Control of Distributed Micro Air Vehicles for Varying Topologies and Teams Sizes

Daniel James Collins and Arvin Agah

Abstract: This paper focuses on the study of simulation and evolution of Micro Air Vehicles. Micro Air Vehicles or MAVs are small flying robots that are used for surveillance, search and rescue, and other missions. The simulated robots are designed based on realistic characteristics and the brains (controllers) of the robots are generated using genetic algorithms, i.e., simulated evolution. The objective for the experiments is to investigate the effects of robot team size and topology (simulation environment) on the evolution of simulated robots. The testing of team sizes deals with finding an ideal number of robots to be deployed for a given mission. The goal of the topology experiments is to see if there is an ideal topology (environment) to evolve the robots in order to increase their utility in most environments. We compare the results of the various experiments by evaluating the fitness values of the robots i.e., performance measure. In addition, evolved robot teams are tested in different situation in order to determine if the results can be generalized, and statistical analysis is performed to evaluate the evolved results.

Keywords: robot control, flying robots, micro air vehicles, evolutionary robotics, distributed robotics, genetic algorithms

I. Introduction

A single ant can perform many amazing tasks such as carrying a food morsel that weighs several times more than the ant's own body weight. It is often more interesting to look at the work of a colony of ants. A team of hundreds of ants can efficiently work together on tasks such as building anthills. This distributed approach has been adopted in many aspects of the human world and the robotics world. In general, a team of distributed robots can perform various tasks more efficiently than a single robot. Specifically, using a team of robots adds fault tolerance to the system. If a single robot fails to function, then the remaining robots can possibly take over the duties of the failed robot, thus ensuring the team still meets the goal. There are many applications for distributed robotics such as surveillance, collection of samples, exploration of space, or evacuation of land mines.

Another area that is being explored in robotics is the idea of evolving control of the robots. In robotics, the robots are given rule sets that guide the robot to perform certain actions. The robot has a list of conditions that correspond with different actions. Instead of explicitly writing the rule sets for the robots, scientists have explored evolving the rule sets using Genetic Algorithms (GAs). GAs are methods of searching, using Darwin's ideas of natural selection. The idea of using GAs to solve problems is not a new idea. In fact, GAs have been used to classify and optimize solutions in many arenas, including robotics.

This paper concentrates on simulation and evolution of a team of distributed flying robots. Specifically, the focus will be on the task of surveillance, using Micro Air Vehicles (MAVs). MAVs are lightweight, autonomous air vehicles that can be equipped with various sensors and payloads for various missions (Wu *et al.*, 1999). In this research, we assume the vehicles can detect certain features of the land below the vehicle, and can detect other MAVs within a certain radius of itself. The MAVs' task, as a group, is to fly over an area and perform reconnaissance. They need to maximize the area covered, concentrating on areas of more importance, and minimizing duplication of effort. This paper focuses on two important issues in simulated design of evolved multi-robot systems, namely, the size of the robot team and the environment for evolution. Therefore, the focus will be on the effects of team size and environment on evolution, and how robots evolved for a certain team size or a certain environment would perform when utilized differently. It should be noted that the team size refers to the number of MAVs deployed in each simulation experiment, and is different from the population size of Genetic Algorithms.

The paper is organized into five sections, including the introduction. Section 2 focuses on background and related work in the areas of distributed robotics, genetic algorithms, and evolutionary robotics. Section 3 describes in detail the MAVs, the control structure, and the simulation program. Section 4 details the experiments, the results, and the analysis. Section 5 concludes this paper, providing the limitations and future work of this research.

Manuscript received: May 1, 2001, Accepted: Nov. 5, 2001.

Daniel James Collins: Celeritas Technologies, Overland Park, Kansas 66210, USA.

Arvin Agah: Department of Electrical Engineering and Computer Science The University of Kansas, Lawrence, Kansas 66045, USA (agah@ku. edu)

II. Background and related work

1. Distributed robotics

Distributed robotics involves the studying of robots teams instead of individual robots. Most research in this area can be divided into two areas of simulated systems (use of simulation programs to study issues of distributed robotics) and physical systems (fabrication and testing of robots in hardware).

1.1 Simulated systems

There are numerous examples of distributed robots in simulation. One example scenario in simulation experiments is the predator versus the prey experiments. One such experiment placed a group of robots in a simulated environment similar to a Skipper Computer Zoo (Peters *et al.*, 1998). The computer zoo in this instance was a complete electronic ecosystem that consisted of computer programs that represented the behavior of different animals, in an environment generated

of different animals, in an environment generated dynamically by the user. The animals competed against each other in the predators versus prey simulation (two types of robots). The robots were allowed to evolve and to control the energy source of any robot that they captured. The main goal of the robots was to survive in the environment, using energy sources that were placed in the environment. There were four variables forming the main characteristics of the prey: maximum speed, maximum sensor range, energy level, and method to avoid predators. The predators tried to capture prey robots while avoiding obstacles in the environment and other attackers. For evolution, the fitness of the robots was calculated using two different fitness functions. The fitness of the prey was a summation of the maximum speed, maximum sensor range, and the number of times that the prey successfully escaped the predator. The fitness of the predators was a summation of the maximum speed and the sensor range. It is interesting to note that the predators were not rewarded in the fitness function for capturing the prey. The experiments performed showed that the prey robots became more adept at hiding as evolution proceeded (Peters *et al.*, 1998).

1.2 Physical systems

More recent studies in robotics have dealt with using teams of distributed robots to accomplish a task, rather than a single robot. Teams of distributed robots provide a more robust solution with higher fault tolerance than a single robot system. Often robots are used in environments that are considered unsafe for humans, such as clearing of minefields. Specifically, the project Basic UXO Gathering System or BUGS tested the feasibility of having a team of distributed robots properly dispose of unexploded mine shells (Debolt *et al.*, 1997). The robots first used reconnaissance to detect the mine shells, and then a robot could either carry a shell away or blow it up. The system focused on the ideas of distributed robots, i.e., cheap, simple design robots that are less expensive than a single, highly intelligent robot. Another area in the study of distributed robotics is robot soccer competitions. Mirosot is a robotics competition held annually (Mirosot, 1999). There are many different competition areas in terms of the number and the size of robots. Soccer is a good arena for testing distributed robotics, as the robots must work together in order to accomplish their objective: to win. Companies and universities develop the robots and the algorithms to compete in the competition. Teams take varying approaches to issues such as robot control, path-planning, and learning (Park *et al.*, 1997) (Kim *et al.*, 1997).

Another application for distributed robotics is in the area of factory automation. Often a task is too burdensome for one robot to carry out. For example, robots often need to move large objects on the factory floor. This task has been illustrated through the cooperation of two manipulators moving a large object on the factory floor (Sugar and Kumar, 1998). The object used in the experiment was large enough to warrant the need for two robots to carry the object. To coordinate the control of the two mobile platforms, a leader for the two robots is chosen, responsible for communicating the plans with the other team members. In the case of this experiment, there was one leader and one additional team member. The leader was responsible for the path-planning and obstacle avoidance. The leader decided which path to take and the trailing robot receives communication from the leader on the desired position and follows the lead robot with minor adjustments in position.

Another interesting experiment with teams of robots deals with having a group of robots work together to overcome challenges such as climbing over objects bigger than the robots or crossing over large gaps in surfaces (Hosokawa *et al.*, 1998). The robots used in the experiment were 90x90x90 cm cubes, with two arms on opposite faces of the cube. The arms of two robots could be connected using a lock and key system. The robots were built so that one robot, robot A, can lock arms with another robot, robot B, and lift robot B over the top of robot A. The control scheme used in the experiments was similar to cellular automata, where a the robot took a course of action based on its own state and the state of the near-by robots. These ideas were tested using four robots. The objective was for the robots to climb a structure that was twice the size of the robots. The first three robots formed a stair-like structure and the fourth robot was allowed to get to the top of the larger object. Apparently, the robots performed the experiment without failures such as toppling over.

1.3 Robot team size

An important classification of studies on distributed robotic systems involves the size of the robot teams. One set consists of studies focusing on very large robot populations, called swarms, pertaining to group sizes measured in the hundreds or more. Studies of swarm systems include (Deneubourg *et al.*, 1990), (Beni and Hackwood, 1990), and (Hackwood and Beni, 1992). The second set of studies focuses on small group populations, usually measured in the tens of robots. One such study focused on multi-robot systems and specifically with cooperation of small robot populations (Kube and Zhang, 1994). The team developed the *SimbotCity* simulator that tested the ability of small groups of robots to communicate and work together to push a box to the desired location. In the simulator, there was no explicit communication between the robots. The robots could determine what other robots were doing by observing their actions. It was shown that as the number of robots involved in the simulation increased, the time to carry out the goal of pushing the box to the desired location decreased. The performance gain from adding more robots dropped when the robot population reached 10 to 14 robots. This was due to the world becoming too crowded with robots.

In another study, experiments were performed on interactions between heterogeneous robot populations (Mataric, 1992). The populations consisted of approximately 20 mobile robots, where each robot was equipped with four wheels, a gripper, infrared sensors, bump sensors, and radio transmitters and receivers. Three control structures were defined with various levels of intelligence. The most basic control structure, *ignorant coexistence*, forced the robots to treat all other robots as mere obstacles in the environment. The second level, *informed coexistence*, allowed the robots to distinguish other robots from obstacles in the environment. In this model, when a robot detects another robot blocking the desired path, then

the detecting robot paused and waited for the other robot to carry out its action and move out of the way. The final control structure, *intelligent coexistence*, allowed the robot to sense the density of the surrounding environment, to balance flocking to and avoiding populated areas.

Many researchers use insect-like behaviors to study different distributed robotics issues. One study used an approximate population of 20 heterogeneous robots to illustrate task assignments in *Polisites* wasp colonies (Theraulaz *et al.*, 1991). They implemented two types of interactions between the robots. The first type, *hierarchical*, used dominance to assign tasks. The second type, *trophic*, used interactions with the environment to assign tasks. Another small population example is the study of chain-making behavior of robots (Goss and Deneubourg, 1990). In the experiments the robots communicated via beacons. One robot acted as a central beacon for the surrounding robots. Once a robot moved out of the range of a beacon, then that robot became a beacon. This allowed the robots to spread out and travel through the environment without becoming lost.

2. Genetic algorithms

Genetic Algorithms (GAs) are based on the principles of genetics and natural selection. GAs search for the optimal solution to a problem by evolving many generations of string structures (DNA) and determining the fittest of the population (Goldberg, 1989). The initial population used in GAs is initialized to have random string structures. The population is tested and the individuals are evaluated for the fitness or the effectiveness of the individual. The most fit individuals form the model for next generation. The new population carries characteristics from their parents and possible mutation of characteristics.

In order to illustrate how GAs work, we will focus on the black box optimization problem, involving a number of switches that can be turned on or off. The object is to turn certain switches on and others off to receive the highest reward or output. Considering a black box with seven switches, it could have 2^7 possible outputs or 128 possibilities. Rather than simply moving the switches through the 128 possibilities, GAs can provide a solution to the problem. Starting with an initial population with size four, and randomly choosing the seven bits of each individual, initializes the population: 1010101, 0101010, 1100110, and 0101100. The input values are supplied to the black box and the output is calculated. The output in this example serves as the fitness value. The higher the fitness value the more fit the individual.

The most common operators to use are reproduction, crossover, and mutation. Reproduction states that the most fit individuals in a generation are represented in the new generation according to the percentage of the individual compared to the total fitness of the group. For instance, if the string 1010101 has a fitness of 25 and the overall fitness for all four members of the generation is 100, then that specific string should have its traits passed to approximately 25% of the next generation. Generally, a "biased roulette wheel" is developed using the individual fitness values of the strings compared with the group or generation fitness. This roulette wheel has spaces allocated for each string in the current generation according to the string's percentage of the total fitness of all the members of the generation. For a string with a fitness value of 25 out of a total generation fitness of 100, the string will occupy 25% of the wheel. The crossover operator is the position in which parent strings should be cut, swapped, and pasted to form new strings. For example, if the crossover is three, then the two parent strings 1010101 and 0101010 should form the two new strings 1011010 and 0100101. Fig. 1 illustrates this example. This allows traits from the parents to join together to form the child. The mutation operator is the random switching of bits in the string representation. The mutation allows new traits to be developed without removing the characteristics of the parents. Studies have shown that mutation rates should be approximately one mutation per thousand bits (Goldberg, 1989).

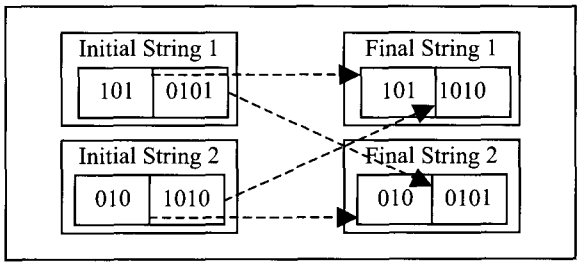


Fig. 1. An example of crossover operation in genetic algorithms.

Additional advanced features of genetic algorithms surround the area of adding penalty functions to the algorithm. A central problem with genetic algorithms is how to handle occasional infeasible offspring that are created from the genetic operators. There are basically four techniques for removing the constraints caused by infeasible offspring which include the rejecting strategy, the repairing strategy, the modifying genetic operators strategy, and the penalizing strategy (Gen and Cheng, 1996). The rejecting strategy simply removes the chromosomes that are unreasonable during the entire evolution process. The repairing strategy "fixes" an unrealistic offspring through a repair function that converts the chromosome into a feasible chromosome. The third technique, modifying genetic operators, puts the burden on the algorithm designer to create a solution that is problem-specific with specialized genetic operators that will ensure that infeasible offspring are not produced. This technique is rewarding; however, it has been shown that more rapid optimization and more robust solutions are often reached by not limiting the search space to only feasible chromosomes. In penalizing strategy technique, the goal of a penalty function is to guide the search to a desired area of the search space.

3. Evolutionary robotics

Evolutionary robotics is the result of applying the concepts of genetic algorithms to the realm of robotics, evolving brains (controllers), and possibly the body of the robots. There are numerous studies on the use of GAs to evolve the control of robots in many different environments. One environment that has been used to test the success of GAs is robot soccer (Agah and Tanie, 1997). They used GAs to evolve the Tropism-Based control structure, based on "likes" and "dislikes" of the

robots in terms of actions that they like to perform. The fitness function that was used rewarded the robots for scoring goals, assisting teammates in scoring goals, and blocking opponent's shots on goal. They showed that after evolving the robots for many generations, the robots produced a better team than the initial populations. GAs have also been used to evolve neural networks for robot control (Lund and Miglino, 1996). They developed a system using a combination of the physical and simulated world. They first developed a physical world for the Khepera robot (Mondada, 1996). Then, they took the robot and allowed it to go through many different motions. All the sensory data was read from the robot into the computer during this phase. The simulator could then take real sensory and motor data from the physical world, and model those in the simulated world, using them as inputs for the neural network. They argue that this method significantly reduced the amount of time for transition of a simulated system to a physical system.

Another common application for evolutionary robotics is the area of biped locomotion. Researchers have developed a Steady State Genetic Algorithm to search for the necessary torques for moving each joint of the robot leg in the appropriate path (Rodrigues *et al.*, 1996). They modeled the robot in software as a connection of five rigid bodies and two revolutionary joints for each of the legs. One revolutionary joint was for the knee and the other was for the hip. Each simulation was allowed only three seconds to carry out the desired action. They studied the complexity of the problem in that it took 2000 iterations for the robot to learn how to stand. They repeated the walking experiments with varying velocities, showing the success and failures of the application of genetic algorithms.

Utilization of GAs in evolving robot controllers has also been investigated in (Cliff *et al.*, 1993) (Deneubourg *et al.*, 1991) (Grefenstette *et al.*, 1990) (Shibata and Fukuda, 1993) (Agah and Bekey, 1996) (Ueyama *et al.*, 1992). The work in this paper extends previous work in a number 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 rule sets for teams of varying size.

III. Micro air vehicles

1. Robots

The micro air vehicles used in these experiments are set to realistic parameters based on prototypes, as described in (Wu *et. al*, 1999) (Collins, 1999). The robots in the experiment are circular and have a radius of five units. The MAV is equipped with eight sensors, giving it a sensing radius of 30 units. The sizes are parameters that can be changed for each experiment. These sensors can be used to detect other robots and the boundary of the survey area; however, the sensors simply return one bit, 1 or 0, depending on if something is present or not. In other words, the MAVs detect other MAVs and the boundary edges, but the MAVs will not be able to distinguish between another MAV and a boundary edge of the environment.

In addition to sensing other robots and the boundary, each robot has eight sensors that can survey a circular area on the ground below the robot, with a total radius of 15 units. The ratio of the sensing radius compared to the survey radius ensures that it is possible that no two robots will overlap surveying the ground. In other words, a robot will be able to detect another robot, preventing the same area of ground being covered by both robots. However, this depends on the control system of the MAVs and their efficiency in generating the behavior of the MAVs.

2. Control system

The control of the MAV is in the typical pairing of conditions with actions. In other words, the action a robot takes is based on the robot's sensory information. As mentioned earlier, the robot has eight sensors for sensing other robots, each returning a bit for whether an entity is detected or not. Also, there are eight sensors for surveying the land. The eight survey sensors return two bits corresponding to the level of topology on the ground that is detected. Bit values representing the topology states are: 00 for level 0 (no topology), 01 for level 1 topology, 10 for level 2 topology, and 11which is not used. These topologies represent the level of interest in the location, i.e., places that are of more interest would have a higher topology value. The 8 bits from sensing and the 16 bits from the surveying make up the condition component of the condition-action pairing. The action portion of the rule is made up of four bits. The first bit tells whether the robot should move forward or hover in place. The other three bits correspond to a turn direction. The directions the robot can turn are the eight compass directions (N, NE, E, SE, S, SW, W, and NW). Table 1 shows all the three bit combinations that make up the degree change in direction.

The condition and action clauses are paired together to make one MAV rule of 28 bits. Table 2 shows an example of a MAV rule broken into the condition and action components. The sensors are numbered from zero to seven as the condition component in Table 2 is read from right to left. The sensors 0, 2, and 4 have detected another entity; thus, they each registered a one for the sensing data and the other sensors return zero. The topology sensors return varying values. Sensors zero through two detect topology level two while the remaining sensors detect topology level one. The bits in the action clause tell the MAV to move forward and turn +45 degrees, for that specific condition.

A robot's rule set is made up of one or more of these rules. The sensors are queried at each time step of the simulation and the sensory information is compared with the condition clauses of the rules. The condition clause with the best degree of match to the sensory data is selected. The degree of match is the difference between the condition component of the rule and the current sensor values. In the case that multiple rules qualify for the best match, then one of the best matching rules is chosen at random. If the degree of match is within a certain threshold, then the action clause of the rule is fired. In the simulation, every MAV is able to collect sensory information, determine the action to take, and execute the action. Although in the physical environment some moves may take longer than

others may, in the simulation all moves are assumed to take one time step, thus simplifying the simulation task.

Table 1. The turn bits of the action clause with the corresponding compass directions.

Bit Values	Relative Turn Direction(in degrees)
000	-135
001	-90
010	-45
100	0
110	+180/-180
101	+45
011	+90
111	+135

Table 2. An example MAV rule.

Condition		Action	
8 bits for sensing	16 bits for topology	Move	Angle
101010000	1010100101010101	1	101

3. Environment

As previously mentioned, there are three different levels of topology (Wu *et al.*, 1999). The levels of topology correspond to how interesting is the section of land. In other words, if the MAV is used for military surveillance then a section of land that contains a military base or airport is perhaps more interesting or more relevant than a flat, dry plain with farms. Therefore, in this example, the military base could be classified as a level one or level two and the flat plain could be classified as a level zero or no topology. The topology is normally defined in the following fashion. First, the boundaries of the survey area are defined and everything within is given the level zero topology value. The boundaries are defined so that if a MAV tries to go outside the survey area then the MAV is considered inoperable and can no longer function. Rectangular level one topologies can be defined within the survey boundaries. Smaller, rectangular level 2 topologies can be defined within the rectangular, level 1 topologies. Fig. 2 describes a topology with three level 2 topologies within a large level 1 topology. The level 2 topologies have one larger topology to the north of two smaller topologies.

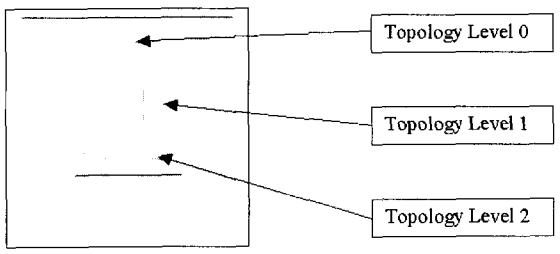


Fig. 2. The defined topology levels of the environment.

4. Simulation

In the experiments, each MAV is conFig.d with eight sensors, has a radius of five, survey range of 30, and a sensor range of 50 units. In the setup files, the rule set for each MAV is defined. All of the MAVs have the same rule set in the experiments, i.e., a homogenous team of robots. This was done because it is more difficult to evaluate the individual success of a robot in a heterogeneous team of robots. Using the simulator, each robot checks its sensory data during each time step and compares the data to the list of conditions in the rule set. When the sensory data matches a certain condition, then the corresponding action is fired. Since, there are 24 bits of sensory data then there are 2^{24} or 16,777,216 possible conditions. In the experiments, a maximum and a minimum rule set sizes have been set. An example of MAV control rules is provided in Table 3.

To evaluate the success of the team of MAVs, the percentage of surveillance area (with the added weight value for more significant topology) is calculated for all the MAVs. The value is averaged over the entire experiment time steps and is used as the final metric for evaluating the success of the team, representing the fitness function of the Genetic Algorithm. It should be noted that the credit assignment problem is eliminated using this approach, since the performance of the entire team is considered instead of individual robots (homogeneous members). The simulator runs without animation to speed up the testing cycle. However, configuring the input files to print a trace can produce a trace file that holds information about where the robots are located and what they are surveying during each time step. If the option to print a trace is enabled, then a file is generated from the simulation program that shows the position of every robot at each time step. This file can then be used to view the animation through a Java applet.

Table 3. Example of one MAV with 10 control rules.

010000101110111111000111	0	0
101000101001011010000110	1	90
000110001101010100001111	1	0
111001000110000100110111	1	315
101001010111111111111110	0	315
010000001010110000100100	1	270
010110001100101101001100	0	270
011110111101000011011100	0	225
010011110111101010000000	1	0
111110100010010011011010	0	270

5. Animation

A Java applet was developed to view the simulation of the MAVs. The applet displays data from trace file, as generated by the simulation. The applet provides the options of running, pausing, and stepping through the animation. In addition, the animation may be reset at any time or change the speed in which it executes. In the animation, the MAVs are started on the west boundary of the survey area. As mentioned earlier, a thin-lined box denotes the boundary area and the areas of interest are the rectangular shapes within the survey area. The screenshots illustrating the initial and mid-experiment animations using the Java applet-viewer are shown in Fig.s 3 and 4,

respectively. The MAV is the inner of the three circles. The second circle represents the area of the ground that the MAV can survey and the third circle represents the MAV sensor range for detecting the boundaries and other MAVs. The thin line from the center of the MAV to the edge of the MAV indicates the current direction that the robot is heading.

6. Evolution

A GA was used to evolve rule sets for controlling teams of MAVs. Each individual in a GA population represents a complete rule set. The fitness of a GA individual (a rule set) is determined by the performance of a team of MAVs using that rule set in the simulation. Consequently, the GA must execute one MAV simulation for every individual in its population. Table 4 gives the parameter settings for the GA that was used to learn the rule sets. We used a population size of 100 individuals and each run evolved for 150 generations. The evolved rule sets could vary in length (i.e. the number of rules), however the number of rules was set to 60. . Since each rule is 28 bits long, this translates the chromosome length of 1680 for each MAV.

The fitness of a particular rule set is determined by testing its effectiveness in the MAV simulation. For any GA run, the MAV parameters are held constant throughout the entire run. Table 4 gives the parameter values used in our MAV simulations. It is necessary to specify the number of MAVs, the number of sensors, the number of bits for the condition, the number of bits for the action, the size of the MAVs, the sensor range, the survey range of the MAVs, and the initial positions of the robots. An additional specification that is needed is the topology that describes the environment to test the MAVs.

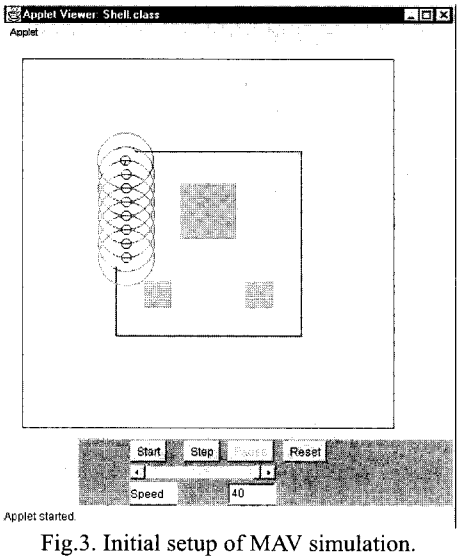


Fig.3. Initial setup of MAV simulation.

IV. Experiments

1. Experiment Setup

The experiments were performed on Sun Ultra 5/333 MHz Workstations, running Sun Solaris. Experiments were run using a number of such computers. On average, a Sun Ultra 5/333 could evolve a population of 100 MAVs for 150 generations in just over 24 hours. The experiments were typically divided among six to seven different machines. Table 4 shows the GA parameters and MAV parameters. In the first series of experiments, the topology description was fixed. The topology used had a 400x400 region with a level one 300x300 topology centered in the middle of the region. . The topology was identical to the one described previously, as shown in Fig. 1. The topologies (environments) were varied for the second set of experiments. In all the experiments, the data presented concerning the evolution program is based on three runs, i.e., the average of three repetitions of the experiments. Furthermore, all data that were gathered solely from the simulation program were based on 10 runs. The experiments were repeated to increase the reliability of the results.

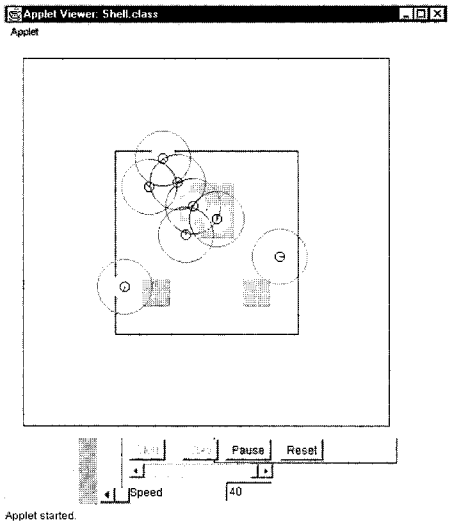


Fig. 4. The simulated MAV world.

2. Team size experiments

2.1 Simulation

The first series of experiments focused on investigating the effect of the MAV team size on the on the GAs ability to evolve effective and robust rule sets. The term *team size* refers to the number of MAVs used in a particular MAV simulation. For instance, a MAV team size of eight means that eight MAVs will be used to survey the land in that simulation. It

Table 4. Parameters of the program for GA and MAVs.

Parameter	Value
Population Size	100
Initial Population Size	2
Max Number of Generations	150
Chromosome Length	1680
Mutation Rate	0.005
MAV total	6 (varies in some)
Survey Range	15
Sensor Range	30
Number of Sensors	8
Radius	5
Condition Length	24
Action Length	4
Spacing	5

should be noted that, during a simulation, the MAVs could collide or leave the surveillance area, rendering them inoperable, reducing the number of MAVs operating for the rest of the simulation. Since a single GA run uses the same MAV simulation parameters throughout its entire run, the MAV team size is fixed for individual GA runs.

3. Team size experiments

3.1 Simulation

The first series of experiments focused on investigating the effect of the MAV team size on the on the GAs ability to evolve effective and robust rule sets. The term *team size* refers to the number of MAVs used in a particular MAV simulation. For instance, a MAV team size of eight means that eight MAVs will be used to survey the land in that simulation. It should be noted that, during a simulation, the MAVs could collide or leave the surveillance area, rendering them inoperable, reducing the number of MAVs operating for the rest of the simulation. Since a single GA run uses the same MAV simulation parameters throughout its entire run, the MAV team size is fixed for individual GA runs.

Using the GA and MAV parameters given in tables 3 and 4, we evolved rule sets for MAV team sizes ranging from 2 to 20 MAVs per team. Each experiment used a different MAV team size. Each experiment averaged the performance of three GA runs. The goal of these experiments was to see if these tests would yield an ideal team size for the simulated task. Fig. 5 shows the resulting fitness of the rule sets evolved in our experiments. The fitness of the evolved rule sets decreases as the number of MAVs reached 20. The descent in the fitness values appears to start when the GA is evolving rule sets for teams of 10 or more MAVs. We hypothesize that this decrease may be due to the environment becoming too crowded. The optimal number of MAVs for the environment size that we tested appears to be around six. With a small number of MAVs (e.g. two) the fitness value seemed relatively high; however, the deviation between the best fitness and the worst fitness is significantly larger.

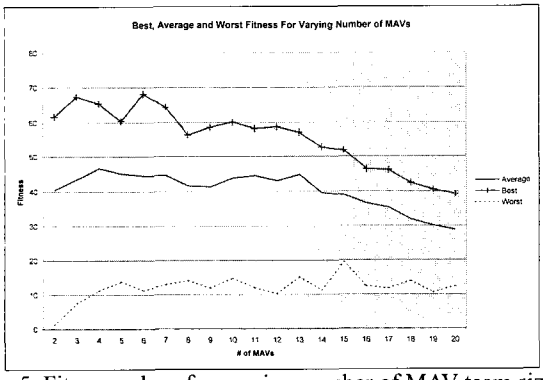


Fig. 5. Fitness values for varying number of MAV team sizes.

3.2 Scalability of evolved rule sets

The previous set of experiments looked at evolving rule sets for a fixed, known team sizes. A more interesting problem involves evolving rule sets that will work well for unknown or a range of team sizes. That is, we would like to evolve an effective rule set for controlling a team of MAVs in a particular task; however, the actual number of MAVs that will be available may not be known at the time that we are learning the rule set. Is it possible to evolve rule sets that will work well for team sizes other than the one used during its evolution?

In this second set of experiments, rule sets were evolved using team sizes of 2, 5, 10, 15, and 20 MAVs. The best rule set from generation 150 of these runs was then selected to be tested in simulations using teams of 2, 5, 10, 15, and 20 MAVs. Each selected rule set was tested10 times with each team size. Fig. 6 shows the fitness of robots using the various rule sets and the various testing population sizes. These results suggest that it is better to evolve rule sets using larger MAV team sizes if the actual deployed team is unknown in advance. For each rule set selected from a GA with a fixed team size, Table 5 shows the fitness or performance of that rule set in simulations using other team sizes.

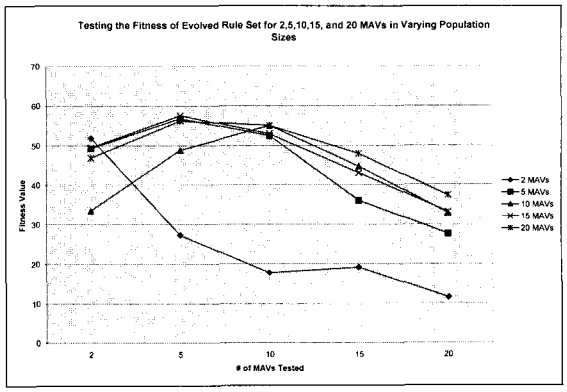


Fig. 6. The fitness of evolved rule sets in different team sizes.

In addition, the average fitness/performance of each rule set over all team sizes is calculated, excluding the diagonal elements, where the number of MAVs were the same for evolved and simulated cases. As shown in Table 5, the average performance of a rule set that was evolved using a team size of 20 is much better than the average performance of a rule set evolved using a team size of two. Performance gradually increases as the evolving team size increases. We speculate that lack of interaction between MAVs in small team sizes may play a large role in these results. The statistical analysis in the next will be used to support this hypothesis.

3.3 Statistical analysis

Statistical analysis was performed to help support or deny the hypothesis that it is better to evolve larger population of MAVs. The Student's t test is an acceptable method for comparing the two groups of data (Caprette, 1998). The t test determines if the two populations are the same based on the variable data that is collected. The t test was applied to the evolved rule sets of two MAVs and five MAVs. Calculating the value for the two samples, with six degrees of freedom, the critical value is 2.45 for a probability of 0.05, i.e., confidence level of 95%. Since the t value is greater than 2.45, it can be stated that the two groups are different. Applying the t test to the other comparisons yields similar results. The t values from comparing the various samples are listed in the Table 6. Look-

ing up the confidence probability yields the numbers included in Table 6. It is important to realize that the t tests that allow the rejection of the null hypothesis concern the rules evolved for two MAVs and any other rule set. There is a significant probability that the rule sets for 10 MAVs and 20 MAVs are different. The remainder of the evolved rule sets for 5, 10, 15, and 20 MAVs provide too similar results. It can be argued that the rule sets evolved for five MAVs and up include similar interactions with other robots when compared with the larger rule sets.

Table 5. The fitness of evolved rule sets in different team sizes and the averages for the varying evolved rule sets (computed excluding the diagonal elements).

# MAVs Tested in Simulation	# MAVS Evolved in GA				
	2 MAVs	5 MAVs	10 MAVs	15 MAVs	20 MAVs
2	51.899	49.242	33.493	49.481	46.835
5	27.294	56.871	48.835	57.608	56.268
10	17.724	52.456	55.056	52.901	55.074
15	19.143	36.032	44.694	43.008	47.906
20	11.489	27.587	32.795	33.120	37.314
AVERAGE	18.910	41.320	39.950	48.278	51.521

4. Topology experiments

4.1 Simulation

The focus of the second set of experiments was on whether genetic algorithms could develop a general solution evolved in one topology (environment) that would be useful for other topologies, i.e. testing the ability to generalize the simulation results. These experiments were performed focused on trying to find a good topology for evolving the MAVs. Five different topologies were developed and tested, as shown in Fig. 7. All of the topologies had an initial 300x300 area of interest in the center of the testing area. The topology was the only variable in these experiments. All the experiments were run with six MAVs, while other parameters were the default parameters described previously.

Table 6. The *t* values comparing the fitness of evolved rules for varying numbers of MAV team sizes, and the probabilities that the two rule sets are similar.

# of MAVs	2	5	10	15	20
2	---	t=3.373 p=0.025	t=4.067 p=0.010	t=4.710 p=0.005	t=8.048 p=0.005
5	t=3.373 p=0.025	---	t=0.195 P=*	t=0.883 P=*	t=1.622 P=*
10	t=4.067 p=0.010	t=0.195 P=*	---	t=1.248 P=*	t=2.463 p=0.050
15	t=4.710 p=0.005	t=0.883 P=*	t=1.248 P=*	---	t=0.555 P=*
20	t=8.048 p=0.005	t=1.622 P=*	t=2.463 p=0.050	t=0.555 P=*	---

There are many potential items to compare when evaluating the evolution in different topologies. First, the focus will be on how much the fitness values increased for each individual topology. Fig. 8 shows the fitness values through the evolution process with topology 1. As shown, there is a significant change in the fitness value, with an increase of approximately 40 points over the 150 generations. The fitness values for topology 2 are shown in Fig. 9. Topology 2 has an increase in fitness values of approximately 25. While the amount of increase is not the same, the percent increase is approximately the same, as the average fitness value in both experiments doubles throughout the evolution process. Fig.s 10, 11, and 12 show the fitness values for the other three topologies. It is interesting to observe that the starting and ending fitness values are almost identical for topologies 2, 3, 4, and 5. All of the average fitness values appear to start around 20 and increase to just over 40. However, it does appear that the MAVs have a different learning curve for each of the environments. For instance, it appears that the average fitness for topology 5 levels off at around 80 generations, however, the average fitness value for topology 4 appears to level off at around 120 generations. It appears that the different learning curve may be due to the complexity of the environment, although there is no direct proof of this hypothesis.

The next step after evolving the rules for each of the different topologies was to test the evolved rules in each of the five

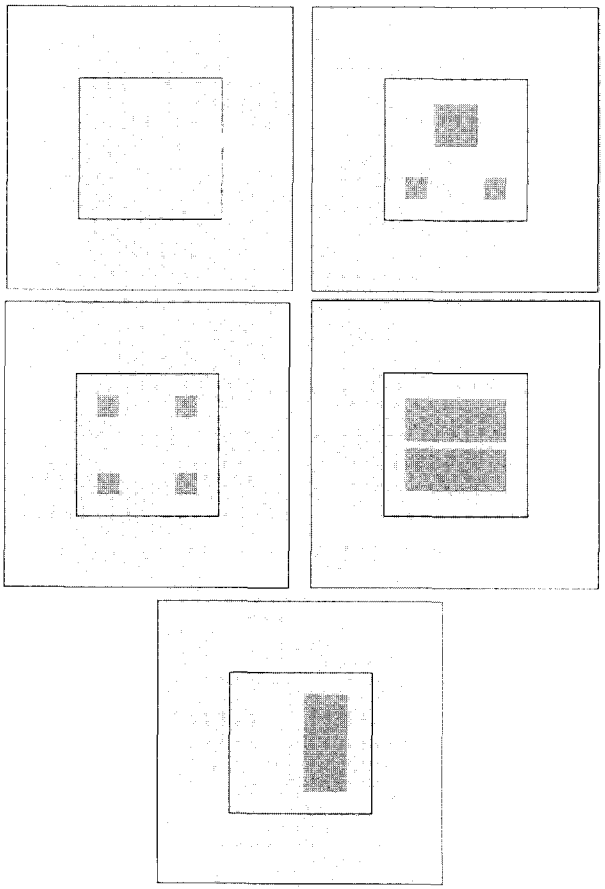


Fig. 7. Topologies 1, 2, 3, 4, and 5, from left to right and top to bottom.

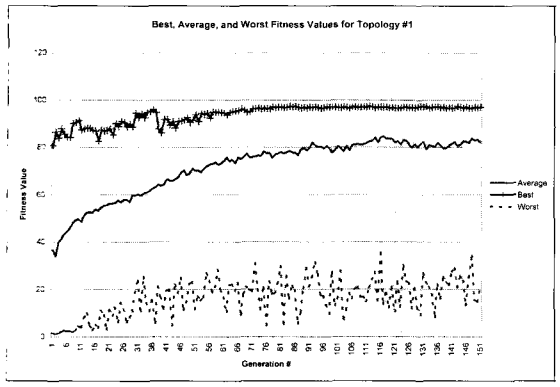


Fig. 8. Th best, average, and worst fitness values for topology 1.

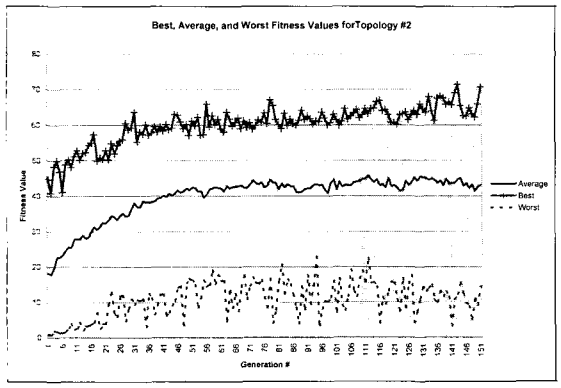


Fig. 9. The best, average, and worst fitness values for topology 2.

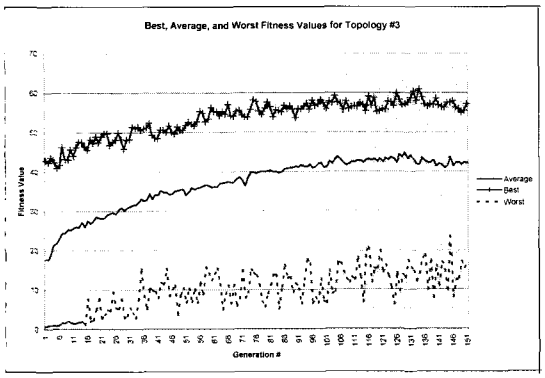


Fig. 10. The best, average, and worst fitness values for topology 3.

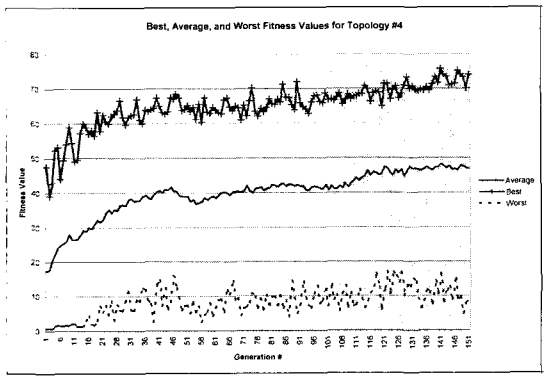


Fig. 11. The best, average, and worst fitness values for topology 4.

environments. Each simulator run testing an evolved rule set was repeated 10 times. Table 8 shows the average fitness for each rule set tested in every environment with the varying

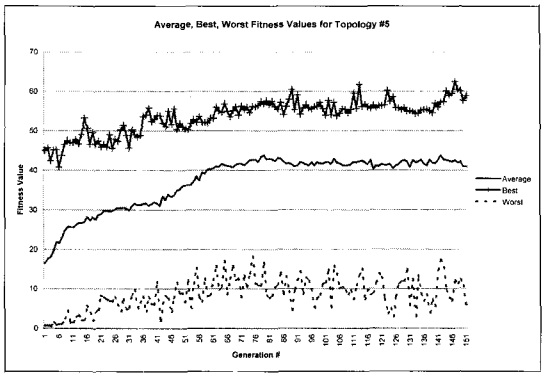


Fig. 12. The best, average, and worst fitness values for topology 5.

topology levels. Examining the data, it appears that usually the evolved rule sets perform the best when tested in the same topology in which it was evolved. While running the experiments, it was evident that the lower fitness values were usually attributed to MAVs becoming inoperable during the testing phase. For instance, when the evolved rule set for topology 4 was tested in topology 3 approximately half of the testing runs resulted in one or two MAVs becoming inoperable.

4.2 Statistical analysis

The student t test was applied to the data to see whether the evolved rule sets are different enough from each other. Table 7 shows the average and standard deviation of the fitness values for each of the evolved rule sets. It appears that the topology used in the team size experiments (topology 2) yielded the best results of the various topologies. The calculated t values are included in Table 8, along with the critical values. It is evident that the rules evolved from topology 1 are significantly different from the other topologies. We can reject the null hypothesis when comparing the results of topology 1 to the other topologies. This may be attributed to topology 1 not having many different levels of topologies. In other words, topology 1 is too simple or plain to evolve good rule sets. Therefore, we can hypothesize that it is better to evolve the rule sets with some arrangement of different levels, as in topologies 2 through 5.

V. Conclusion

1. Contributions

In this paper, it was shown that a realistic simulation model of distributed Micro Air Vehicles (MAVs) can be used to study the effects of varying MAV team size, and environmental topology on the simulated evolution of controllers for MAVs. In this paper, we investigate the use of GAs to evolve rule sets for controlling teams of distributed MAVs. We looked at two main aspects of this topic: (1) does the size of the team and the environmental topology affect the effectiveness of the rule sets that can be evolved, and (2) can we evolve rule sets that will work in simulations using team sizes and environmental topologies different from the ones that were used during the evolution of the rule set. Our results suggest that it is more difficult to evolve effective rule sets for larger team sizes than smaller team sizes. We speculate that this difficulty arises from the fact that rule sets for larger team sizes must be able to deal with more interactions than rule sets for smaller team

sizes. When the actual size of the team is unknown in advance, however, it appears to be better to evolve rule sets using larger team sizes. The larger sized teams appear to provide the learning process with more instances of interactions, allowing the GA to evolve a more complete rule set. The experiments also showed that it is good to have some somewhat more "complex" environmental topology, although the layout of the topology is not too important. The fitness values from the various topologies were approximately the same, and henceforth among complex topologies, the setups have minimal effects on the fitness of the evolved robots.

2. Limitations

Since the experiments were carried out in a simulated world, many limitations were not accounted for in these tests. Many of the variables of the physical world did not come into play in the simulated world. For instance, the robots were limited to two-dimensional motion. In the physical world, the robot would need to be able to ascend and descend vertically in the environment along with moving horizontally. In addition, other constraints on the robot were not taken into consideration such as battery levels. In the real world, the robots would base their control system not only on the sensing and survey data, but also on the internal sensors such as battery usage. Finally, the surveying of the robot was greatly simplified in the simulation. In the real world, the robots would have to perform image recognition or be given satellite maps and corresponding topology levels.

3. Future work

This project's future work can take many new directions. First, the dynamics of the physical world and the constraints of the robots can be added to the current software package. Three-dimensional maps can be added to the environment along with additional sensors for the robots to handle the three-dimensional world. A new environment can be developed, complete with buildings and recharging stations for the robots. Additionally, the information from these experiments can be tested in the physical world. Other areas would need to be explored and implemented such as image recognition. Implementing these experiments in the real world would convincingly show the feasibility of using genetic algorithms to evolve teams of flying robots, validating the results obtained from simulation experiments.

Table 7. Average fitness values for each evolved rule set tested in each of the five different topologies, and average and standard deviation of the five evolved rule sets in each of the five topologies.

Tested topology in Simulation	Evolved topology in GA				
	1	2	3	4	5
1	95.539	68.647	69.614	57.125	68.794
2	47.952	56.718	59.789	57.238	47.525
3	46.158	47.267	52.263	34.322	50.015
4	39.396	61.573	57.138	55.683	48.578
5	47.871	56.075	49.732	50.243	54.587
Average	55.383	58.056	57.707	50.922	53.900
Standard Deviation	22.720	7.853	7.741	9.709	8.751

Table 8. The t values comparing the different topology rule sets, and the probabilities that two of the rule sets are similar.

Topology #	1	2	3	4	5
1		t=2.766 p=0.025	t=2.791 p=0.025	T=2.355 p=0.050	t=2.566 p=0.050
2	t=2.766 p=0.025		t=0.045 p=*	t=0.665 p=*	t=0.342 p=*
3	t=2.791 p=0.025	t=0.045 p=*		t=0.709 p=*	t=0.387 p=*
4	t=2.355 p=0.05	t=0.665 p=*	t=0.709 p=*		t=0.328 p=*
5	t=2.566 p=0.050	t=0.342 p=*	t=0.387 p=*	t=0.328 p=*	

References

[1] A. Agah, and K. Tanie, "Robots playing to win: evolutionary soccer strategies." *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 632-637, 1997.

[2] A. Agah, and G. A. Bekey, "A genetic algorithm-based controller for decentralized multi-agent robotic systems." *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 431-435, 1996.

[3] G. Beni, and S. Hackwood, "The maximum entropy principle and sensing in swarm intelligence." *Toward a Practice of Autonomous Systems*, pp. 153-160, 1990.

[4] D.R. Caprette, Student's Test for Independent Samples. http://www.ruf.rice.edu/~bioslabs/tools/stats/ttest.html. 1998.

[5] D. Cliff, I. Harvey, and P. Husbands, "Explorations in evolutionary robotics." *Adaptive Behavior*, vol. 2, pp. 73-110, 1993.

[6] D. J. Collins, *Evolutionary Brains For Distributed Flying Robots*. M. S. Thesis, Department of Electrical Engineering and Computer Science, The University of Kansas, 1999.

[7] C. Debolt, C. O'Donnell, C. Freed, and T. Nguyen, "The Bugs "Basic UXO gathering system" Project for UXO Clearance and Mine Countermeasures." *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 329-334, 1997.

[8] 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." J.-A. Meyer, and S.W. Wilson, (Eds.) *From Animals to Animats*. MIT Press, Cambridge, Massachusetts, pp. 356-363, 1991.

[9] J. L. Deneubourg, G. Theraulaz, and R. Beckers, *Swarm-made architectures*. Toward a Practice of Autonomous Systems, pp. 123-133, 1990.

[10] M. Gen, and R. Cheng, "A survey of penalty techniques in genetic algorithms." *Proceedings of the IEEE International Conference on Evolutionary Computing*, pp. 804-809, 1996.

[11] D. Goldberg, *Genetic Algorithms in Search, Optimization and Learning*. Addison-wesley publishing Company, Inc., Reading, Massachusetts, 1989.

[12] S. Goss, and J. L. Deneubourg, "Harvesting by a group of robots." *Toward a Practice of Autonomous Systems*, pp. 195-204, 1990.

[13] J. J. Grefenstette, C. L. Ramsey, and A. C. Schultz, *Learning sequential decision rules using simulation models and competition.* Machine Learning, vol. 5, pp. 355-381, 1990.

[14] S. Hackwood, and G. Beni, "Self-organization of sensors for swarm intelligence." *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 819-829, 1992.

[15] K. Hosokawa, T. Tsujimorf, T. Fujii, H. Kaetsu, H. Asama, Y. Kuroda, and I. Endo, "Self-organizing robots with morphogenesis in a vertical plane." *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2858-2863, 1998.

[16] J. -H. Kim, H. -S. Shim, H. -S. Kim, M. -J. Jung, L. -H. Choi, and J.-O. Kim, "A cooperative multi-agent system and its real time application to robot soccer," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 638-643, 1997.

[17] C. R. Kube, and H. Zhang, Collective robotics: from social insects to robots. *Adaptive Behavior*, vol. 2, pp. 189-218, 1994.

[18] H. H. Lund, and O. Miglino, "From simulated to real robots." *Proceedings of the IEEE Conference on Evolutionary Computing*, pp. 362-365m, 1996.

[19] M. J. Mataric, "Minimizing complexity in controlling a mobile robot population." *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 830-835, 1992.

[20] Mirosot. http://www.fira.net/fira_games/mirosot. html.1999.

[21] F. Mondada, http://www.epfl.ch/lami/robots/K-family/Khepera.html. 1999.

[22] S. -W. Park, J. -H. Kim, E. -H. Kim, and J. -H. Oh, "Development of a multi-agent system for robot soccer game." *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 626-631, 1997.

[23] J. F. Peters, J. Wong, S. Ramanna, and S. A. Ehikioya, "Evolution of competing situated robots: concepts and experiments with a java applet." *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 3371-3375, 1998.

[24] L. Rodrigues, M. Prado, P. Tavares, K. da Silva, and A. Rosa, "Simulation and control of biped locomotion-GA optimization," *Proceedings of the IEEE Conference on Evolutionary Computing*, pp. 390-395, 1996.

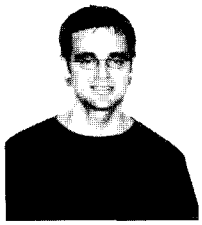
[25] T. Shibata, and T. Fukuda, "Coordinative behavior in evolutionary multi-agent robot system." *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 448-453, 1993.

[26] T. Sugar, and V. Kumar, "Decentralized Control of Cooperating Mobile Manipulators." *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2916-2921, 1998.

[27] G. Theraulaz, S. Goss, J. Gervet, and J. L. Deneubourg, "Task differentiation in Polistes wasp colonies; a model for self-organizing groups of robots." *From Animals to Animats*, pp. 346-355, 1991.

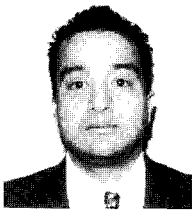
[28] T. Ueyama, T. Fukuda, and F. Arai, "Structure configuration using genetic algorithm for cellular robotic system." *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1542-1549, 1992.

[29] A. Wu, A. Schultz, and A. Agah, "Evolving control for distributed micro air vehicles," In *Proceedings of the 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Monterey, California, November 1999.



Daniel James Collins

VI is a Consultant for Celeritas Technologies located in Overland Park, Kansas. He received a B.S. in Computer Science and Mathematics with Honors from Baker University in 1997 and a M.S. in Computer Science with Honors from the University of Kansas in 1999. His research interests include robotics, genetic algorithms, neural networks, and fuzzy logic.



Arvin Agah

Dr. Arvin Agah is Assistant Professor of Electrical Engineering and Computer Science at the University of Kansas. His research interests include human interactions with intelligent systems (robots and agents). He has published one book and over 70 refereed articles in these areas. Dr. Agah has been a co-investigator on numerous projects funded by NSF, NASA, DARPA, and Sprint. He has taught courses in artificial intelligence, robotics, software engineering, computer systems design laboratory, and intelligent agents. He has served as the technical program committee member, conference session chair, and organizing committee member for various international technical conferences. He is a senior member of IEEE and a member of ACM. Dr. Agah received his B.A. in Computer Science with Highest Honors from the University of Texas at Austin (1986); M.S. in Computer Science from Purdue University, West Lafayette, Indiana (1988); M.S. in Biomedical Engineering from the University of Southern California, Los Angeles, California (1993); and Ph.D. in Computer Science from the University of Southern California (1994).

Dr. Agah has been a member of research staff at Xerox Corporation's Webster Research Center, Rochester, New York; IBM Corporation's Los Angeles Scientific Center, Santa Monica, California; Ministry of International Trade and Industry's Mechanical Engineering Laboratory, Tsukuba, Japan; and Naval Research Laboratory's Navy Center for Applied Research in Artificial Intelligence, Washington, D.C. He has been an instructor at Mansfield Business School, Austin, Texas; Purdue University's Department of Computer Science, West Lafayette, Indiana; and University of Tsukuba's Department of Engineering Systems, Tsukuba, Japan. He has also worked as a systems analyst and software engineer for entertainment law firms and management companies in Century City and Beverly Hills, California.