AFRL-IF-RS-TR-2002-99
Final Technical Report
May 2002

SIMULATION OF A SWARM OF UNMANNED COMBAT AIR VEHICLES (UCAVS)

University of Central Florida

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2002-99 has been reviewed and is approved for publication.

APPROVED:

STEVEN M. ALEXANDER, 1stLt, USAF
Project Engineer

FOR THE DIRECTOR:

JAMES W. CUSACK, Chief
Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
MAY 2002

3. REPORT TYPE AND DATES COVERED
Final Jun 01 – Feb 02

4. TITLE AND SUBTITLE
SIMULATION OF A SWARM OF UNMANNED COMBAT AIR VEHICLES (UCAVS)

5. FUNDING NUMBERS
C - F30602-01-1-0559
PE - 62702F
PR - 558B
TA - UC
WU - AV

6. AUTHOR(S)
Kuo-Chi "Kurt" Lin

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Central Florida
Institute for Simulation and Training
3280 Progress Drive
Orlando Florida 32826

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFSB
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2002-99

11. SUPPLEMENTARY NOTES
AFRL Project Engineer: Steven M. Alexander, 1stLt, USAF/IFSB/(315) 330-4304/Steven.Alexander@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

This research focuses on the control of the collective performance of a swarm of UCAVs. One control command string controls the motion of all UCAVs in a mission. There is no explicit coordination among diem. If the control command string is properly chosen, the motion of the swarm of UCAVs will perform well collectively. Genetic Algorithms (GA) are used in this research to find suitable control command strings. It is an effective method to get a very good solution if the mathematical optimum is not necessary.

The objective is for UCAVs to maximize surveillance coverage in 20 time steps. The final fitness value is the average of the coverage percentiles of five 20-time-step results. Using GA, the best control command string found to control 10 UCAVs has the fitness value 0.9603. This is an average of 96.03 % of coverage, a very good result.

Parametric and robustness analyses show that control may not be very robust. Monte Carlo simulation in conjunction with Genetic Algorithm is used to evolve robust control when wind-gust disturbance exists. The results of different approaches are compared.

14. SUBJECT TERMS
UCAV, Multi-Agent System, Genetic Algorithm, Command and Control, Autonomous Control, Cooperative Agents

15. NUMBER OF PAGES
15

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

1. **Introduction** **1**
2. **UCAV Model** **2**
3. **GA Representation** **2**
4. **Simulation** **4**
5. **Fitness Function** **4**
6. **Results** **4**
7. **Robustness Analysis** **4**
8. **Summary and Future Research** **9**
9. **References** **10**

List of Figures

Figure 1. UCAV motion model 2
Figure 2. The positions UCAVs entering the battlefield 4

List of Tables

Table 1. Decoding a command string, command string used for this example: 3
Table 2a. GA with deterministic fitness functions, control command strings 5
Table 2b. GA with deterministic fitness functions, comparisons with MC Simulation 6
Table 3a. GA with disturbance in the fitness functions, but only one evaluation, control command string 6
Table 3b. GA with disturbance in the fitness functions, but only one evaluation, \comparisons with MC simulation 7
Table 4a. GA with partial MC simulation using average as the fitness value, control command string 7
Table 4b. GA with partial MC simulation using average as the fitness value, comparisons with MC simulation 8
Table 5a. GA with partial MC simulation using average as the fitness value, control command string 8
Table 5b. GA with partial MC simulation using average as the fitness value, comparisons with MC simulation 8

INTRODUCTION

The control of autonomous Unmanned Combat Air Vehicles (UCAVs) has drawn attention from researchers. [1, 2] The problem of controlling a swarm of vehicles to do collaborative tasks is more difficult. Researchers have suggested an approach that uses one command string to control the motion of all UCAVs in a mission. [3-6] Each UCAV moves according to the control decoded from the same control command string. There is no explicit communication among them. However, the decoding of a control command string partially depends on the UCAVs surrounding it. If the control command string is properly chosen, the motion of the swarm of UCAVs will reflect the desired collective behavior. This approach was inspired by social insects, such as ants. No single ant has the global view that can guide the group in the collaborative task. It only follows the environment and local situation. Nevertheless, the whole group of ants can perform complicated tasks. The research reported in this paper uses this approach.

Instead of choosing the control command string manually, the authors propose to search through the solution space using a Genetic Algorithm (GA) approach. This allows sensory information to be automatically tied to a set of commands, forming a rule set for all possible sensory input combinations; without any help from the model developer. The programmer is no longer concerned with whether it is "good" for a UCAV to turn left when approaching another UCAV from the front, or if UCAVs should stay a certain distance apart from one another to cover more ground. The definition of an appropriate fitness function for the employed GA is enough to cause these rules to surface on their own.

GAs have been used for a wide variety of applications, and are generally useful for solving optimization problems [7, 8]. These algorithms are modeled after the process of evolution observed in biology. A GA works with a population of individuals where each individual represents a potential solution to the problem to be solved. These individuals are typically encoded as binary strings. The initial population of a GA may be randomly generated or seeded with user generated solutions. The GA then proceeds through the following steps:

1. Individuals in the current population are evaluated on their effectiveness as a solution to the problem to be solved. Better solutions are assigned higher fitness values, and worse solutions, lower fitness values.
2. If the stopping condition is satisfied, stop the evolution process and return the best solution. The stopping condition may be to find a solution that meets a minimum fitness level, or to run the system until it exceeds a maximum number of generations.
3. A selection function selects the individuals that reproduce, and consequently, contribute information to the next generation. This selection function is fitness proportionate, causing more fit individuals to be more likely to contribute to the next generation.
4. The selected parents undergo genetic reproduction during which genetically-inspired operators (such as crossover and mutation) create offspring from the selected parents.
5. The new population of offspring individuals becomes the new current population.
6. Go to step 1.

Over time, a GA is able to evolve better and better solutions in its population.

UCAV MODEL

The battlefield is composed of grid points. The UCAV moves on the grid points only. The coordinates of the UCAV are (x, y, θ), where position (x, y) are integers and heading angle θ is measured counterclockwise from the East, and can only be one of the following values: 0, 45, 90, 135, 180, 225, 270, or 315 degrees. The speed of the UCAV is (constant) 10 grid points per time step. When the heading angle is 45,135, 225, or 315 degrees, the UCAV moves 7 diagonal grid points, which is equivalent to 9.9 grid points. The model is shown in Figure 1.

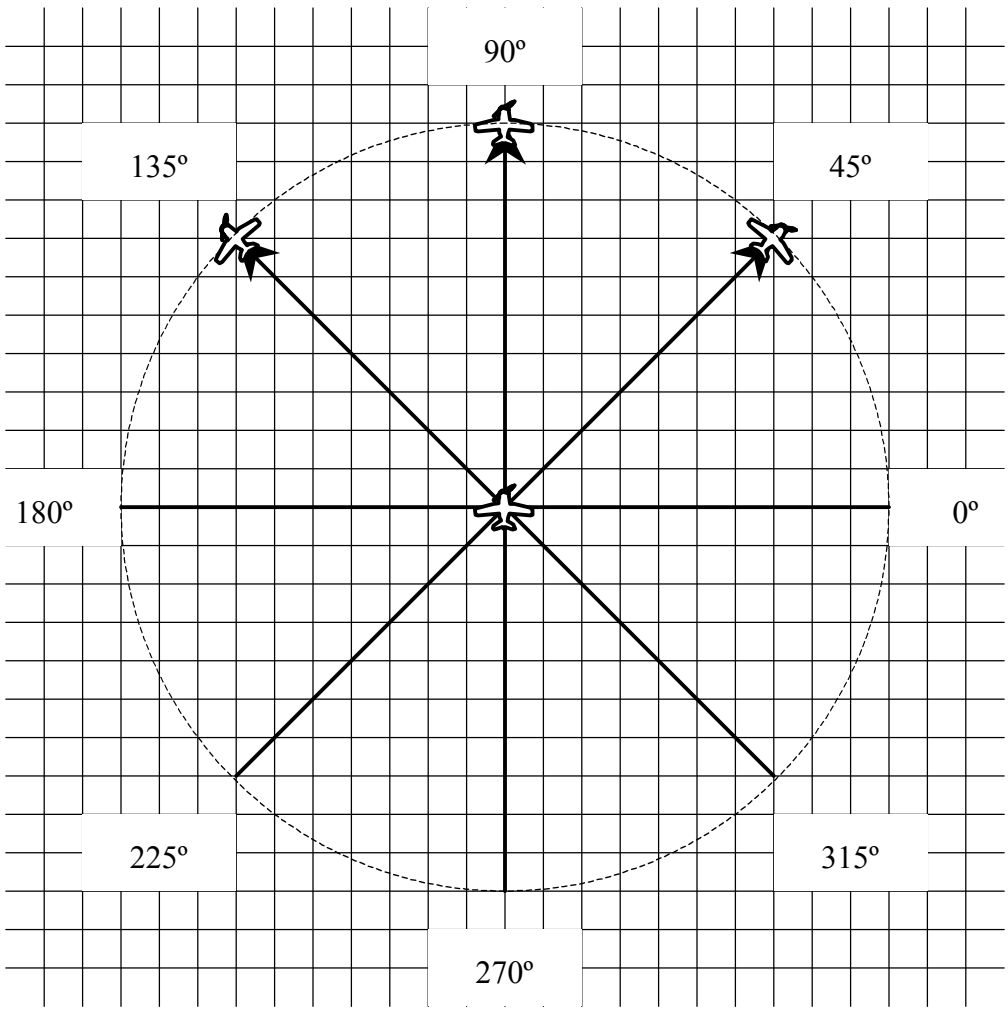

Figure 1. UCAV motion model

GA REPRESENTATION

The motion of each UCAV in a time step is controlled by a 2-bit signal. If the control is either (0, 0) or (0, 1), the UCAV maintains its heading angle. If the control is (1, 0), the UCAV turns -45º. If the control is (1, 1), the UCAV turns +45º.

The same control command string controls all the UCAVs in the battlefield. It is a 32-bit binary string composed of 16 2-bit control pairs. Four event sensors detect the event surrounding the UCAV. Using the UCAV orientation, the sensor outputs are 4-digit binary codes representing events in front, right, left, and back, respectively. Since four bits represent the sensors, there are sixteen possible combinations of sensory inputs. Hence we made the sensor value, *S*, an integer varying from zero to fifteen, given by:

$$S = 2^3 * f + 2^2 * r + 2^1 * l + 2^0 * b \qquad [1]$$

where *f* is the front sensor input, *r* is the right sensor input, and *l* and *b* are the left and back sensor inputs, respectively. The above equation simply translates sensor inputs from binary to decimal format.

The on-board computer decodes the control command string based on the sensor outputs to decide the necessary maneuver. The decoding scheme is shown in Table 1 using a sample control command string.

Table 1. Decoding a command string, command string used for this example:
0 0 0 0 1 0 1 0 1 1 1 1 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 0 1 0 (32-bits)

Events	Sensor	***S***	Bits	Motion	Direction
None	0000	0	1,2	(0, 0)	same
b	0001	1	3,4	(0, 0)	same
l	0010	2	5, 6	(1, 0)	-45
l, b	0011	3	7, 8	(1, 0)	-45
r	0100	4	9, 10	(1, 1)	+45
r, b	0101	5	11, 12	(1, 1)	+45
r, l	0110	6	13, 14	(0, 0)	same
r, l, b	0111	7	15, 16	(0, 0)	same
f	1000	8	17, 18	(1, 0)	-45
f, b	1001	9	19, 20	(1, 0)	-45
f, l	1010	10	21, 22	(1, 0)	-45
f, l, b	1011	11	22, 24	(1, 0)	-45
f, r	1100	12	25, 26	(1, 1)	+45
f, r, b	1101	13	27, 28	(1, 1)	+45
f, r, l	1110	14	29, 30	(1, 0)	-45
f, r, l, b	1111	15	31, 32	(1, 0)	-45

SIMULATION

The battlefield is a grid of 201 by 201 points. Ten UCAVs enter the field from the boundary, as shown in Figure 2. When two UCAVs are closer than or equal to four grids apart, they collide and are taken out of the simulation. The UCAVs that go out of the boundary are also taken out of the simulation. The simulation lasts 100 time steps.

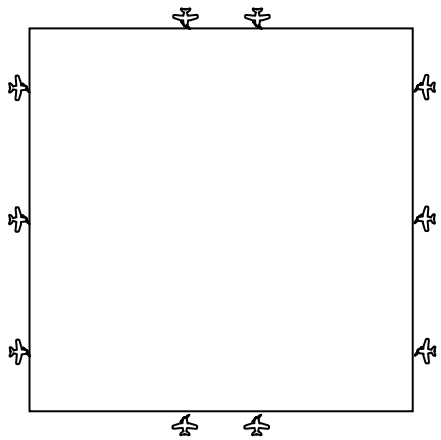

Figure 2. The positions UCAVs entering the battlefield

FITNESS FUNCTION

The fitness function is defined as the coverage of UCAV surveillance sensor in 20 time steps. Coverage is defined as the percentage of the field covered by the surveillance sensor ranges of all UCAVs. The overlap and repeated grid points are only counted once. The final fitness value is the average of the coverage percentiles of the five 20-time-step results. The range of a surveillance sensor used in the simulation is a circle of 24-grid-point radius.

RESULTS

We have made many GA runs, keeping track of the maximum, minimum, mean and standard deviation of the fitness of the control command strings. The best control command string is {0 0 1 0 0 1 0 1 1 1 0 1 0 0 1 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 1 0} with fitness value of 0.9603. It represents an average of 96.03% of coverage; a very good result. Other results will be discussed in more detail in the following sections.

We set the GA runs to stop at 25 generations in most of the cases. If we extend the number of generations, we may get better results. However, since GA is a probabilistic algorithm, there is no guarantee that it will reach a certain fitness value. In a typical GA run, the fitness value increases rapidly in the first few generations and levels off. Therefore, GA is an effective method of getting a very good solution if the mathematical optimum is not necessary.

ROBUSTNESS ANALYSIS

Genetic Algorithms can find control command strings that perform satisfactorily. However, the GA runs reported in the last section are based on a deterministic system. If there is uncertainty,

will the control command strings still perform well? This question can be answered by robustness analysis. Disturbance from the environment or the battlefield situation can lead to large deviations from the desired results. The following are results of simulations using certain disturbances as examples to demonstrate the influences of disturbances. In these examples, think of the disturbance as a wind gust that instantly changes the heading of a UCAV. The occurrences of disturbance are random with certain percentages as their probabilities.

Monte Carlo (MC) simulation [9] is used to evaluate the robustness. The simulation procedure is the same as the procedure reported in the previous sections, except disturbances are added. A random number generator is used to decide whether disturbance affects a particular airplane in a certain step. When the simulation is repeated, a different seed is assigned to the random number generator; therefore, the occurrences of the disturbances are different from simulation to simulation. In the following tests, the disturbance probability is set to 5% and the simulation is repeated 1,200 times. The averages and standard deviations are used to represent the performance and robustness of a certain control command string.

In the following paragraphs, four sets of results are compared with MC simulations. The first one is the set of results obtained from the GA with deterministic fitness functions, as described in the previous sections. In other words, the wind-gust disturbance is not considered when running these GAs. Table 2a shows the better control command strings evolved from GA. Table 2b shows the comparisons with MC simulation results. Column two shows the fitness values obtained from GA runs with deterministic fitness value evaluations. The Monte Carlo simulation results, columns three and four, show that control command string #1 has good performance (high average) and robustness (low standard deviation).

Table 2a. GA with deterministic fitness functions, control command strings

#	Control Command String
1	001001011101001111011010111110110
2	011010101100111011111111111110111
3	011010001100001011101010111110000
4	001101101101010111011001111111101
5	001101101101010111011001111111111
6	001010101101111111111101111001100
7	001110111101000110111010110011110
8	010110101101111011111111111101011
9	010010101101111011111111111100101
10	001010011101011011101010110011110

Table 2b. GA with deterministic fitness functions, comparisons with MC Simulation

#	GA fitness w/o dist.	MC sim. avg.	MC sim. std.
1	0.9603	0.9006	0.0342
2	0.9597	0.8452	0.0493
3	0.9486	0.8444	0.0490
4	0.9473	0.9006	0.0355
5	0.9473	0.8926	0.0340
6	0.9462	0.8494	0.0564
7	0.9441	0.8506	0.0560
8	0.9411	0.8506	0.0560
9	0.9411	0.8955	0.0340
10	0.9409	0.8874	0.0329

The second set of results is obtained from the GA runs with 5% wind-gust disturbance added. The wind-gust disturbance is implemented in the fitness evaluation and the simulation is only run once. Because of the uncertainty in the fitness evaluation, the good fitness values obtained by GA may be accidental. The controls evolved from this approach may not perform well in different simulations. Table 3a shows the better control command strings evolved from GA with 5% wind-gust disturbance. Table 3b shows the comparisons with MC simulation results. The second column in Table 3b shows the fitness values of the direct GA runs with wing-gust disturbance. The third column is the fitness values (surveillance coverage) of the same set of control strings but without disturbances. The disturbance changes the surveillance coverage in both directions, increasing and decreasing. GA, by its nature, retains the simulations with better results. As discussed in the previous section, those good results may be accidental. The performance of each control string is evaluated by MC simulation. The average coverage and standard deviations are listed in column four and five, respectively. The results show that not all control command strings performs well in MC simulations. Among them, the control command string #10 has good performance and robustness.

Table 3a. GA with disturbance in the fitness functions, but only one evaluation, control command string

#	Control Command String
1	011110111100011110011010111111101
2	000010000100111111011011111111110
3	011110111100011110011010111110010
4	010100001100010111011010111110010
5	011110111111011110011010111110010
6	011110011100111110011010111111101
7	011110111100011110011010111111110
8	011110101101011111011010111110011
9	011110011100111110011010111111011
10	011110111111011110011010111111110

Table 3b. GA with disturbance in the fitness functions, but only one evaluation, comparisons with MC simulation

#	GA fitness with dist.	Fitness w/o dist.	MC sim. avg.	MC sim. std.
1	0.9655	0.8656	0.9029	0.0327
2	0.9637	0.7266	0.8374	0.0688
3	0.9625	0.8744	0.9001	0.0339
4	0.9620	0.9046	0.8458	0.0701
5	0.9598	0.9172	0.9076	0.0316
6	0.9590	0.9370	0.8950	0.0336
7	0.9589	0.8744	0.9036	0.0321
8	0.9567	0.9119	0.8722	0.0424
9	0.9543	0.9226	0.9034	0.0317
10	0.9542	0.9172	0.9105	0.0314

The third and fourth sets of results are obtained from GA runs that incorporated a "partial" MC simulation, which is different in concept from the MC simulation that is used to evaluate the results. The wind-gust disturbances change the positions or the headings of the UCAVs. Those are equivalent to new sets of initial conditions. Therefore, the partial Monte-Carlo simulation is set up as follows. One hundred sets of initial positions and headings of the 10 UCAVs are selected randomly. The simulations are run for only 20 time steps without wind-gust disturbance. Simulations are repeated 100 times. The fitness value is either the average or the standard deviation of the 100 runs.

Table 4a shows the better control command strings obtained from the GA runs with partial MC in fitness evaluations. Table 4b shows the comparisons with the MC simulations. The second column shows the averages of the partial MC runs, which are used as the fitness values of the GA. The MC simulation results are shown in columns four and five. The fact that all standard deviations are relatively high (all above 0.075) implies that this set of data does not contain any robust control command strings.

Table 4a. GA with partial MC simulation using average as the fitness value, control command string

#	Control Command String
1	000100101110011011101010110011 01
2	001110100011111011111011110000 00
3	001010100011111011111011110001 00
4	000010100011111011111011110001 00
5	000010100011111011011011110001 00
6	001010100011111111111011110001 00

Table 4b. GA with partial MC simulation using average as the fitness value, comparisons with MC simulation

#	GA partial MC sim. avg.	GA partial MC sim. Std.	MC sim. avg.	MC sim. std.
1	0.9162	0.0545	0.7968	0.0755
2	0.9159	0.0517	0.6905	0.0917
3	0.9151	0.0575	0.7253	0.0917
4	0.9149	0.0531	0.6703	0.1041
5	0.9147	0.0526	0.6695	0.1029
6	0.9141	0.0532	0.7411	0.0909

Table 5a shows the better control command strings obtained from the GA runs with partial MC in fitness evaluations. Table 5b shows the comparisons with the MC simulations. The second column shows the results of the partial MC runs using standard deviations as the fitness values of the GA. The MC simulation results are shown in columns four and five. This set of data, in general, gives more robust control command strings than the set shown in Table 4a and 4b. The fact that the average coverages shown in Table 5b are all above 86% indicates that performance is not sacrificed.

Table 5a. GA with partial MC simulation using average as the fitness value, control command string

#	Control Command String
1	000110001100010110111010110001 10
2	010110001100010110111010110001 11
3	000110001100010110111010110001 00
4	000110001111010110111010110001 10
5	001110001101011010011010110001 11
6	011110001101011010011010110001 10

Table 5b. GA with partial MC simulation using average as the fitness value, comparisons with MC simulation

#	GA partial MC sim. avg.	GA partial MC sim. Std.	MC sim. avg.	MC sim. std.
1	0.9177	0.0327	0.8626	0.0533
2	0.9180	0.0331	0.8618	0.0549
3	0.9182	0.0333	0.8623	0.0544
4	0.9189	0.0339	0.8641	0.0545
5	0.9168	0.0341	0.8890	0.0379
6	0.9168	0.0342	0.8899	0.0372

SUMMARY AND FUTURE RESEARCH

To evolve robust control using Genetic Algorithms, Monte Carlo simulation is needed in the fitness evaluations. However, Monte Carlo simulation is computationally expensive. The authors have tried four different approaches. None of them can guarantee robustness of control. However, they can provide a set of candidates. Running MC simulation using this limited candidate set can save a lot of time.

There are several possible directions for future research. The first one is to improve the approach reported in this paper. For example, the GA with partial MC simulation in fitness evaluations can perform better. The partial MC simulation is designed to reduce the simulation, and hence the GA, run time. However, because it only runs 20 time steps, instead of 100 time steps in the regular simulation, the effects of losing UCAVs are downplayed. In other words, if two UCAVs are lost due to collision, the coverage percentage may not suffer much in the 20 step time. But if the simulation keeps running to 100 step time, the coverage will be reduced by a much larger amount. To solve this problem, the fitness values of the GA with partial MC simulation can add a weighting factor: (# of surviving UCAVs/10). If all 10 UCAVs survive, the factor is one. Otherwise, the fitness is reduced according to the number of UCAVs lost.

The second possible research direction is to try multi-objective GA. [10, 11] Both performance (average coverage) and robustness (standard deviation) are used as the fitness functions. The multiple-objective optimization uses Pareto optimality, i.e., instead of looking for a single solution, it looks for a set of solutions called Pareto-optimal set. A Pareto-optimal solution is a solution that is not dominated by any other solution in the solution pool.

The third possible research direction is to study the system parameters' influence on the robustness of control. One such parameter is the sensor range. There are two kinds of sensors, the surveillance sensor and event sensors. The surveillance sensor is used to survey the battlefield. The range of the surveillance sensor is limited by the capability of the sensor, and therefore, is treated as a fixed parameter. The event sensor is used to detect events, such as collision threats and the boundaries. The event sensor need not be an on-board system. Information, such as where other UCAVs are and how close they are to the boundary, can be sent to the UCAV from a ground station or other information sources. Therefore, the range of the event sensor can be viewed from another perspective, namely the distance at which the UCAV reacts to an event. For example, if the range of the event sensor is 1 km, it really means that the UCAV may make a maneuver when another UCAV (or the boundary) is within 1 km. Maybe it is better to call it “reaction distance.” The influence of reaction distance on the robustness of control is an interesting topic.

REFERNCES

1. Wills, L., et al, “An Open Control Platform for Reconfigurable, Distributed, Hierarchical Control Systems”, *Proceedings of the Digital Avionics Systems Conference*, Philadelphia, PA, October 2000, vol. 1, pp. 4D2/1 -4D2/8.
2. Grecu, D., Gonsalves, P., “Agent-Based Simulation Environment for UCAV Mission Planning and Execution,” *Proceedings of the 2000 AIAA Guidance, Navigation and Control Conference*, Denver, CO, August 2000, pp. 14~17.
3. Wu, A. S., Schultz, A. C., Agah A., “Evolving control for distributed micro air vehicles”, *Proceedings of the 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Monterey, CA, November 1999, pp. 174-179.
4. Collins, D. J., Agah, A., Wu, A. S., Schultz, A.C., "The effects of team size on the evolution of distributed micro air vehicles", *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*, Las Vegas, NV, July 2000, pp. 949-956.
5. Alexander, S., Sisiti, A., Lin, K. C., "Autonomous UCAVs in a Synthetic Battle Field", Proceedings of 2001 Summer Computer Simulation Conference, Orlando, FL, July 2001, pp. 310~314.
6. Lin, K. C., Farr, S., Sisiti, A., Alexander, S., "Dynamic Situation Assessment and Prediction on Controlling a Swarm of UCAVs", Proceedings of the Third Collaborative Technologies and Systems Symposium, San Antonio, TX, Jan. 2002, pp. 207~212.
7. Holland, J. H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975.
8. Goldberg, D. E., "Genetic Algorithms in Search, Optimization, and Machine Learning, Reading", Addison-Wesley. 1989.
9. Dubi, A., "Monte Carlo applications in systems engineering", Wiley, NY, 2000.
10. Goldberg, D. E., “Genetic Algorithms in Search, Optimization, and Machine Learning”, Addison-Wesley, Reading, MA, 1989.
11. Zitzler, E., “Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications”, Ph.D. Dissertation, Swiss Federal Institute of Technology (ETH) Zurich. TIK-Schriftenreihe Nr. 30, Diss ETH No. 13398, Shaker Verlag, Germany, ISBN 3-8265-6831-1, December 1999.