EVOLVING BEHAVIORS FOR A SWARM OF UNMANNED AIR VEHICLES

Paolo Gaudiano and Eric Bonabeau
Icosystem Corporation
10 Fawcett Street
Cambridge, MA 02138, USA
{paolo, eric}@icosystem.com

Ben Shargel
Courant Institute
New York University
New York, NY 10012, USA
Bls272@courant.nyu.edu

1. ABSTRACT

We have previously reported on a project involving the control of a swarm of Unmanned Air Vehicles (UAVs) carrying out search or search-and-destroy missions. We developed and tested (in simulation) a number of strategies for swarm control, and proposed systematic evaluation techniques and performance metrics.

In this paper we report some additional results in which we evolved some of the swarm control parameters using a Genetic Algorithm (GA). While the improvements were modest, the results show how evolutionary computing algorithms can be used to facilitate the design of swarm control algorithms.

2. INTRODUCTION

We have developed a simulation tool to test control strategies for a swarm of Unmanned Air Vehicles (UAVs). We focused on two types of missions: (1) search missions, in which a swarm of UAVs enters an area and tries to identify a number of stationary targets; (2) suppression of enemy air defense (SEAD) missions, in which the targets are moving, and the UAVs attempt to strike them.

We have previously published conference papers describing the model and a number of results [1,2]. The main focus of our prior publications has been on establishing performance metrics and measuring systematically the performance of the swarm as conditions change. We believe that this sort of systematic evaluation is critical for the widespread adoption of swarm approaches to control in military and civilian applications.

In this paper we focus on an extension of our prior work in which we used a Genetic Algorithm [3,4] to evolve parameters for the behavior of UAVs during the SEAD mission. The behavior of UAVs in the SEAD mission is based on a small set of behaviors and transitions (5 possible behaviors and 11 transitions). However, even with this simple control model, the number of possible combinations of parameters is rather large. Hence we developed a GA to search the parameter space.

The remainder of the article describes the simulation system we developed, the GA, and our results.

3. MODEL OVERVIEW

In order to study UAV control strategies, we have developed an agent-based simulation tool. The following is a list of the assumptions and functionality we designed into the simulator:

- The terrain is defined as a rectangular region (actually a parallelepiped in 3D), which for convenience is subdivided into a grid of arbitrary coarseness. The grid is used primarily to determine coverage and to track “pheromone” signals left by each UAV during target search.
- Each UAV is able to fly at variable speed with independent pitch and yaw control. Control dynamics are simplified by specifying a maximum turn rate, and the ability to increase or decrease thrust.

- Each UAV is equipped with various sensors:
  - One forward-looking, cone-shaped ground sensor with adjustable radius and angular aperture, to detect terrain and possible targets. The ground sensor is stochastic, with the probability of detecting a target dependent on distance, elevation, and the amount of time spent flying over a given terrain cell.
  - One circular sensor to detect the presence of other UAVs within a prescribed (adjustable) radius.
  - "Pheromone" sensor: each UAV can detect, within a small rectangular region centered on itself, how much each terrain cell has been covered by itself or by other UAVs. This is used to promote efficient coverage of the search area (see below).
  - GPS-like positioning capability.
- Each UAV is aware of the terrain boundaries and will turn as it approaches each boundary to remain within the target area.
- Communication between UAVs is possible. Our original simulations tested different communications strategies and allowed for the possibility of noise or transient loss of communications.

The simulator is written in the Java language for portability. Through a mixture of command-line parameters, GUI widgets and built-in variables, it is possible to modify nearly every aspect of the terrain, UAVs and targets.

3.1 Search mission strategies

In our prior reports [1,2] we tested a variety of strategies for collaborative search, ranging from simple random movement to globally coordinated search strategies. We found that the best performance could be obtained through the use of a *pheromone* search strategy: each UAV leaves a trace over the terrain it has visited. At the same time, each UAV monitors the terrain in its immediate surroundings, and navigates preferentially toward areas that are less covered by this synthetic pheromone. To implement the synthetic pheromone, each UAV sends information to a central controller indicating what terrain cells it has just explored. The central controller accumulates this information from all UAVs, and sends to each UAV coverage information regarding a small area centered around its location. We have shown [1,2] that this technique can work reliably with minimal bandwidth, and even when communications are unreliable. When the targets are moving, the synthetic pheromone includes an *evaporation rate* that determines how quickly the pheromone signal fades.

In addition to this "local" navigation strategy, we tested a more "global" strategy. We assume that the search space is divided into a number of large, square sectors (e.g., a 3x3 grid), and that a central controller monitors the level of coverage within each sector, as well as the number of UAVs currently in that sector or already heading there. UAVs constantly monitor the value of their own sector and compare it to the "value" of other sectors, which is a function of coverage, occupancy, and distance. If a more desirable sector is found, the UAV stops searching locally and heads to the new sector. We found that this strategy required significant hand-tuning – for instance selecting a proper distance weighting factor, or the proper threshold to trigger a move to a new sector.

In a later section we describe how we used a GA to select some of these parameters so as to maximize the success of the mission.

Finally, we tested strategies that combined local pheromone-based navigation with the global navigation strategy. In general, we found that the pheromone navigation scheme was most effective in leveraging the presence of multiple UAVs, though the global strategy could yield a slight further improvement.

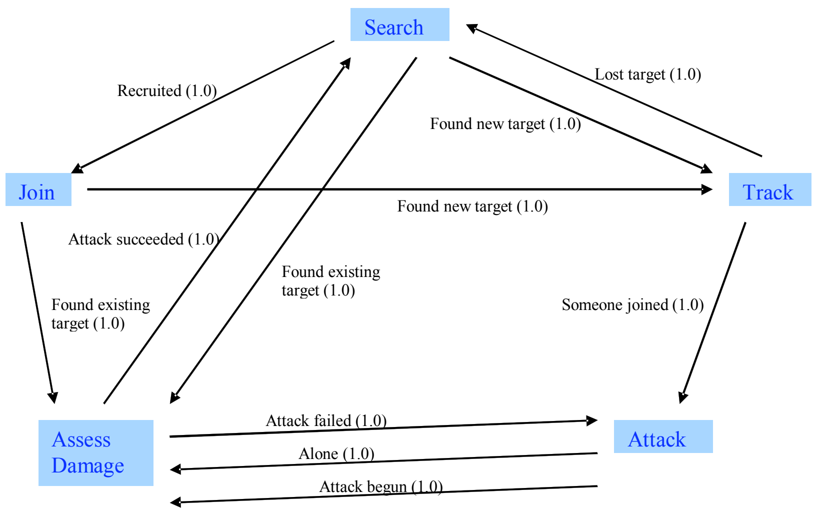

Figure 1: State transition diagram for switching between behaviors.

3.2 SEAD mission behaviors

For search missions, the UAVs operated independently: the only direct interactions consisted of avoiding one another during flight. In addition, the UAVs cooperated indirectly through the use of synthetic pheromone signals. For SEAD missions, the UAVs need to cooperate: UAVs begin by searching for targets. When a UAV finds a target, it switches to tracking mode, and it broadcasts a request for support: before attempting to strike a target, it must wait for another UAV to join it near the target, so that Battle Damage Assessment (BDA) can be performed once the UAV dives into the target. Whichever UAV first responds to a request for support will fly toward the requestor. Once the supporting UAV is sufficiently close to the target, the requestor switches to strike mode, diving into the target. The strike is successful (target hit) with some probability, usually 80% in our simulations. If the target is hit, the supporting UAV switches back to target search mode. If the strike fails, the supporting UAV now switches to active target tracking, and broadcasts a request for a new supporting UAV.

To summarize, each UAV can be in one of five behavioral states: search, track, support, attack and BDA. Transitions between behavioral states are triggered by signals received by the UAVs (Fig. 1). These are not necessarily communication signals, but rather abstractions of cues a UAV can receive from itself, its physical environment or other UAVs. As not all signals are relevant to all states, after the description of each signal is a list of states within which the signal can be received, enclosed in square brackets.

- **Found New Target**: The UAV has detected a target below that no other UAV has as its current target [Search, Support]
- **Found Existing Target**: The UAV has detected the current target of one or more other UAVs [Search, Support]
- **Lost Target**: The UAV's current target has left the target area [Track, BDA, Attack]
- **Recruited**: A communication has been received from another UAV requesting support in monitoring its current target; by "monitoring" we mean is a general term that applies to the Tracking, BDA and Attack states, in which a UAV has homed in on a specific target beneath it [all states]
- **Someone Joined**: Another UAV has joined this UAV in monitoring its current target [Track, BDA, Attack]

- **Attack Begun**: Another UAV has just begun an attack on this UAV's current target [BDA, Attack]
- **Attack Succeeded**: An attack on the UAV's current target has successfully destroyed it [BDA]
- **Attack Failed**: An attack on the current target failed to destroy it [BDA]
- **Alone**: The UAV is in the Attack state but there are no other UAVs performing damage assessment [Attack]

The behavior of each UAV depends on its state-transition rules, which conceptually represent a state-transition diagram. The default state-transition diagram is represented in Figure 1. Each node in the diagram represents a state, while the directed edges represent probabilistic transitions between states caused by the reception of a given signal.

Note that in the default case, the transition probabilities are set to 1.0, meaning that if the right conditions are met, the UAV will switch behaviors. For many of these transitions, a probability of 1.0 is sensible. For example, whenever a UAV is searching and it finds a target, it should start tracking it. However, there are cases in which a lower probability may be better. For instance, if a UAV is tracking a target waiting for another UAV to join it for BDA, it may sometimes be better to strike the target without waiting, even though there is a chance that the strike will fail. In the next section we will describe how we used a GA to evolve some of the transition probabilities.

4. EVOLUTION OF UAV CONTROL PARAMETERS

In this section we described a Genetic Algorithm (GA) hat we developed to identify combinations of parameters that would yield particularly good results. The reason for this approach is that, in most real-world scenarios, the space of possible control strategies is vast or even infinite, which makes an exhaustive search for the "best" control strategy impossible. Evolutionary algorithms such as GAs make it possible to search a vast space quickly.

The principle behind GAs and other forms of evolutionary computing is to encode certain characteristics of the model in a *chromosome*, and to allow multiple copies of the model (individuals), each with its unique set of genes, to compete on the target task during one *generation*. At the end of each generation, the performance of each individual is expressed as a *fitness* value. A *selection mechanism* is used to allow the fittest individuals to move on to subsequent generations, and to create *offspring* that inherit certain genetic traits from the successful individuals. This process makes it possible to search parameter spaces that could not be searched systematically because of their size.

Our goal in this part of the project was primarily to develop the software necessary to evolve the simulator. As an initial test, we used the GA to evolve a combination of five parameters. The performance of the model after applying the GA was only slightly superior to that obtained without the GA, but the results represent an interesting starting point for future work.

4.1 Description of the GA

Our goal in using a GA was to see if we could increase the average number of hits during the course of one mission. We selected the following parameters for manipulation by the GA:

1. The probability that a UAV in *Join mode* will abandon its recruiter when it locates a new target in order to track it
2. The probability at each time step for a UAV in *Track mode* to enter *Attack mode*, despite the absence of another UAV that could perform BDA
3. The pheromone decay rate
4. The "Good" region value – the value of a UAV's current global region above which it will not consider traveling to a new global region

5. The region value threshold – the amount the most desirable global region must be better than the current one for a UAV to decide to travel to it

Each of the parameters was normalized to have a range between 0 and 1, via a suitable encoding, and chromosomes were represented as fixed-sized, real-valued strings. Populations consisted of 20 individuals, which were evolved across a variable number of generations (see below).

Reproduction involved both *elitism*, whereby the top half of a generation was carried into the next generation automatically, and *recombination*. The latter was achieved by selecting two individuals from the population, performing one-point crossover, and mutating each gene of the offspring with a probability of 1/5 (where 5 = the number of genes). Mutation adds a number uniformly chosen between -.1 and .1 to a gene, and selection uses a combination of ranking and roulette-wheel, where an individual's chance for selection is proportional to its rank in the population, as determined by fitness.

For the fitness function, we initially thought of using simply the total number of hits obtained by an individual during one run. However, we found that the results could vary dramatically depending on the initial conditions (distribution of targets and UAV drop point): a single run with exactly the same parameters could result in as few as 0 hits and as many as 8. As a result, the GA tended to favor "lucky" individuals at the expense of individuals that may have done better on average over several runs, but happened to do poorly on a single run. To avoid this problem, we averaged the fitness of each individual over as many generations as it survived.

We also tried to use two other measures of fitness. In one case, we created a fitness function that combined the average number of hits (accumulated over multiple generations as before), the number of generations an individual had survived, and the standard deviation of the number of hits across generations for the individual (lower standard deviation being favored). The idea of this strategy was to promote longevity and consistency. The other strategy we tried was to let one generation for one individual consist of doing five runs with five different random seeds. The fitness obtained in a single generation was the average over the five runs, and this fitness was also averaged across generations as before.

For each simulation, we set the world size to 1500x1500 units, and used 10 UAVs and 10 targets. We selected these conditions on the basis of previous results, to ensure that baseline performance was neither too poor nor too good, so that the GA would have a basis for evolution, and room for improvement. We also constrained the time of each simulation to only 800 seconds (the results reported above were based mostly on runs of 1600 seconds). We allowed this relatively short simulation time for two reasons: first, we increased the pressure to find and hit targets quickly. Second, the shorter simulations allowed us to run more generations (a typical GA evolution took in the order of 24-48 hours of computing time). From our previous analyses we expected this configuration to perform somewhat poorly, with less than half of the targets being hit during a mission.

4.2 Summary or results

The results obtained from the GA were not as impressive as we had hoped: in all cases, after an initial rise, the average number of hits leveled off quickly to somewhere around 5 (i.e., 50% of the targets were hit), and then continued to oscillate around this value *ad infinitum*. Figure 2 shows the evolution of the average number of hits across generations under the three different fitness measures described above.

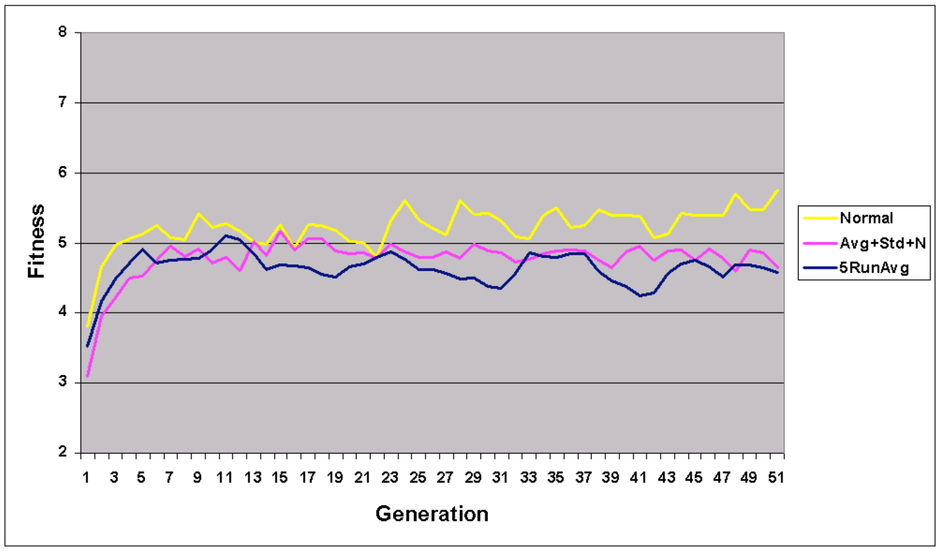

Figure 2: Changes in fitness for three different fitness measures.

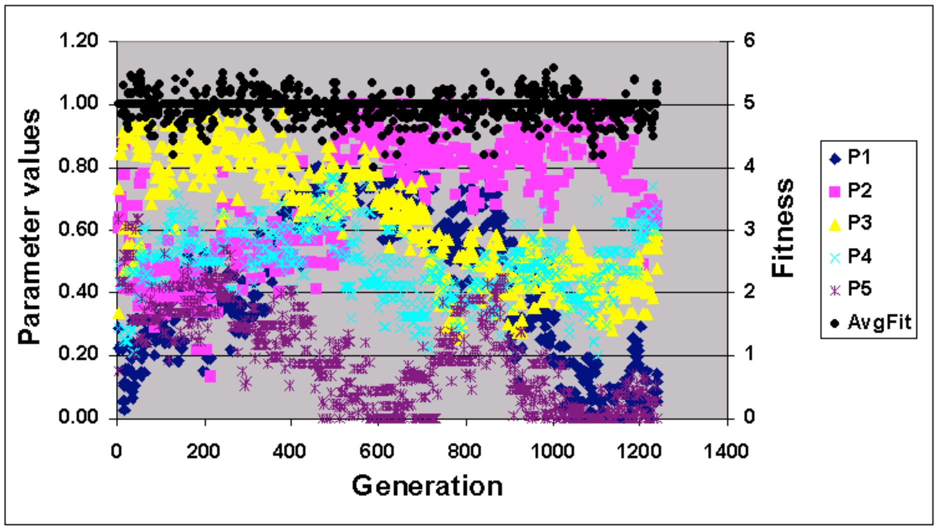

Figure 3: Evolution of the five parameters and baseline fitness function.

The first thing to note, is that in all three cases the fitness rises rapidly and reaches a plateau after only a handful of generations. Figure 2 only shows about 50 generations, but we have verified that this trend continued for as many as 1200 generations. It is possible that, had we let the GA run for several days or weeks, the fitness might have eventually risen further, though we think it is unlikely. Instead, after analyzing the results more closely, we believe that the reason is different: by monitoring the five parameters that were being modified during evolution, we found that most parameters moved quickly away from some ranges that yielded poor performance, but otherwise could be changed over significant ranges without adversely affecting the results. Figure 3 shows the evolution of the five parameters over approximately 1200 generations with the original fitness function (simple averaging across generations), along with the average number of hits. It is clear from the figure that some parameters are bounded within specific ranges, but otherwise change significantly, even though the overall fitness remains approximately constant.

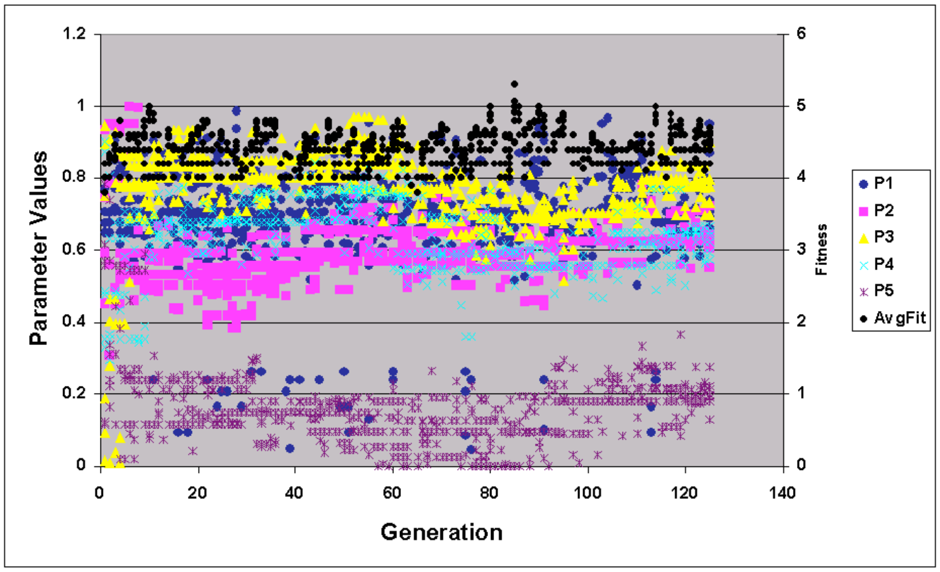

Figure 4: Evolution of the five parameters and fitness when each generation consists of five runs.

We can compare the parameter values evolved when the fitness for each generation is averaged over five runs: the results are shown in Figure 4. It is interesting that in this case the parameters converge to their final value very quickly, and do not show the same type of oscillation that was seen with the standard fitness function. This, undoubtedly, is due to the reduced noise resulting from averaging each fitness measure across five runs, which is equivalent to smoothing out the fitness landscape.

Par	Baseline	Avg+Std+N	5RunAvg
P1	0.0-0.8	0.4-0.7	0.6-0.9
P2	0.2-0.9	0.4-0.6	0.5-0.7
P3	0.3-0.9	0.8-0.9, 0.2	0.6-0.9
P4	0.3-0.7	0.2-0.3, 0.6	0.5-0.8
P5	0.0-0.5	0.1-0.3	0.0-0.3

Table 1: Parameter ranges under two different evolutionary conditions.

Table 1 shows the parameter ranges obtained with the different evolution strategies. Parameters 3 and 4 in the second case showed a slight bimodal distribution (not shown in figures). In general, there seems to be fairly good agreement across all runs as to the "good" ranges for each parameter. However, the overall average number of hits in the second case was slightly worse than in the other two cases, and we also noticed that by including the longevity and standard deviation explicitly in the fitness function, the GA tended to latch on to a few good individuals from the outset of the experiment, and keep them in the population for as long as we let the system evolve (over 500 generations). This sort of statistical bias is probably undesirable.

We can draw some specific conclusions about these parameter ranges. First off, the fact that both probability parameters (P1, P2) are non-zero suggest that the model does well to avoid fully deterministic rules of the type we embedded in the UAV state transition matrix. Intuitively, this should not be surprising: if UAVs always have to wait for someone to join them before trying a strike, they will miss some opportunities. By comparison, since each strike has an 80% probability of success, it is obviously worth taking the chance on occasion to strike without BDA. Likewise, if a UAV is flying to join another UAV that has spotted a target, and during its flight it sees a new target, it is probably a good idea to occasionally abort the ongoing mission and track the new target. It is interesting to speculate that this behavior should more useful if the first probability is nonzero, because the recruiter will eventually stop waiting for support and attack the target anyhow. Second, the behavior of P4 and P5 is significant because these parameters jointly control the balance between local and global searching. If the global search had been deleterious, we would have expected extreme values, especially for P5, which should have approached 1.0 to prevent comparison with other regions.

5. CONCLUSIONS

The results shown here, and additional analyses we carried out, have given us some useful insights into using evolutionary algorithms to identify control parameters.

Perhaps the biggest problem we found is that the variability resulting from changes in initial conditions can make it extremely difficult for the GA to find a good solution. When we tried to promote individuals with higher longevity and lower standard deviation across runs, we introduced an unwanted bias that actually caused overall performance to decrease[1]. It is possible that we could adjust the relative weighting of longevity and standard deviation relative to the average kills, but ultimately if we have to be that careful in designing a fitness function, the validity of the entire approach could be called into question.

A more general conclusion, which reinforces our experience with other projects using GAs, is that the way in which the GA is set up can have a profound impact on the results. In addition to the method described here, we tried several other ways of evaluating individuals and of applying evolutionary rules. One could easily spend months "playing around" with conditions, without really gaining any insight into the problem itself.

Another, related observation is that the choice of what parameters will be evolved, as well as the way they are represented as genetic strings, can also have a profound effect on the results. We chose the five parameters above because they seemed to have the greatest potential of influencing the behavior of the UAVs.

Finally, it is also clear that the mission specification is crucial to overall success of the GA. In this case we emphasized number of hits in a limited-time mission. It is likely that a more tightly defined mission type would be more amenable to parameter search through evolution.

The results reported here were preliminary in nature. However, they show the potential application of evolutionary computing to the design of control strategies.

6. REFERENCES

[1] Gaudiano, P., Shargel, B., Bonabeau, E., Clough, B. (2003a). Control of UAV swarms: what the bugs can teach us. *AIAA "Unmanned Unlimited"*. San Diego, CA.

[2] Gaudiano, P., Shargel, B., Bonabeau, E., Clough, B. (2003b). Swarm Intelligence: a New C2 Paradigm with an Application to Control of Swarms of UAVs. *8th International Command and Control Research and Technology Symposium*. Washington, DC.

[3] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing.

[4] Forrest, S. (1993). Genetic algorithms: Principles of adaptation applied to computation. *Science* **261**: 872-878.

[1] It is worth pointing out that in this experiment the fitness as we defined it actually did increase gradually over generations. However, the average number of hits decreased slightly. Hence we were getting solutions that sacrificed performance on individual runs in favor of more stable performance in the long run.