Evolving Control for Distributed Micro Air Vehicles[1]

Annie S. Wu
Naval Research Laboratory
Code 5514
Washington, DC 20375
aswu@aic.nrl.navy.mil

Alan C. Schultz
Naval Research Laboratory
Code 5514
Washington, DC 20375
schultz@aic.nrl.navy.mil

Arvin Agah
Department of EECS
The University of Kansas
Lawrence, KS 66045
agah@eecs.ukans.edu

1 Introduction

The general idea of distributed robotics (or multi-robot systems) is that teams of robots, deployed to achieve a common goal, can outperform individual robots in terms of efficiency and quality and, in some cases, can perform tasks that a single robot cannot. Consider, for example, Micro Air Vehicles (MAVs), each of which has an extremely small payload capacity. Though individual MAVs may have limited capabilities, teams of MAVs, possibly carrying different payloads, can be deployed as a group to perform complex tasks. Groups of robots provide an added level of robustness, fault tolerance, and flexibility over individuals, as the failure of one robot does not result in the failure of the mission, as long as the remaining robots can redistribute and share the tasks of the failed robot. Examples of tasks appropriate for robot teams are large area surveillance, environmental monitoring, large object transportation, planetary exploration, and hazardous waste cleanup.

In this paper, we focus on the task of large area surveillance. Given an area to be surveilled and a team of MAVs with appropriate sensors, the task is to dynamically distribute the MAVs appropriately in the surveillance area for maximum coverage based on features present on the ground, and to adjust this distribution over time as changes in the team or on the ground occur. We have developed a system that will learn rule sets for controlling the individual MAVs in a distributed surveillance team. Since each rule set governs an individual MAV, control of the overall behavior of the entire team is distributed; there is no single entity controlling the actions of the entire team. Currently, all members of the MAV team utilize the same rule set; specialization of individual MAVs through the evolution of unique rule sets is a logical extension to this work.

A Genetic Algorithm (GA) is used to learn the MAV rule sets. A GA is a search method based on principles from natural selection and genetic reproduction. GAs have been successfully applied to a wide range of problems, including optimization, classification, and design. The typical GA evolves the composition of fixed length individuals, each of which represents a potential solution to the problem to be solved. There has been increasing interest in the evolution of variable length individuals in which both the size and the composition of a solution are dynamically evolved by a GA. The increased flexibility and evolvability of variable length systems appears to be beneficial to the GA's search process. In particular, studies have found interesting links between parsimony pressure (rewarding for compactness and small size), mutation rate, and evolved genome length and fitness. Size issues are important in the evolution of rule sets. Smaller rule sets require less time to evaluate; larger rule sets are capable of containing more specific rules. In this paper, we examine some of the issues regarding variable length GAs as we investigate the evolution of variable sized rule sets for controlling MAVs.

[1] This work was supported by the Office of Naval Research, the Naval Research Laboratory, and the National Research Council.

2 Related Work

2.1 Multiple Robot Systems

A number of researchers have built multi-robot systems in order to investigate the cooperation and pooled capabilities of distributed robots, focusing on team organization, interaction and task performance. Using different levels of control strategies, Mataric [16] has used groups of up to twenty mobile robots to study group behavior. Each robot used a measure of local population density and population gradient to balance its behavior between collision and isolation. Kube and Zhang [14] have shown how a team of five robots without explicit communication can cooperate in a collective box pushing task. Arkin [2] has shown that the behavior of robots in a team can be composed of a collection of motor schemas, and the robots will alternate among a number of states (forage, acquire, etc.) in order to find and deliver certain objects. Agah and

Bekey [1] investigated the development of a specific theory of interactions and learning among multiple robots performing certain tasks. This work showed the feasibility of a robot colony in achieving global objectives, when each robot is provided with only local goals and information. Goss and Deneubourg [9] studied chain-making behavior in robots, where robots spread themselves out in the environment while remaining in contact with each other. The fact that robots could function as beacons effectively enlarges the area of coverage.

Utilization of GAs in evolving robot controllers has been investigated in a number of research efforts [5, 6, 10, 12, 23, 20, 21, 1]. The work in this paper extends previous work in several ways. First, this work is oriented towards control of distributed unmanned air vehicles, not land-based vehicles. In addition, this study will focus on the dynamics of evolving variable sized rule sets, including (1) the effects of both initial and evolved rule set sizes on the performance of a robot colony and (2) the role of parsimony pressure in evolutionary robotics.

2.2 Variable Length GA Representations

Within evolutionary computation, variable length representations are most prominent in genetic programming (GP), a variant of the GA which directly evolves programs that vary in both size and content. Interestingly, with no bound on size, GP tends to evolve programs that are much larger than necessary, containing sections of code that are never used [3, 13]. Though many early efforts focused on "editing out" these excess regions, later studies indicate that the size of evolved programs may affect their fitness and evolvability [15, 17, 19, 22].

Although most GA applications do not use variable length genomes, there are several examples in which variable length genomes have been used successfully. The messyGA [8] employs individuals whose length and content vary dynamically. The flexibility provided by this representation appears to be advantageous in deceptive problems. The SAMUEL learning system evolves variable length rule sets and has been used to develop collision avoidance, tracking, and other behaviors for mobile robots. It uses detailed, high-level rules and heuristic techniques for modifying rule sets. The Virtual Virus (VIV) project [4] investigated the link between mutation rate and evolved length in a variable length GA system [18]. Studies found that parsimony pressure, is essential to both keeping individuals manageable in size and in maintaining reasonable fitness. More interestingly, there appears to be a direct connection between the evolved length and fitness of individuals and the mutation rate. The work described here will be compared to some of the conclusions from the VIV project.

3 Experimental Details

The goal of this work is to develop a system that is able to learn rule sets for controlling the behavior of a team of MAVs that are continuously surveilling a specified area. The learning mechanism is a genetic algorithm which evolves the rule sets that govern MAV behavior. The fitness of the evolved rule sets is determined by the performance of a team of MAVs in a simulated world. In this section, we describe in detail the problem to which we apply our MAV team, a simulator that is used to evaluate MAV performance, and the evolutionary learning system.

3.1 Large Area Surveillance

In this paper, we focus on the task of large area surveillance. Multi-robot teams are ideal for such a task for several reasons: (1) a team of robots can continuously surveil the entire area in parallel, (2) different types of robots may be sent to surveil different types of areas, (3) teams of multiple, distributed robots may be more robust and fault-tolerant, as the loss of a single robot does not necessarily result in failure of the mission, and (4) teams of small, inexpensive robots may be less expensive and less detectable than a single larger vehicle. Given a specified geographical area and a team of autonomous MAVs, the MAVs must dynamically position themselves to provide maximum coverage of the ground. Different features on the ground may generate different levels of interest, requiring more or less MAVs to adequately surveil. For example, military bases, airports, ports, and other strategic areas can be considered areas of high interest which would require increased surveillance, i.e., more MAVs. Rural areas and open water may be considered areas of low interest, requiring fewer vehicles to cover. In addition, the total number of MAVs involved in the task may change over time as individual MAVs exhaust battery power, as MAVs are destroyed by outside forces, or as new MAVs are deployed. Interest levels on the ground may also change with time. The full team of MAVs should be able to dynamically adapt their behavior to the size of the team and to changes on the ground.

Our goal is to develop a system that will learn rule sets for controlling individual MAV behavior that allows a team of multiple MAVs to successfully perform large area surveillance. We are particularly interested in the factors that affect the size of the final evolved solution, i.e. the number of rules in a rule set. Solution size is important for several reasons. Larger solutions may contain more detailed rules, but require more processing time. On an autonomous robot where resources are limited, CPU processing time is valuable. Smaller solutions are quicker to process and are more likely to contain generalized rules, but may not be able to handle all necessary situations. Previous work indicates that the size of evolved solutions may be affected by GA parameters such as parsimony pressure and mutation rate [18].

3.2 Problem Representation and Simulation

We use a simulator to evaluate the performance of MAV teams in this study. The simulator is initialized with a number of parameters that specify the environment, the MAV team, and other variables of the experiment. The simulation defines the world within which the MAVs move about, sense each other and the ground, while performing their task of surveilling the region. In these experiments, MAVs are assumed to have enough energy capacity to function throughout the entire experiment although they are destroyed if they collide with one another or go beyond the boundaries of the surveillance area. All areas of the ground currently have the same interest level; therefore, good solutions should distribute the MAVs equally over the entire surveillance area. Figure 1 shows a snapshot of a sample run from the MAV simulator. Initially, all MAVs in a simulation are lined up along the west border of the surveillance area.

In the current simulation, all MAVs are identical in configuration and all MAVs are governed by the same rule set. Each MAV has eight sensors, distributed around its perimeter, which indicate if anything (e.g. another MAV or a border) is within a certain range in that direction. This *sensor range* is specified in the initial parameters of the simulator and is represented in Figure 1 as a large black circle around each MAV. Sensors cannot detect the number or distance of objects within their range. As a result, sensor data is binary: "1" indicates that an object has been detected, "0" indicates that no object is detected. A *survey range* is also initialized for each MAV; this value determines the area of ground that a MAV can detect beneath itself. In Figure 1, the *survey range* is the white area surrounding each MAV.

Each MAV is controlled by its rule set. The rule set specifies which action should be taken at any time step given the sensor data. As shown in Figure 2, each rule consists of a condition and an action clause.

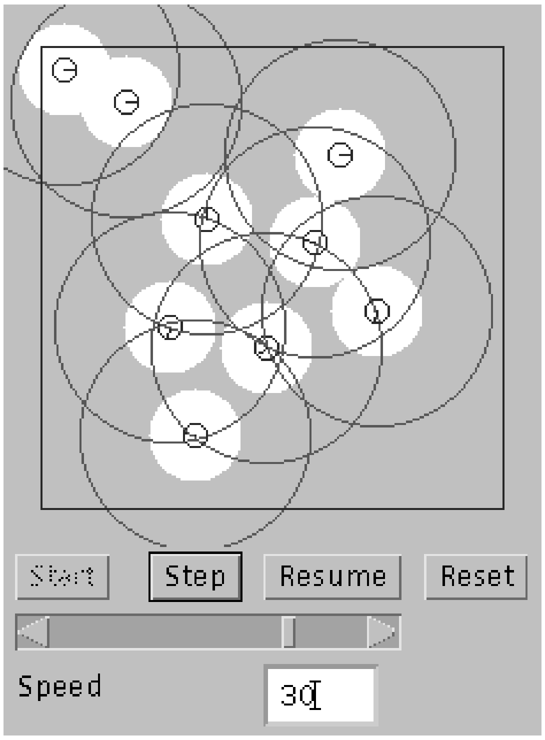

Figure 1: Sample run from the MAV simulator.

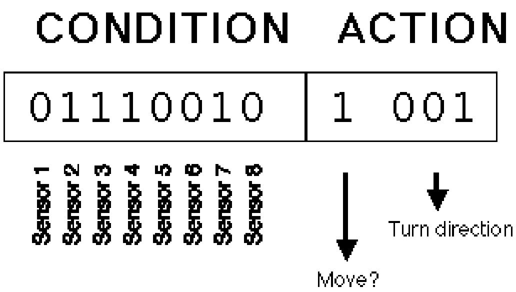

Figure 2: A MAV rule.

Sensor data is compared to the condition clause of each rule in the rule set. The rule with the best match is selected. If multiple rules qualify as the best match, one is chosen at random from the candidates. If the degree of match of the selected rule exceeds a given threshold, the action clause of that rule is executed. The action clause of a rule consists of two fields. The first field indications whether the MAV should move forward or not. The second field indicates the direction in which the MAV should turn. A turn can be in one of the eight compass directions. Although geometrically diagonal moves (NE, NW, SW, SE) result in larger

```
procedure GA
begin
    initialize population;
    while termination condition not satisfied do
    begin
        select parents from population;
        create copies of selected parents;
        apply genetic operators to offspring;
        perform evaluations of offspring;
        insert offspring into population;
    end
end.
```

Figure 3: A genetic algorithm.

changes in position, they are assumed to take place in one time step, similar to non-diagonal moves (E, N, W, S). Every MAV performs this evaluation of rules and sensor data once in each time step and executes an action if the matching threshold is exceeded.

At every time step, we calculate the percentage of the surveillance area that is covered by the combined *survey ranges* of all of the MAVs. This value is averaged over the entire run and returned as an evaluation of the performance of that run.

3.3 The Genetic Algorithm

Some features that distinguish genetic algorithms from other search methods are: (1) a *population* of *individuals* that can be interpreted as candidate solutions to the problem to be solved, (2) a *fitness function* that evaluates how good an individual is as a solution to the given problem, (3) the *competitive selection* of individuals for reproduction, based on the fitness of each individual, and (4) idealized *genetic operators* that alter the selected individuals in order to create new individuals for further testing. A GA simulates the dynamics of population genetics by maintaining a population of individuals that evolves over time in response to the observed performance of its individuals in their operational environment. The fitness function is used to evaluate each individual in a population. Selection exploits and propagates good solutions while genetic operators allow a GA to further explore the search space for even better solutions. The basic paradigm is shown in Figure 3. For additional details, the reader should see [7, 11].

For the experiments described in this paper, each individual of the GA population represents a complete rule set. Each rule in the rule set consists of a condition clause (eight bits) and an action clause (four bits). Individuals are interpreted into rule sets by taking every twelve bits as a rule starting from the left end. Crossover points may occur at different locations on each parent, but must occur at rule boundaries, never within rules. As a result, all individual lengths are multiples of twelve. Mutation can occur at any location. To evaluate a particular individual of the population, the GA converts the individual to its corresponding rule set, and runs the MAV simulator with this rule set. The performance of the MAV team using this rule set becomes the fitness of that individual.

Population size	100
Generations	200
Maximum genome length	600 bits (50 rules)
Crossover operator	1 point random
Crossover rate	1.0
Initial genome lengths	60, 360
Mutation rate	0.001, 0.005, 0.01
Parsimony pressure	OFF, ON

Table 1: GA parameter settings.

Arena height	200
Arena width	200
Number of MAVs	9
MAV size (radius)	5
Survey range	30
Sensor range	50
Number of sensors	8

Table 2: MAV simulator parameter settings.

4 Experiments and Discussion

Table 1 shows the GA parameter settings and Table 2 shows the MAV simulator parameter settings used in the experiments described in this paper. The experiments described here focus on the effects of three GA parameter setting on two main aspects of the evolved MAV rule sets: fitness and length. Recall that longer individuals represent larger rule sets. As shown in Table 1, we vary the initial genome length (size of individuals in the initial population), the mutation rate, and whether or not parsimony pressure is applied.

Figure 4 shows the best, average, and worst fitness of six example runs. The top row does not use parsimony pressure; the bottom row does. Columns one through three use mutation rates of 0.001, 0.005, and 0.01, respectively. These plots indicate that lower mutation rates appear to produce better individuals (rule sets). In addition, parsimony pressure appears to have little effect on the plateau fitness, where plateau fitness

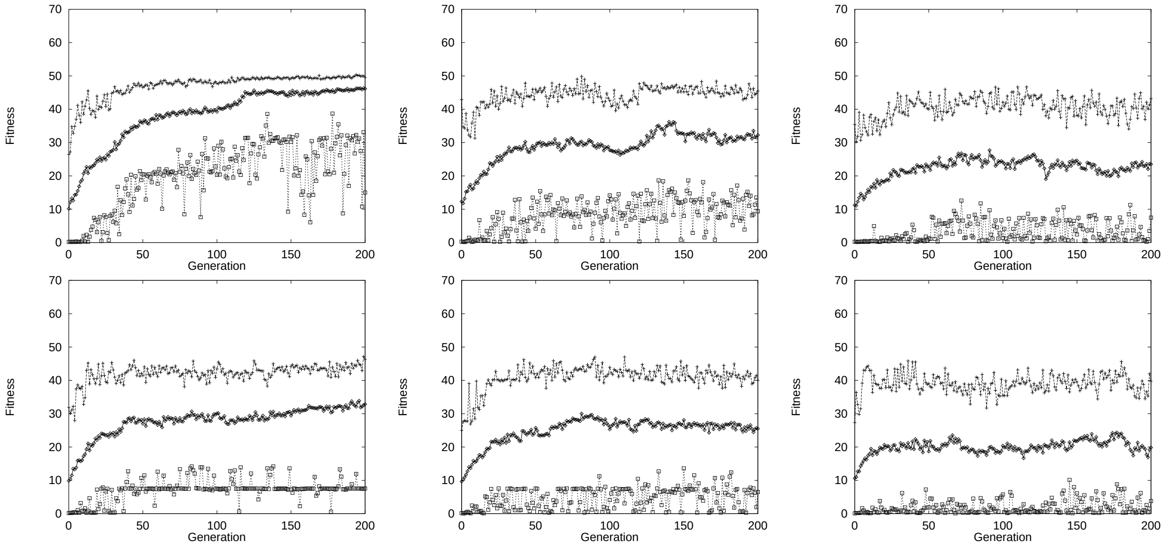

Figure 4: The top line shows the best fitness of each generation; the middle line shows average fitness; and the bottom line shows worst fitness. The left column uses a mutation rate of 0.001; the middle column, 0.005; and the right column, 0.01. The top row does not use parsimony pressure; the bottom row does.

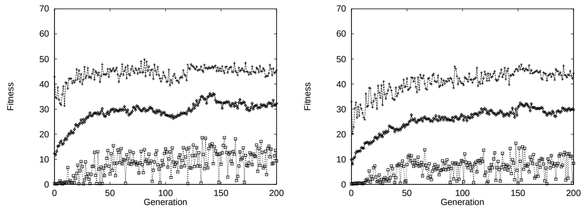

Figure 5: Effects of initial genome length on plateau fitness. The top line shows the best fitness of each generation; the middle line shows average fitness; and the bottom line shows worst fitness. In the left plot, initial genome length is 60; in the right, 360.

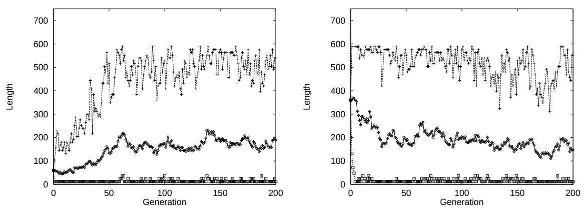

Figure 6: Effects of initial genome length on plateau length. The top line shows the longest individual of each generation; the middle line shows average length; and the bottom line shows shortest length. In the left plot, initial genome length is 60; in the right, 360.

refers to the fitness at which a run levels off. Figure 5 shows that the genome length of the initial population also has little or no affect on the plateau fitness. Regarding the evolved length of the individuals or rule sets, parsimony pressure appears to be an important controlling factor. When there is no parsimony pressure, the GA evolves individuals that are as large as the maximum allowed size (600 bits in these experiments). When there is parsimony pressure, the GA evolves more compact individuals that, as indicated in Figure 4, have just as good fitness as when parsimony pressure is off. Figure 6 illustrates the fact that the evolved plateau length of the individuals is also independent of the length of the individuals in the initial population.

Given the parameters settings from Table 2 The maximum percentage of the surveillance area that can be covered by the MAV team is 63.6%. The results here show that our system evolves rule sets that allow the team to continuously surveil approximately 40% of the area. Because the MAV team is typically continuously moving, the actual percentage of the area monitored over a stretch of time may be larger than 40%. These results were achieved with no fine tuning of the GA to our specific problem.

The fact that lower mutation rates result in better fitness and initial genome length does not affect the final evolved plateau fitness and length agrees with previous studies on variable length GA systems [18]. Unlike Ramsey, et al. [18], where parsimony pressure produces higher fitness, parsimony pressure appears to have little effect on the plateau fitness of the MAV rule sets. We speculate that the reason for this difference may be due in part to differences in problem representations as well as differences in the difficulty of the problems. Further studies are planned to investigate the impact of these differences.

5 Future Work

The future work on this project can continue in a number of directions. One approach is to introduce the quality of task performance in the computation of the fitness function (in addition to the quantity). The quality metric would be a measure of how well the MAVs survey the given region over time. Another extension is the addition of varying interests levels to the ground that is being surveilled. Regions with high interest levels are more important and may require more MAVs, or more frequent surveys. Another planned direction is to transition from simulation to real robots, testing the evolved rule sets on a team of physical robots, either with mobile robots on the ground or actual flying robots.

Bibliography

[1] A. Agah and G.A. Bekey. Phylogenetic and ontogenetic learning in a colony of interacting robots. *Autonomous Robots Journal*, 4:85–100, 1997.

[2] R.C. Arkin. Cooperation without communication: multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9:351–364, 1992.

[3] T. Blickle and L. Thiele. Genetic programming and redundancy. In *Genetic Algorithms Within the Framework of Evolutionary Computation (Workshop at KI-94)*, pages 33–38, 1994.

[4] D.S. Burke, K.A. De Jong, J.J. Grefenstette, C.L. Ramsey, and A.S. Wu. Putting more genetics into genetic algorithms. *Evolutionary Computation*, 6(4):387–410, 1998.

[5] D. Cliff, I. Harvey, and P. Husbands. Explorations in evolutionary robotics. *Adaptive Behavior*, 2:73–110, 1993.

[6] J.L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien. The dynamics of collective sorting robot-like ants and ant-like robots. In *From Animals to Animats*, pages 356–363, 1991.

[7] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[8] D.E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3:493–530, 1989.

[9] S. Goss and J.L. Deneubourg. Harvesting by a group of robots. In *Toward a Practice of Autonomous Systems*, pages 195–204, 1990.

[10] J.J. Grefenstette, C. L. Ramsey, and A. C. Schultz. Learning sequential decision rules using simulation models and competition. *Machine Learning*, 5(4):355–381, 1990.

[11] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

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

[13] J. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.

[14] C.R. Kube and H. Zhang. Collective robotics: From social insects to robots. *Adaptive Behavior*, 2:189–218, 1994.

[15] W.B. Langdon and R. Poli. Fitness causes bloat. In *2nd On-line World Conference on Soft Computing in Engineering Design and Manufacturing*, 1997.

[16] M.J. Mataric. Minimizing complexity in controlling a mobile robot population. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 830–835, 1992.

[17] P. Nordin and W. Banzhaf. Complexity compression and evolution. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 310–317, 1995.

[18] C.L. Ramsey, K.A. De Jong, J.J. Grefenstette, A.S. Wu, and D.S. Burke. Genome length as an evolutionary self-adaptation. In *Parallel Problem Solving from Nature 5*, pages 345–353, 1998.

[19] J. Rosca. Generality versus size in genetic programming. In *Genetic Programming 1996*, 1996.

[20] A.C. Schultz. Learning robot behaviors using genetic algorithms. In *Proceedings of the First World Automation Congress*, pages 607–612, 1994.

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

[22] T. Soule, J.A. Foster, and J. Dickinson. Code growth in genetic programming. In *Genetic Programming 1996*, pages 215–233, 1996.

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