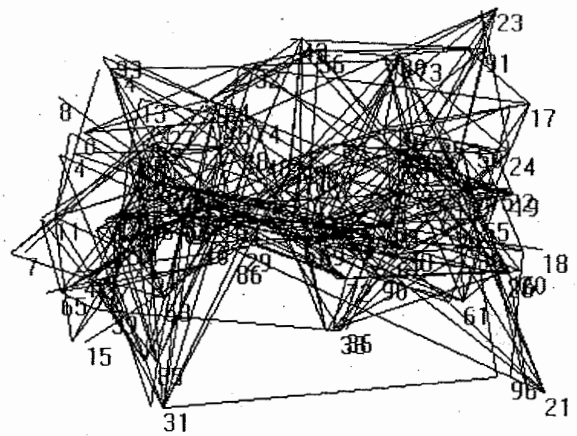

Figure 3: The networks evolved after 200 generations.

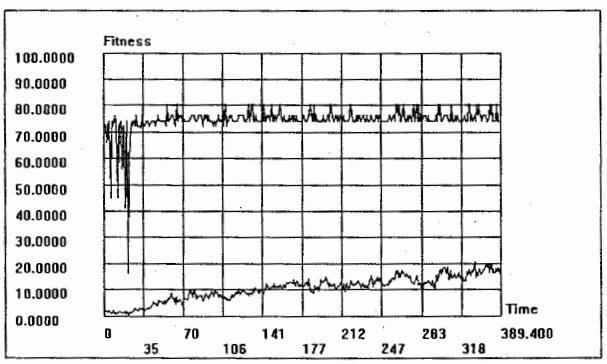

Figure 4: Change of the best and average fitness values.

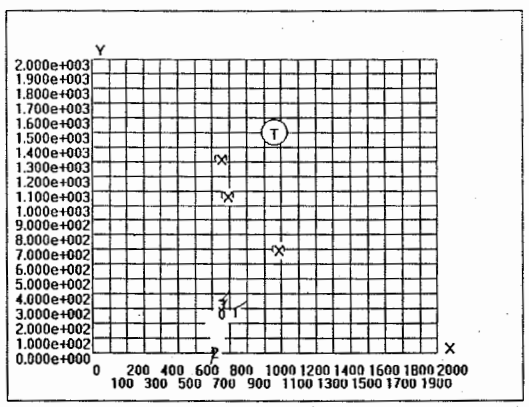

Figure 5: The simulation environment showing the robots moving in group to transport the table.

5. Summary and Future Work

We have described a method for evolving collective behavior of a group of small mobile robots and demonstrated its performance in the table-transport problem. Due to the complexity of the problem, we took the increased-complexity learning approach and experimented in this paper with part of the entire task. Future work should extend and refine the current framework. For instance, to be more realistic, the robots will need more sensors than used in the present simulations. Another improvement involves the development of more efficient learning algorithms that evolve recurrent neural networks to control the collective behavior of the robots. The idea of asynchronous activation update combined with Hebbian learning in recurrent networks seems interesting enough to pursue further research.

Acknowledgments

This research was supported in part by a grant from the Korea Science and Engineering Foundation (KOSEF) under contract number 961-0901-001-2.

References

[1] Ronald C. Arkin and J. David Hobbs, "Dimensions of communication and social organization in multi-agent robotic systems," *Proc. Second Int. Conf. on Simulation of Adaptive Behavior*, pp. 486-493, 1993.

[2] T. Haynes, S. Sen, D. Schoenfeld, and R. Wainwright, "Evolving a team," *Proc. AAAI-95 Fall Symposium on Genetic Programming*, pp. 23-30, AAAI Press, 1995.

[3] C. Ronald Kube and Hong Zhang, "Collective robotic intelligence," *Proc. Second Int. Conf. on Simulation of Adaptive Behavior*, pp. 460-468, 1993.

[4] Luke, S. and Spector, L., "Evolving teamwork and coordination with genetic programming," in *Proc. 1996 Genetic Programming Conf.*, J.R. Koza *et al.* (eds.) Cambridge, MA: MIT Press, pp. 150-156, 1996.

[5] Reynolds, C. W., "An evolved, vision-based behavioral model of coordinated group motion," in *Proc. Second Int. Conf. on Simulation of Adaptive Behavior*, J.-A. Meyer *et al.* (eds.) Cambridge, MA: MIT Press, pp. 384-392, 1993.

[6] Gregory M. Werner and Michale G. Dyer, "Evolution of herding behavior in artificial animals," *Proc. Second Int. Conf. on Simulation of Adaptive Behavior*, pp. 393-399, 1993.

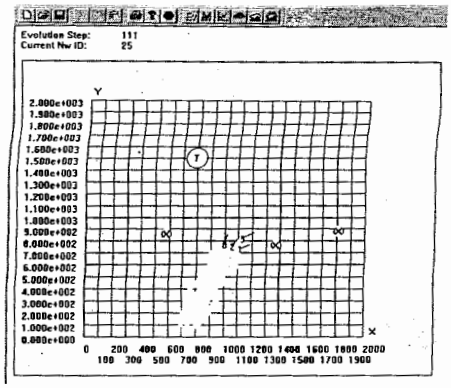

Figure 6: sim.ps file

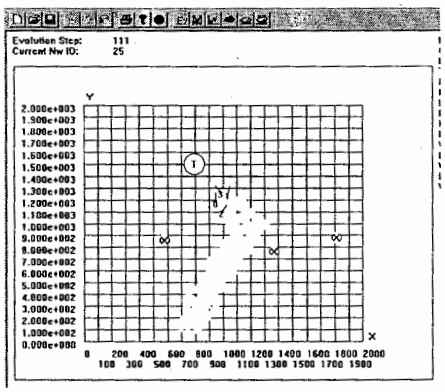

Figure 7: sim2.ps file

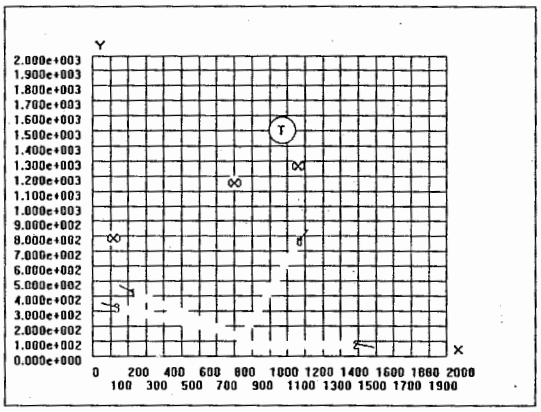

Figure 9: The dispaly of the simulation environment showing the robots moving in group to transport the table.

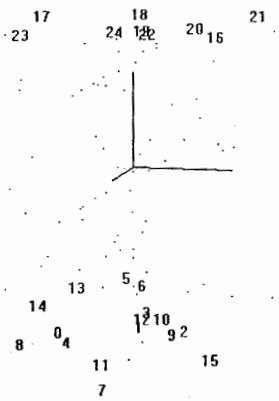

Figure 8: The network architecture evolved after 200 generations. For clarity, the connections are not shown here.