A Multinet Neural Architecture for Evolving Collective Robotic Intelligence

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

A novel neural architecture is designed for cooperative group motion of multiple mobile robots. The architecture consists of a recurrent network followed by a self-organizing map to process and detect the spatio-temporal patterns of sensory inputs. Motor outputs are generated by a classification network attached on top of the self-organizing map. An evolutionary algorithm is used to evolve the structure and weights of the networks for goal-directed collective behavior. Simulations have been performed on the task of transporting a large table in teamwork to demonstrate the feasibility of this approach.

1 Introduction

We study the use of neural networks for the evolution of cooperative behavior of autonomous mobile robotic agents. 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. The global behavior of the group of mobile agents is determined by the local interactions of their constituent parts. These interactions merit careful study in order to understand the global behavior of the group. In natural systems, such interactions resulted in the evolution of complex and stable behaviors that do not lend themselves to traditional, top-down style of analysis [Mataric, 1993].

In this paper we present a novel neural architecture designed for cooperative group motion of multiple mobile robots. Robots are cloned to form a homogeneous team. Cooperation strategies for a team of the robots are represented in the neural architecture consisting of multiple neural networks. Evolutionary algorithms are used to evolve the multinet architecture fitted for goal-directed group behavior. The application domain is the transport of a large table.

The paper is organized as follows. Section 2 motivates the approach by describing the cooperative work in a group of mobile robots. Section 3 presents the multinet neural architecture. Section 4 describes the method and experimental results for evolving the neural networks for collective robotic behavior.

2 Teamworking Robots

The task is to transport a large 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.

The sensory inputs needed to control the robots include the pressure for measuring the unbalance of weight of the table among the member robots, 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. The outputs for motor control include the values for driving the left/right motors of the wheels and the signal to send for communication necessary for finding and lifting the table.

The problem of cooperation and communication between multiple mobile agents has recently been studied by many researchers. Haynes *et al.* proposed a genetic programming method to generate a team by considering the whole team as one individual [Haynes *et al.*, 1995]. 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 implemented heterogeneous breeding strategies which is func-

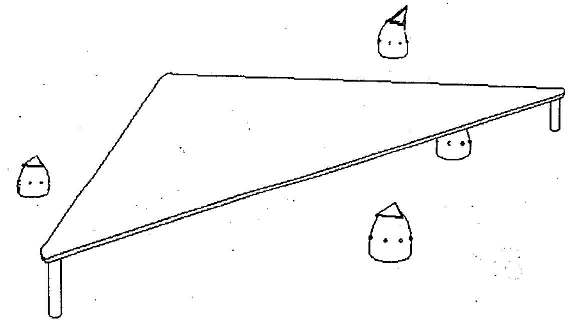

Figure 1: The robots in motion to transport the table.

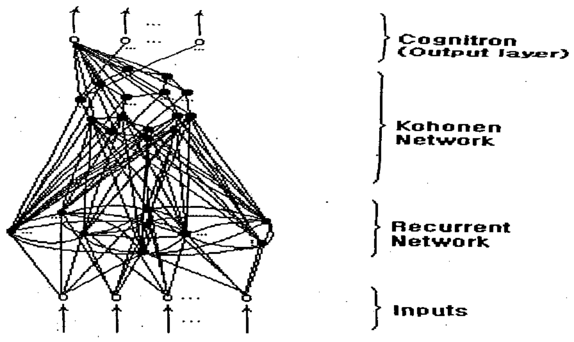

Figure 2: The multinet neural architecture.

tionally very similar to the clear method of producing homogeneous individuals [Luke and Spector, 1996]. 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 subtrees.

3 The Multinet Neural Architecture

The network consists of four layers (Figure 2). The bottom layer is the input layer that receives sensory inputs. The next three layers are the recurrent neural network (RNN) layer, the self-organizing map (SOM) layer, and the Cognitron output layer.

The recurrent neural network receives the environmental inputs and processes them to detect the spatio-temporal pattern of the inputs. It contains both excitatory and inhibitory neurons. The recurrent network used in the experiments contains 84 neurons in a virtual cylinder. The neurons are connected with other neurons in the neighborhood more frequently than the neurons in the distant area within the cylinder. We adopted a neuron model with chaotic dynamics [Adachi and Aihara, 1997]. The activation is determined by:

$$
\begin{aligned}
x_i(t+1) = f\Bigg[& \sum_{j=1}^{M} v_{ij} \sum_{d=0}^{t} k_e^d A_j(t-d) \\
& + \sum_{j=1}^{N} w_{ij} \sum_{d=0}^{t} k_f^d x_j(t-d) \\
& - \alpha \sum_{d=0}^{t} k_r^d g\{x_i(t-d)\} - \Theta_i \Bigg]
\end{aligned} \qquad (1)
$$

where v_{ij} and w_{ij} are synaptic weights to the ith constituent neuron from the jth external input and from the jth constituent neuron, respectively, and k_e, k_f, and k_r are the decay parameters for the external inputs, the feedback inputs, and the refractoriness, respectively. We set $k_e = k_f = k_r$ and $g\{x_i(t-d)\} = x_i(t-d)$. The recurrent network considers the spatio-temporal summation of both external inputs and feedback inputs from other chaotic neurons.

The neurons in the recurrent network learn their weights for each input data. The modified Hebbian learning rule is used to adapt the weights:

$$
\begin{aligned}
w_{ij}(t+1) = \; & w_{ij}(t) \\
& + r\{1 - w_{ij}(t)\} h\{x_i(t), x_j(t)\} \\
& - K\{w_{ij}(t)\}
\end{aligned} \qquad (2)
$$

where $K\{w_{ij}(t)\}$ is the weight decay function. The parameters are determined by an evolutionary method. For each input data, the network is activated iteratively to converge to a pattern or oscillate between two or more patterns. For our application, it is useful if the network produces similar output patterns for similar input patterns.

The next layer is the self-organizing map (SOM). Its objective is to recognize the activation pattern of the recurrent network and classify it into a class. The SOM learns for every update of activation at the recurrent network layer:

$$
w_{ij}(t+1) = w_{ij}(t) + r_{SOM}\{a_j - w_{ij}(t)\} \qquad (3)
$$

This implies that the SOM layer learns to recognize the activation patterns of the recurrent network. If the RNN converges to a single pattern, then SOM recognizes it. The ultimate winner of the SOM is the unit whose weight vector has the smallest distance to one of the activation patterns at RNN.

The output layer determines the output of the network based on the winner of the SOM network. This layer is trained using reinforcement learning as follows:

$$
y(s) = \begin{cases} w_{cij}(t) + r_c\{1 - w_{cij}(t)\} \cdot s \cdot a_j & \text{if } s \geq 0 \\ w_{cij}(t) + r_c \cdot w_{cij}(t) \cdot s \cdot a_j & \text{otherwise} \end{cases}
$$

where $s, -1 \leq s \leq 1$, is the normalized difference between the present penalty and the previous one. a_j is the activation value of the winner at the SOM layer.

4 Evolving Multinets for Robotic Teamwork

In the simulations described below, we have used as input to the network the detectors for the table to transport, colleagues, and obstacles. The 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 is 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

$$\begin{aligned} A &= K_{away} \cdot \text{NumAways} \\ &\quad + K_{coll} \cdot \text{NumCollisions} \\ &\quad + K_{sight} \cdot \text{NumOutsights} \end{aligned} \tag{4}$$

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

$$S = K_S \cdot \text{NumStepsMoved} \tag{5}$$

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:

$$D = K_D \cdot \text{FinalDisplacement} \tag{6}$$

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:

$$F_i = (\text{Bucket} - A) \times S - D \tag{7}$$

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:

$$R_i = \frac{F_i - F_{min}}{F_{max} - F_{min}} \tag{8}$$

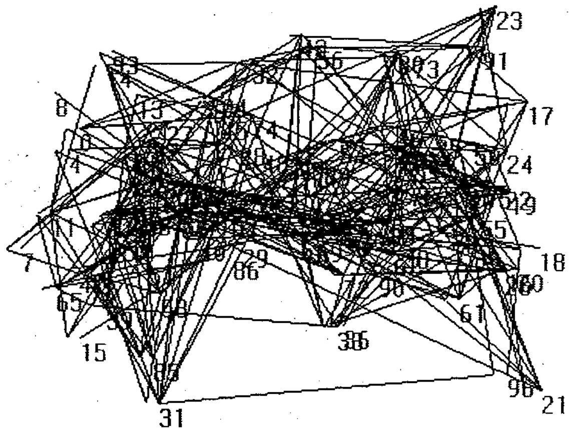

Figure 3: A recurrent neural network evolved after 200 generations.

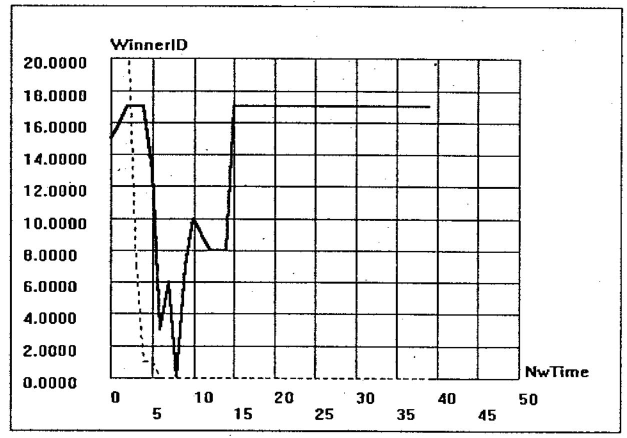

Figure 4: Change of the winner at the SOM layer while the RNN updates its activation. The dotted line shows the distance between the input vector and the weight vector of the winner unit.

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 recurrent neural network architecture evolved after 200 generations. Figure 4 shows the change of winners at the SOM layer as the recurrent neural network updates its activation. The tendency of convergence can be observed; after a period of oscillations for 15 iterations, the winner finally converged to unit 17. Figure 5 plots the best and average fitness at each generation during evolution. Figure 6 demonstrates the behav-

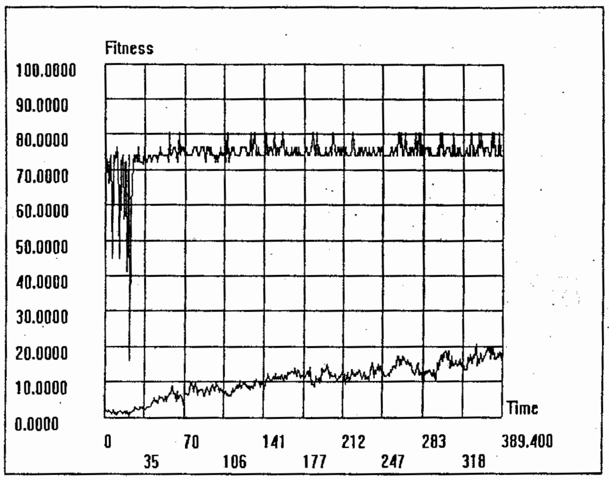

Figure 5: Change of the best and average fitness values.

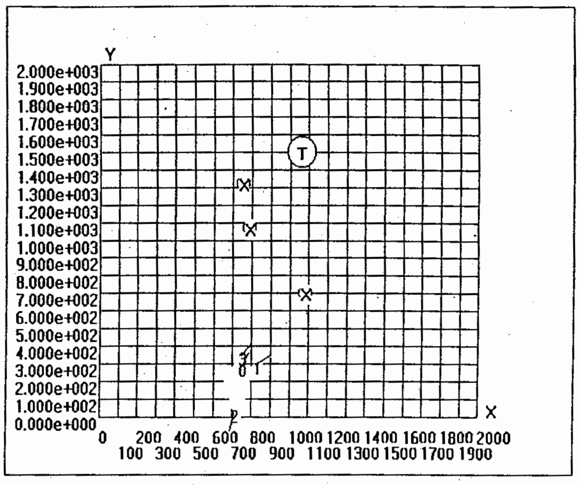

Figure 6: The robots moving in a group toward the table.

ior 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.

5 Concluding Remarks

We have presented a multinet neural architecture for evolving collective behavior of a group of small mobile robots. The network consists of a chaotic neural network for processing spatio-temporal patterns of input stimuli, a self-organizing map for detecting convergent patterns, and a classification layer for producing motor outputs corresponding to the stimuli pattern. The networks were evolved by situating the robots in the simulation world and reinforcing positive actions and penalizing negative actions. The performance was demonstrated in the table-transport problem.

Future work should refine and improve the current framework. The chaotic dynamics of the recurrent network layer is worthy of further analysis. 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 strategies that assign credits to the component networks to evolve more goal-directed collective behavior.

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

[Adachi and Aihara, 1997] M. Adachi and K. Aihara, "Associative dynamics in a chaotic neural network," *Neural Networks* **10**(1):83-98, 1997.

[Haynes *et al.*, 1995] 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.

[Hong and Zhang, 1997] Y.J. Hong and B.-T. Zhang, "Evolving cooperation strategies for multiple mobile robots, *Proc. ASCC-97*, Seoul, July 1997.

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

[Luke and Spector, 1996] S. Luke and L. Spector, "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.

[Mataric, 1993] M.J. Mataric, "Designing emergent behaviors: From local interactions to collective intelligence," *Proc. Second Int. Conf. on Simulation of Adaptive Behavior*, pp. 432-442, 1993.

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

[Yao and Freeman, 1990] Y. Yao and W.J. Freeman, "Model of biological recognition with spatially chaotic dynamics," *Neural Networks* **3**:153-170, 1990.