A Genetic Algorithm-Based Controller for Decentralized Multi-Agent Robotic Systems

Arvin Agah

Bio-Robotics Division
Mechanical Engineering Laboratory, AIST-MITI
1-2 Namiki, Tsukuba 305, JAPAN
agah@melcy.mel.go.jp

George A. Bekey

Computer Science Department
University of Southern California
Los Angeles, CA 90089, U.S.A.
bekey@robotics.usc.edu

***Abstract* -- In this paper the results of evolution on the task performance of a robot colony are discussed. The cognitive architecture of individual robots of a colony are modified, using genetic algorithms, producing a generation of robots with superior task performance, compared with those of the initial robot population. The effects of mutation probability and fitness scaling parameters on simulated evolution are also studied in this paper.**

I. INTRODUCTION

In this paper the effect of evolution on task performance of a robot colony is studied. The cognitive architecture of a group of robots that perform specific tasks in an environment is evolved by using genetic algorithms, hence producing a better generation of robots in each successive generation. The results reported in this paper are based on a colony of simulated robots in a virtual world.

The cognitive architecture of a robot must enable it to sense and act upon the world. The novel architecture used in this paper, termed *Tropism System Cognitive Architecture*, is based on the tropisms of the robot, i.e., its likes and dislikes. Such an architecture transforms the robot's sensing of the world to appropriate actions, thus enabling it to survive and function in an uncertain world. The architecture uses the concepts of positive and negative tropism. An agent's likes and dislikes will form its perceptions and, therefore will result in its actions in the tropism architecture (Agah, 1994; Agah and Bekey 1995). The concept of positive and negative tropisms as principal mechanisms of intelligent creatures was first discussed in (Walter, 1953).

A few examples of related research include: evolutionary robotics (Cliff *et al.*, 1993), ant-like robots (Deneubourg *et al.*, 1991), cellular robotic system (Kawauchi *et al.*, 1992), evolutionary multi-agent robot system (Shibata and Fukuda, 1993), cellular robotic system (Ueyama *et al.*, 1992), and simulated evolution (Wilson, 1988).

II. TROPISM SYSTEM CONTROLLER

After the robot senses the world, the tropisms that match the current perception of the world are selected. A mechanism is then used to select one tropism among the matched tropism, as described later in this section. The selected tropism is then used to produce an action by the robot. The tropism system of the robot can be automatically updated through two processes, namely, learning and evolution. Learning allows individual robots to automatically update their tropism system and can be classified as ontogenetic learning. Evolution enables the robot to learn through simulated Darwinian evolution and can be classified as phylogenetic learning.

Entities in the world include robots, predators, obstacles, a home base and various objects to be used. Each robot is capable of sensing the type and state (e.g., *active* or *inactive*) of other entities. In each tropism, an entity and the state of that entity are associated with an action by the robot, and the associated tropism value. The larger the magnitude of the tropism value, the more likely it is for the robot to perform the action. Once a robot performs a sensory sweep of its surroundings (available sensory area), the set of the tropism elements are checked for any matching entity and entity state. For all the matched cases, the selected action and the corresponding tropism value are marked. The selection of one action from the chosen set is done by the use of a biased roulette wheel. Each potential action is allocated a section on the wheel, proportional to the associated tropism value. Consequently, a random selection is made on the roulette wheel, determining the robot's action, i.e., the selection based on the wheel results in the action that is to be performed by the robot.

III. GENETIC ALGORITHM-BASED CONTROLLER

The automatic updating of the tropism system architecture can be achieved through the evolution of the

Proceedings of The 1996 IEEE International Conference on Evolutionary Computation (ICEC'96), Nagoya, Japan,

system. Using genetic algorithms (Goldberg, 1989), a colony of robots can be evolved for a number of generations, improving the performance of the colony. The techniques are inspired by the processes of natural selection. Techniques of fitness determination, selection, cross-over, reproduction, and mutation are applied to the robot and the robot's chromosomal representation.

At the end of each generation, the parent robots are selected based on a fitness computation. The higher the fitness of the robot, the more likely it is for the robot to be selected. After the two parent robots are selected, each is represented by a chromosomal string. The two strings are combined, using a cross-over point. The two new chromosomes are subjected to potential mutation, and are then converted back to their equivalent robot representation. The selected parents are then replaced in the colony by the two robot offsprings. It is expected that such mechanism of natural selection will eventually result in a population with a higher performance. The first generation (initial population) can be either set randomly, or set to predetermined values.

III.A. Robot Selection

The selection of the robots for reproduction is based on their performance. The selection process is based on a biased roulette wheel selection. The portion of the wheel assigned to a robot is proportional to the robot's performance. The selection of the roulette wheel gives preference to larger portions, as they are more likely to be chosen. The fitness of a robot is determined based on the number of tasks it performs and its energy consumption. Enumerating the types of task performed, and denoting the counts of the number of each type of tasks performed by the robot with p_i, and denoting its energy consumption with e, the overall performance can be measured. The multipliers λ_i will be used to assign different weights (strengths) to the different types of tasks. The energy multiplier of λ_e will also be used to assign a weight to the energy consumption of the robots. The computation of the total performance can be done using two different methods. The first method will be based on the energy per unit task performed, using the multipliers of the task count and the energy consumptions. A larger multiplier implies a higher weight being placed on a certain type of task performance or the energy consumption:

$$\Phi_1 = \frac{(\lambda_e)e}{\sum_i (\lambda_i)p_i} \tag{1}$$

In this method of fitness computation, the lower numbers signify robots that are more fit. Using the second method of computation, the overall fitness of a robot is measured by adding the inverse of energy consumption to the performance:

$$\Phi_2 = \frac{1}{(\lambda_e)e} + \sum_i (\lambda_i)p_i \tag{2}$$

The larger numbers in this method signify robots that are more fit. As the energy consumption increases, the fitness decreases.

The fitness function used in the genetic algorithms is scaled during the determination of the most fit parent robots. The fitness scaling is needed in order to prevent a few very fit individual robots from initially taking over a significant proportion of the population, preventing formations and explorations of new characteristics. The other problem is associated with the later stages of the evolution where the population average fitness and the population best fitness are too close to one another. In both such cases the quality of the population does not improve at its best. Linear scaling of the fitness values can help alleviate these problems:

$$\Phi' = a\Phi + b \tag{3}$$

The coefficients of scaling a and b are computed at each generation, using the minimum, maximum, and average fitness values before the scaling.

III.B. Robot Representation

The tropism system architecture can be converted to its equivalent in the form of a string representing the robot's chromosomes. The chromosome string consists of symbols of 0 and 1. Representation of the tropism elements is done in four steps: (1) The set of all entities, states, and actions are enumerated, and the number representing them is used in the tropism element. (2) The numbers are converted into the binary format (base two), and then converted to the equivalent string format. (3) The strings representing the parts of the tropism element (entity, state, action, and tropism value) are appended to produce one string for each tropism element. (4) The strings representing tropism elements are appended to form the complete chromosome of the tropism system. For example, the entity that is enumerated as entity 11 is converted into the binary number 1011, which is then converted into the string "1011", and appended to the remaining strings.

The size of a section of the chromosome, representing one tropism element, can be computed using the set of all entities E, the set of all states S, and the set of all actions A. The cardinality of a set (number of members) is denoted by $|\ldots|$. The cardinalities of the entity, state, and action sets are assumed to be powers of two:

$$\log_2(|E|) + \log_2(|S|) + \log_2(|A|) + \log_2(\tau_{\max}) \tag{4}$$

Since the tropism elements can be enumerated with a predefined sequence of entities and actions, the chromosome could be simplified to include only the action and the tropism value:

$$\log_2(|A|) + \log_2(\tau_{max}) \quad (5)$$

The total length of the robot chromosome, representing the tropism cognitive architecture system is then computed:

$$|E||S|(\log_2(|A|) + \log_2(\tau_{max})) \quad (6)$$

The conversion of a string to the equivalent tropism system is the reverse of the above process. The chromosome string is first decomposed into sections of predetermined size, representing the tropism elements and the sections of the elements. The decomposed strings are then converted into binary numbers, which are in turn converted into decimal numbers. The resulting numbers represent the enumeration of the set members for entities, states, and actions. The tropism elements are then recovered from the chromosomal string, resulting in the cognitive architecture of the robots in the colony.

An example is presented here to further describe the process of generation of a chromosomal string from a robot's tropism system. A portion of a robot's tropism system is included in Table 1. As described earlier, the sets of entities, states and actions can be enumerated, therefore eliminating the need for including those values in the chromosome. The chromosomal string is then composed only of the concatenation of the tropism values in a predetermined order.

Entity Type	Entity State	Action	Tropism Value
10	0	4	33
11	1	6	195
3	2	7	476

Table 1: Sample tropism elements.

The tropism values are concatenated:

$$(33, 195, 476) \quad (7)$$

The values are converted into binary substrings:

$$(0000100001, 0011000011, 0111011100) \quad (8)$$

And finally the substrings are combined into one string.

$$000010000100110000110111011100 \quad (9)$$

III.C. Robot Combination

The two chromosomes representing the selected robots (parents) must be combined to determine the chromosomes of the two resulting robots (offsprings). The combination of the chromosomes is accomplished by first selecting a cross-over point. This is a location within the chromosomal string that is used as a dividing point for the combination process. The second sections of the two chromosomes are exchanged to form the resulting chromosomes.

The cross-over operation takes place, given a certain probability. If the cross-over does not happen, the original chromosomes are copied into the next generation. The next step in the production of new chromosomes for the offspring robots is mutation. Mutation takes place using a certain probability, where a randomly selected bit in the string will be changed from a 0 to a 1, or from a 1 to a 0. Mutation allows the colony to test new robots, not resulting from the cross-over operation. The probability of mutation (usually less than 0.1) is generally much lower than that of cross-over (usually greater than 0.5). The values of mutation probability and cross-over probability used in the experiments were determined based on earlier experiments. The cross-over and the mutation could result in portions of the chromosomal string that are considered invalid, once converted to the robot cognitive architecture. Such resulting invalid tropism elements in the tropism system will be unusable and remain as such until the next generation of the colony.

IV. ROBOT EVOLUTION EXPERIMENTS

This section addresses a number of issues: in which ways does simulated evolution affect the performance of the robot colony? And what are the effects of mutation probability on colony performance? And what are the effects of the fitness scaling parameter on colony performance? Different types of experiments were performed while genetic algorithms were utilized to evolve a colony of robots. The first two classes of experiments involved the evolution of colonies for performing gathering and attacking tasks. In these experiments a colony with the population of 80 was evolved for 60 generations. The other classes of experiments concerned the effects of the parameters of mutation probability, and the fitness scaling parameter on colony performance. The world resources (objects and predators) were replenished at the end of each generation. The gathering experiments included 1000 small objects, and the attacking experiments included 100 predators. The initial tropism system parameters were set randomly, and the experiment parameters were set according to Table 2, based on the values obtained in the previous experiments.

IV.A. Performance Analysis

The plots for the experiments in gathering and attacking are included in Figures 1 and 2, respectively. In the gathering experiments, the gathering count increased at each generation and eventually leveled off, as all the resources were consumed. After approximately 40 generations, the maximum performance was reached via evolution. The energy consumption of the colony generations decreased as the consequence of evolution, also leveling off at about 40 generations. The same held for the energy per gather. Evolution did create a better colony, reaching its best performance (most efficient) after about 40 generations. The attacking experiments produced similar results, though the

performance reached its peak at about 20 generations, where all predators were attacked. It is evident that the genetic algorithm produced superior colonies of robots. The number of generations required, before the maximum efficiency and performance was reached, was dependent on the availability of the resources for the given tasks. The attacking experiments required fewer generations to reach the maximum performance than the gathering experiments, because of the higher availability of the small objects, compared to the predators.

Figure 1: Performance (gathering) of generations.

Parameter	Value
Cross-over Probability	0.7
Mutation Probability	0.01
Fitness Scaling Parameter	1.5
Fitness Function Multiplier	100

Table 2: Parameters for the experiments.

Figure 2: Performance (attacking) of generations.

The effects of the mutation probability on the colony performance were studied in a set of experiments. The mutation probability, was varied from 0.00 to 0.10 in increments of 0.01. The task performance count (gathered objects), the energy consumption, and the energy per unit gathered are plotted in Figure 3. The experiments were conducted for 40 generations. The colony's performance decreased with larger values of the mutation probability. High probability of mutation does not allow for stable performance improvement of the generations. Low values such as 0.01 seemed to be good choices for this parameter. Figure 4 includes the plots for the experiments varying the fitness scaling parameter from 1.1 to 2.0 in increments of 0.1. The performance count increases as the parameter increases, and the energy consumption decreases. The performance measures are the best for the value of 2.0.

V. CONCLUSION

In order to illustrate the changes to the colony due to the evolution, one robot was randomly selected from generation 1 and one robot was selected from generation 30. The colony population was set to 80 in the experiments. The parameter settings were cross-over probability of 0.4, mutation probability of 0.01, fitness scaling multiplier of 1.5, and fitness function multiplier of 100. The tropism system of the robot at generation 1 (randomly set) and at generation 30 (after evolution) are shown in Table 3. As shown, since the experiments included only gathering tasks, the evolved colony members had a higher tropism value for gathering, than the initial population members had. The tropism value for moving in a new direction was decreased, as a robot consumes less energy when continuing in the original direction, instead of changing and moving in a new direction. The tropism values for the cases that the populations did not encounter decreased, such as helping a robot in the need of help, i.e., the tropism elements dealing with non-occurring events. The chromosome representations of these two tropism systems are contained in Table 4.

It was shown that simulation evolution through the use of genetic algorithms could indeed be used to evolve a colony of robots with superior task performance, compared with the initial colony.

Figure 3: Mutation probability and performance.

Figure 4: Fitness scaling and performance.

Entity Type	Entity State	Robot Action	Tropism Value (Gen. 1)	Tropism Value (Gen. 30)
None	None	No Action	0	0
None	None	Move In Original Direction	643	826
Space	Inactive	Move In New Direction	629	23
Small Object	Inactive	Gather	634	913
None	None	Place	752	107
Large Object	Inactive	Decompose	270	769
Dual Object	Inactive	Call For Help	35	734
Predator	Active	Attack	284	608
Robot	In Need Of Help	Help	452	68

Table 3: Tropism system of a robot.

Pre-Evolution	Post-Evolution
0000000000	0000000000
1010000011	1100111010
1001110101	0000010111
1001111010	1110010001
1011110000	0001101011
0100001110	1100000001
0000100011	1011011110
0100011100	1001100000
0111000100	0001000100
0000000000	0100000010
0000000000	0001000010
0000000000	0000000001
0000000000	0000111000
0000000000	0100100000
0000000000	0000000000
0000000000	0100100000

Table 4: Robot representations.

REFERENCES

[1] Agah, A. *Sociorobotics: Learning for Coordination in Robot Colonies*. Ph.D. Dissertation, Computer Science Department, University of Southern California, Los Angeles, California, (1994).

[2] Agah, A. and Bekey, G. A. Autonomous Mobile Robot Teams. In *Proceedings of the AIAA/NASA Conference on Intelligent Robots in Field, Factory, Service, and Space*, Houston, Texas, Vol. 1: 246-251, (1994).

[3] Cliff, D., Harvey, I., and Husbands, P. Explorations in evolutionary robotics. *Adaptive Behavior*, 2: 73-110, (1993).

[4] Deneubourg, J. L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., and Chretien, L. The dynamics of collective sorting robot-like ants and ant-like robots. In Meyer, J.-A. and Wilson, S. W. (Eds.) *From Animals to Animats*. MIT Press, Cambridge, Massachusetts, 356-363, (1991).

[5] Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, (1989).

[6] Kawauchi, Y., Inaba, M., and Fukuda, T. A strategy of self-organization for cellular robotic system (CEBOT). In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1558-1565, (1992).

[7] Shibata, T. and Fukuda, T. Coordinative behavior in evolutionary multi-agent robot system. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 448-453, (1993).

[8] Ueyama, T., Fukuda, T., and Arai, F. Structure configuration using genetic algorithm for cellular robotic system. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1542-1549, (1992).

[9] Walter, W. G. *The Living Brain*. W. W. Norton & Company, Inc., New York, (1953).

[10] Wilson, S. W. The genetic algorithm and simulated evolution. In Langton, C. (Ed.) *Artificial Life, SFI Studies in the Sciences of Complexity*. Addison-Wesley Publishing Company, Redwood City, California, 157-165, (1988).