# Autonomous Agent Behavior Generation Using Multiobjective Evolutionary Optimization

Dustin J. Nowak and Gary B. Lamont
Dept. of Electrical and Computer Engineering, Graduate School of Engineering and Management,
Air Force Institute of Technology
WPAFB, Dayton, Ohio, USA
dustin.nowak@afit.edu, gary.lamont@afit.edu

## ABSTRACT

An agent system that develops and evolves its own structure can facilitate more accurate responses to complex environments. The purpose of the paper then is to explore this idea built upon our unmanned aerial vehicle (UAV) swarm model and simulation that uses autonomous self-organized concepts. The specific objective is to re-engineer this UAV foundation based upon a formal design model with focus on bio-inspired agent attack through emergent control structures. The overall design approach should give UAVs or generic agents the ability to not only react to dynamic environments but develop the controls in order to change behaviors spontaneously. To allow these behaviors to properly evolve, a multi-objective evolutionary algorithm generates a self-organized rule-based agent swarm. Heterogeneous UAV swarms are tested against difficult targeting scenarios that evolve specific attack behaviorial techniques. Statistical observations indicate that bio-inspired techniques integrated with the emerging entangled (non-hierarchical) framework provide desired complex UAV swarming behaviors in dynamic environments.

## Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Miscellaneous

## General Terms

Algorithms

## Keywords

Self-Organization, Agents, Autonomous, Swarm Intelligence

## 1. INTRODUCTION

Embedding desired behaviors in autonomous vehicles or agents is a difficult problem at best and in general probably impossible to completely resolve in complex dynamic environments. Nevertheless, the future deployment of autonomous vehicles or agents depends on large-scale decentralized swarming environments with associated behaviors. Examples include homogeneous and heterogeneous autonomous robotics and unmanned aerial vehicles (UAVs) for reconnaissance and possible action. Biological inspired self-organized[1] systems as found in forging insects (ants, bees, ...), flocking birds and attack activities (bees, wasps, ...), revolve around control approaches that have simple rule sets that generate desired emergent behaviors. In employing this approach, localized agent rules that evolve the requisite swarming behaviors are desired. To computationally develop such a system, an underlying organizational structure or framework is required to control agent rule execution. Observe that existing efforts focus on fixed discrete control structures that limit the viability in dynamic real-world environments.

Our current framework model for unmanned aerial vehicle (UAV) swarm model and simulation uses self-changing (self-organized) rules sets and a priori forced control structures [**?**]. The new approach is to re-engineer this system in order to better identify and coalesce autonomous self-organization (SO) behavioral features and to use better software engineering principles. The desired dynamic vehicle or agent swarm behavior is evolved using a multi-objective genetic algorithm which successfully generates agents that perform dynamic reconnaissance and as appropriate attack en masse targets. This provides the swarm with the evolved autonomous ability to dynamically react to hostile environments with low computational complexity and high effectiveness. A self-organizing multi-objective evolutionary algorithmic approach dynamically determines the proper individual rule weighting and control parameters without requiring parameter tuning, yet provides highly dynamic swarming behavior. Also, the new control structure evolves into an *emergent entangled-hierarchical* framework instead of a less effective a priori strict hierarchical structure.

This paper initially summaries the background of self-organized agent research in Section 2. The specific vehicle or agent swarm problem domain is developed in Section 3. The formal system design model as presented in Section 4 provides a non-ambiguous mathematical model for design. Because of the NP-Complete complexity of this problem domain, we proposed a self-organized multi-objective evolutionary algorithm (MOEA) to explore the associated search space. The concept and design of the self-organized MOEA

[1]Self-organization is a process where the system structure spontaneously changes without being externally controlled


*GECCO'08,* July 12–16, 2008, Atlanta, Georgia, USA.

is developed in Section 5. The system is extensively evaluated in Section 6 with swarms of heterogeneous vehicles in a simulation system with animated graphics. Section 7 concludes the discussion focusing on the relatively high measures of success and future work..

## 2. GENERIC SWARMING APPROACHES

Approaches to control and swarming are quite varied. Some of the exemplar works are discussed along with our extensions. The first part of this section presents a brief background on generic agent swarming problem domain. Then several associated efforts with other solution techniques are discussed and briefly evaluated.

### 2.1 Algorithmic Structures/Frameworks

Our problem domain has a resemblance to Nikovski's [**?**] domain of Decision-Theoretic Navigation of mobile robots. Nikovski's work does not specify the particulars about the learning and domain encountered, but autonomous agent movement and navigation fundamentals are universal. Nikovski finds a Best First Model Merging (BFMM) in which the objective is to accurately predict which actual time-state pair the current Partially Observable Markovian Decision Process (POMDP) observed state is modeling. He also endeavors a State Merging (SMTC) searches for the same objective but includes suboptimal solutions that can be merged in order to solve for an aggregate better solution in the long run. None of the combinations converge with the optimum but the systems do show improvement. The cause of this stems from the systems inability to learn the correct model. There also exist a bit of concern about mapping vectors leading into and out of a state given the Markov assumption. These attempts are all geared toward finding predictable solution in an extremely large solution space.

Khosla [**?**] uses evolutionary algorithms for Weapon Allocation and Scheduling (WAS). In this domain two parameters are optimized, Threat Kill Maximization (TKM) and Asset Survival Maximization (ASM) reflecting. Deterministic and stochastic (GA) approaches are employed with interesting results. A technique of utilizing a single objective genetic algorithm (GA) successfully accomplished dynamic environment reaction in [?]. There are several approaches to developing SO vehicle swarm controllers using genetic programming (GP) [**?**]. Woolley's [**?**] defines a control structure called the Unified Behavior Framework (UBF) which constrains the GP, with consistent success in a constrained problem.

Overall, these research efforts reflect the intractable complexity of the problem domain space and generally heavy computational demands. Many of the mentioned techniques simply constrain or approximate the solution space and then search stochastically. Thus, it is desired to find simple behaviors sets and relations between them to allow lighter computational processing.

### 2.2 Rule Based Behavior Archetypes

In addressing this large and dynamic UAV swarm problem domain, self-organization (SO) is employed in the development of a computational system called Swarmfare [?]. Swarmfare is our initial animated simulation of UAV swarms using formation control and individual rules sets that generate desired swarm behaviors. These rules sets are developed in such a way that all solutions are feasible in sub-domains. In the case of Swarmfare, the system gathers these rules together to form behavioral archetypes (BAs). Through these groupings the rules are weighted and applied to establish each subsequent action. This is similar to the modes in Rosenblatt's architecture [**?**]. Note that Swarmfare initially used a set of neural-network perceptrons to form the control agent and find the appropriate BA to use at a precise moment; but response to new unknown environments was limited. The open scheme of BAs allows for more flexibility and quick response with different variations given the dynamic environment.

*SO Rules Utilized* The Swarmfare simulation system currently combines 10 rules to define each BA:

- Flat Align - vector align with neighbors
- Separation - Cluster Range away
- Cohesion - Cluster range towards
- Obstacle Avoidance
- Evade - a priori collision detection and avoidance
- Target Orbit - orbit target at safe distance
- Attract - towards center of mass of all targets
- Weighted Attract - towards closest target
- Target Repel - repel if with 90% of UAV sensor range
- Weighted Target Repel - repulsion based on proximity

The ten rules are derived from generic swarm and target interactions. The core five swarm rules, flat align, separation, cohesion, obstacle avoidance and evade are from Reynolds work [**?**, **?**]. The orbit stems from Lua's work [**?**]. The target driven rules, attract (and weighted), repel (and weighted), and orbit are derived. Attract and repel tries to form a balance of aggression and respect towards targets. Each rule is weighted differently depending on the makeup defined by BAs of the agent.

## 3. SPECIFIC SWARM PROBLEM DOMAIN

A swarm of vehicles or agents moves through a space governed by basic physical and communication principles. The space contains a set of obstacles and targets (physical or abstractions). The targets are stationary but attack. The sensor information and simple self-organization (SO) rules are given to each individual vehicle or agent. The rule set and interaction amongst the vehicles creates emergent behaviors that allow the agents to swarm and attack the targets nondeterministically. Our Swarmfare simulation includes a UAV swarm animated simulation package that presents the emergent behaviors for visual analysis.

The objectives of UAV swarms are two fold. First the system must establish a Swarm formation. The intent of this objective is to increase search effectiveness, safety, and attack force. The second objective requires the swarm to provide reconnaissance and successfully engage targets. The goal is to maximized damage to the target and minimized causalities in the swarm. All of this must occur while moving through a hostile terrain environment.

The correct control of the vehicles in this space results from exacting combinations of the SO behavior set. The arbitration of the behavior sets and weighings of those behaviors form the real solution space. Search through that extremely large and volatile space requires advanced searching techniques.

*Modeling Constraints* Many intersecting constraints exist on the agents traversing a *real-world* domain. They include:

- Physical dynamics of the vehicles/agents (UAVs, robots)
- Physics constraints of not only the craft but munitions
- Agent sensor's range constraints and uncertainty
- Comm bandwidth constraints and unreliability
- Geographic environment incursion on movement, sensors, and communications
- Fog of battle (human or sensor blurring of reality)
- Friction of battle (movement in the wrong direction)

All of these real-world constraints if hidden create a scenario where relying on incomplete state details can cause unknown incompatibility problems. As a result the system and simulation can only function with a restricted amount of validity compared to the real world. However in SO decomposition, the systems are biologically inspired. This creates juxtaposition between the human need to thoroughly develop a hierarchical state model that represents exemplar biological agents. With SO, we tend to steer away from exact state modeling and allow the system to abstract the states to the level needed for survival in the given domain.

# 4. FORMAL SYSTEM MODEL MAPPING

Mapping the meta-level UAV problem domain to a formal model provides a high-level to low-level design process in order to direct the unambiguous design of the autonomous agent operators. The resulting design efforts of the high-level model are then employed in an informal engineering process that developments the low-level design and implements the UAV operators. This low-level model also includes the innovative integration of an evolutionary algorithm to evolve a dynamic self-organized system with the desired emergent behavior.

In this section we first discuss the formal high-level aspect of the design process which requires explanation of several critical elements. The targets and vehicles or agents have sensors and interact through basic nearest neighbor communications, with epidemic transfer of information. As a result, any agent in the space has only limited knowledge of its circumstances. As the agents move through the space, engagement with the targets follows stochastic modeling and allows for aggregation of forces. This creates a scenario where the domain space changes rapidly and unpredictably. To reduce the confusion and simplify the space, Markov assumptions are used. Without global knowledge and the ability to predict the results of any single action, these assumptions indicate a way to abstract the state.

For these reasons the problem domain falls under the category of Partially Observable Markovian Decision Processes (POMDP). A POMDP is made of the tuple shown in equation **??**, which includes the state set (S), action set (A), transaction set between states (T), observations (O) and feedback mechanism (R).

$$D(S, A, T, O, R) \tag{1}$$

The expansive problem domain space of POMDPs forces the reduction of the state into more abstract pseudo states for the ease of understanding. Thus, we have extended the mapping of this specific swarming problem (target engagement) to a POMDP model [**?**]. The complexity is defined by the size of the swarm of agents or vehicles and targets and the domain. The complexity of the global problem is loosely based on the number of agents, $n$, and the action, $m^2$ (movement in the map), and transaction possibilities, $t_p$, resulting in $O(n^{t_p m^2})$; a NPC problem.

Given that *no scenario* encountered is the same, it is impossible to articulate the search space entirety or control search based on the *immense* state space created by the POMDP model. This phenomenon forces the need for a probabilistic behavior model. Using self-organized behaviors the system can abstract the state and respond to it appropriately in polynomial time. This abstraction forces sets of abstract states that dictate modes and behavior structures. As a result the system needs discrete sets of behaviors with different control weights that allow the flexibility to move in this abstracted state which is presented in Section **??**.

*Interactive Partially Observable Markov Decision Process* An extension of POMDP models focuses on the application of the Markov assumptions to independent agents with independent actions. There are three approaches to this subset of the model. The first defined by Bernstein [**?**], is the Decentralized POMDP. It specifically articulates actions and Observation sets for each individual agent. The second by Boutilier [**?**], called the Multiagent MDP, also specifically articulates the the set of actions for all agents. The Interactive POMDP Interactive Partially Observable Markov Decision Process (I-POMDP) used by Doshi [**?**] specifically defines the state transitions of each agent based on the probability of the interactions with other agents. We selected the Doshi model. His approach focuses the agent actions based on its knowledge base and behavior set independent of the entirety of the domain.

Equation **??** shows the tuple defined by I-POMDP [?]. This approach focuses on decoupling agents acting in the same environment by adding belief of the effects of interaction to the state.

$$I - POMDP_i = \langle IS_i, A, T_i, \Omega_i, R_i \rangle \tag{2}$$

$IS_i$ defines the interactive effect of the agents on each others state through $IS_i = S \times \Theta_j$. The belief state of other agents $\Theta_j$ derives from Equation **??**.

$$\Theta_j = \langle b_j, A, \Omega_j, T_j, O_j, R_j, OC_j \rangle \tag{3}$$

The elements of this belief state derive from the POMDP but focus on the agent $j$. The $OC_j$ outlines the optimum criterion for the agents. This representation mirrors the actuality of the simulation and the stochastic nature of interaction and transitions. Through the POMDP the totality of the domain is represented and the I-POMDP instantiates the domain model of the individual agent and ties the two formal models together.

From the problem domain mathematical model (I-POMDP) stem system decomposition. Figure **??** presents the overall approach with an innovative 'U'-decomposition. Engineering a system starts with the domain understanding provided by the math model. The problem model decomposes into simple implementable rule sets. From those rule sets control structures relating the rules are formed. With that structure in place the behaviors emerge and the system receives feedback at the functional level and returns it to the rule set [?]. Thus in the development process, the abstract formal structure is used to drive the detailed lower-level design and implementation issues.

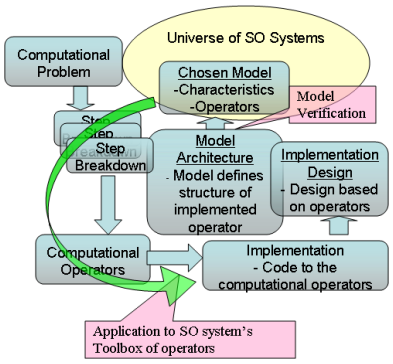


**Figure 1: 'U'-Decomposition Technique for Developing Self Organized Systems [?]**

## 5. SWARM BEHAVIORAL CONTROL

The evolutionary algorithm (a genetic algorithm) is discussed then the decomposition of the new rule sets and resulting control structure are outlined. Figure **??** shows the data flow, where a genetic algorithm (GA) attempts to optimizes control and rule weights. The controller arbitrates between behavior sets (BAs) and the movement vector results for the specific BA rules and their associated weights. In order to find the proper control set for the vehicle or agent swarm behavioral rules, weights must be applied. Thus, a search technique must be chosen to attempt to "optimize" these weighings using a stochastic search technique. The swarming problem and algorithm domain is expanded into the multi-objective problem (MOP) realm and thus, employing a multi-objective genetic algorithm (MOEA). Although optimal solutions are desired, a stochastic algorithm such as an evolutionary algorithm of course can not guarantee that optimal solutions are found.

### 5.1 GA Chromosome & Operations

Formulation of the chromosome data structure derives from the objective functions. In order to reach the objective of target engagement, the system must produce emergent behaviors that aggregate the capabilities of the UAV or agent in a swarm. The EA must control and integrate the SO rule sets to form this emergent behavior. Therefore, the mapping of the chromosomes relates to the control parameters or weighting of the rule sets.

*Control Development* Implementing the transactions defined by the behavior set in Section **??** requires synergistic integration of those behaviors. Behaviors are vector fields that direct the agent's movement, similar to that seen in [**?**].

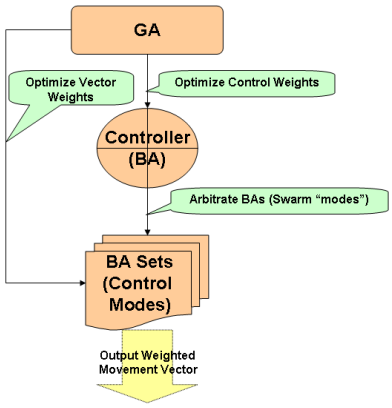


**Figure 2: Data flow in the high-level system [?].**

Arbitration in the benchmark work uses a multi-layer perceptron to chose between SO behaviors sets. The control weighings for this structure are also include with the behaviors sets weights in the chromosome shown in Figure **??**. Each control weighing and corresponding behavior weight set forms an adjacent sub-string in the chromosomes.

Given the multiplicity of states in an environment, multiple sets of rules or modes direct the swarm, shown in [?]. Therefore a control structure must be used to change the mode. In this simulation environment a network of preceptrons senses the current environment and does the arbitration between modes (BAs). These BAs allow the system to dynamically develop several sets of weighings in order to react to different situations. For example a chromosome with three BAs and 9 rules would have 42 alleles. Several of these sets form the full structure of the chromosome.

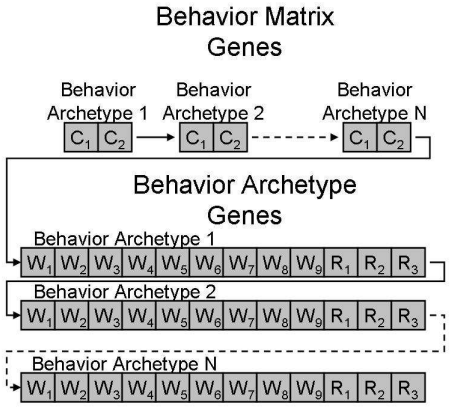


**Figure 3: There is a connection weight for each sense for each behavior archetype. These are followed by 12 genes which describe the weights and radii for the behavior rules for each behavior archetype.**

Figure **??** shows the representation used. The chromosome values are used to map the weighings of each rule. Note that the evolutionary operators work on the bit level, so in order to translate the chromosome values Gray coding must be exploited. With Gray code the system minimizes the effects of those operators, because the change of a single

bit (genotype information) only changes the value of that gene (phenotype information) step as well.

*Fitness Function Description* In order to define the fitness function, we must first analyze the objective functions. Attack UAV swarms focus on destroying targets. A shortfall of the previous version of Swarmfare was agent casualties due to collision with other agents and obstacles were limited. Thus, in order to increase the effectiveness of attack, the second objective of casualties was added to the original damage objective. Equation **??** defines the values based on successful engagement.

$$D_t = \tau_{destroyed} * 100 + \tau_{destruction} * 10 \quad (4)$$

The second fitness function is the casualty rate, defined in Equation **??**:

$$C_t = \nu_{damage} * 10 \quad (5)$$

Here the damage received is multiplied by ten to keep it in scalar concert with the damage inflicted. Complete destruction of an agent results in a score of 100.

*Generate Population* Initialization uses a bit wise generation of each individual.

*Evaluation of fitness* Evaluation is accomplished through simulation, which returns an average damage and casualty score over a predetermined number of runs.

*Crossover and Mutation Operators* Particular to this GA implementation are modified operators. In both instances an entire BA gets modified. The complex form of the chromosome is particular to the given problem domain. This forces the evolutionary operators to specifically address the points at which changes are made. In mutation changes in alleles happens in a single (BA) with both the control section and behavior section of the chromosome. In this implementation each part of each BA mutates.

*Selection for next Generation* The algorithm uses Elitist for generational selection. With the space as diverse as it is and the population and reproduction operators facilitating high levels of exploration, elitist approach allows the algorithm to exploit the good genes. Allowing both the "best" parents and children to continue to the next generation.

*Classical MOEA Approach* The Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) MOEA is chosen because of its specific operators and structure of available software. NSGA-II is used because of its fast-nondominated sort which possesses a balance between exploitation and exploration properties. [**?**] shows increasingly effective solutions while maintaining a wider range, as compared to other MOEAs.

Because we desire that the NSGA-II minimizes the functions, there had to be a slight modification in the objective functions of Section **??**. The system counts the damage in negative points and the number of survival has been turned into the number that is dead. This way both functions have "better" lower values.

## 5.2 Self-Organized UAVs/Agents

In the natural world many systems develop through self-organization (SO); emergent properties evolve as a result of localized agent interaction sans global knowledge. [**?**, **?**, **?**] This is the foundational impetus for a new self-organized genetic algorithm.

There are three benefits using SO decomposition in computational problems: ease of implementation, lowered computations, and dynamic adaptation. *Using SO implies finding some set of behaviors from which a desirable structure emerges.* If done properly each agent's behaviors can be coded as simple rule sets (DNA view). Lower computational cost stems all computations executing localized rule sets at the agent level. Interaction and communications should also be very localized. Finally dynamic response happens as an emergent behavior. The behaviors do not restrict the system to a script (a hierarchical a priori structure), but instead the agents are capable of self-organized quick response to unpredictable stimulus (environmental awareness) through those rules.

What we desire is that the operations of a self-organized genetic algorithm reflect the agent response in a dynamic nature enabling more versatility and universality. To do this we remove some of the restrictions, problem specific constraint, and niche operators, and parameter tuning commonly used. [?] We attempt to finally bag that white rabbit by allowing the population to tell the algorithm what it needs. With higher population commonality the algorithm senses building blocks/genes with successful values and allows them to thrive while still varying aspects with lower known probabilities. When the problem space is first being explored or in a state of high exploration the algorithm response by continued exploration. To do this we modify the highly varied world of recombination operators and add an operator to sense the current genotype space acting upon that information. Recombination has been studied extensively spawning many different approaches to satisfy different problem constraints [?].

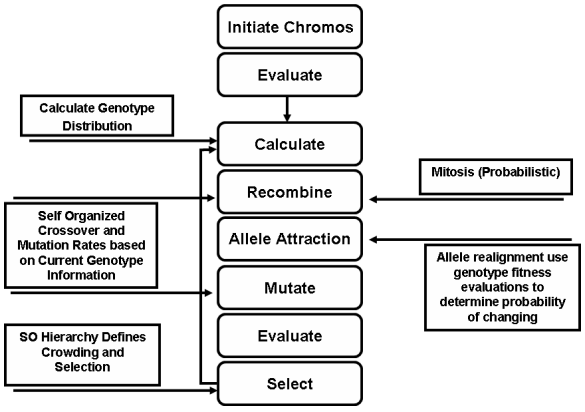


**Figure 4: Self Organized Gene and Allele Genetic Algorithm, the SOGA**

The new Self Organized Gene and Allele Genetic Algorithm, *SOGA*, in Figure **??** shows the added operators and modified genetic algorithm flow. The calculate step determines the Genotype Distribution Collection (GDC) in the population to sense the entropy levels. The levels of entropy define the role of exploitation versus exploration for the rest of the algorithm. Information from the GDC facilitates evolution rate updates. The recombination step is also modified to include knowledge gathered from the GDC. The CAA operator also utilizes this information to interpolate the attraction of each allele in the chromosome. And the selection operator utilizes a new crowding distance operator based on SO hierarchical approach.

*Genetic Distribution Collection* It simply utilizes a data

structure, Γ, the same length of the chromosome to store the highest likely value for an allele (bit) and the normalized histogram weight of that value (double). This histogram gives a reading to the system of overall entropy and also the location of unstable alleles.

*SOGA Crossover and Mutation* First application of the insight gained from the GDC allows the formation of the mutation and crossover rates. By watching the changes in the GDC from the last generation the system can determine the amount of entropy in the population. With lower entropy the system continues to maintain high levels of crossover and mutation. Of course with the higher entropy the system has moved into the exploitation phase of the algorithm and does not require as much variance in the chromosomes. Equations **??** and **??** shows the updating function for the rates of mutation and crossover.

$$c = \frac{\sum_{i=0}^{chromo_{len}} (\Gamma_i(t-1) - \Gamma_i(t))}{\Gamma_{size}} \tag{6}$$

$$m = \frac{\sum_{i=0}^{chromo_{len}} (\Gamma_i(t-1) - \Gamma_i(t))}{\Gamma_{size}} * min((\Gamma_i(t-1) - \Gamma_i(t)! = 0) \tag{7}$$

*Mitosis* The second operator change comes in the recombination step. The system recognizes when good building blocks exist and attempts to perpetuate their existence. The system analyzes the Γ level to determine the strength of each allele in the crossover section. From that the system probabilistically chooses between mitosis, which facilitates exploitation, and meiosis, which enables exploration, recombination based on a SO threshold $\varpi$. If the normalized summation of the alleles in the crossover section is above the threshold it chooses mitosis on the higher gene based on that probability.

*Correcting Allele Attraction* The CAA utilizes that same GDC information and exploits it to establish linkages between pairs or subset of disjoint alleles. As in the modified crossover, this operator allows the system to focus on high exploitation when the population is diverse and solution likely unknown while exploiting known sets, building blocks, or linkages. Here Γ analyzes every allele, and does a replacement based upon probability from equation CAA.

$$P_{change}(x|\gamma) = \Gamma_{w_i} * (W_c - W_n) \tag{8}$$

Here both $W_c$ and $W_n$ are the normalized summation of correct and incorrect mappings, respectively, between $\Gamma_v$ and $\vec{a}_t$.

*SOGA Selection Operator* Selection in the natural world stems from environmental pressures. In order to continue place pressure on the population this algorithm uses a form of elitism. SOGA utilizes the same *fast-nondominated sort* as NSGA-II. However, the crowding operator uses a self-organized ranking structure. First the neighborhood gets defined dynamically by the size of the current population space in all directions of every objective. Then the individuals that qualify as neighbors use a SO ranking structure. The remaining positions in the child population are filled based on the ranking structure. This gives a distributed set of the less fit individuals in a rank. When the higher ranks become more crowded the algorithm pushes the individuals towards the less explored reaches of the front. Equation

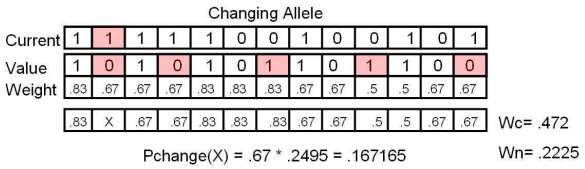


**Figure 5: Changing the allele is highlighted on the top chromosome through the weighing of the GDC shown below with the best known value and percent certainty below it. The result of the GDC for correct predictors ($W_c$) and non-correct predictors ($W_n$) is normalized and added. The probability of change is the product of incorrect prediction of the given allele and that difference.**

**??** shows the formula for determining the probability of an individual winning a hierarchical engagement.

$$Prob_{i_{win}} = \frac{1}{1 + \exp(f_{sup} + (rank_i - rank_j))} \tag{9}$$

Here $f_{sup}$ represents the number of fitness functions $i$ dominates $j$ and $rank_i$ and $rank_j$ represents the current rank of the individual. The ranks are all initialized to 1.

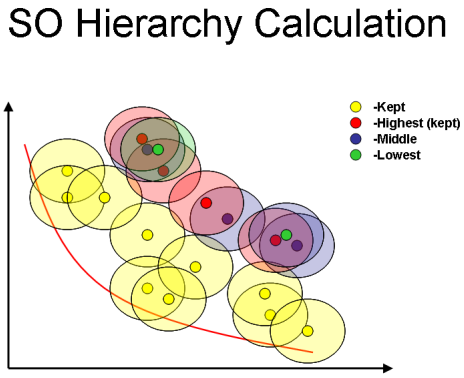


**Figure 6: SO selection operator example. The inner circles are the individuals in the population. The outer circles represent the neighborhood. The Front row individuals are kept because of their rank, the others (moving from top left to bottom right in the back row #: 1, 4, 5, and 7) individuals have the highest levels in the SO hierarchy and are also kept.**

With the selection operator applying pressure, the crossover and mutation exploring, and the CAA acting as a self correcting gyroscope the system finds a balance in exploitation and exploration. With the inclusion of both parent and child in the selection population, the algorithm allows good *building blocks - genes* to be carried not only by the new genetic material but with the old chromosomes as well.

## 5.3 Advanced SO Swarm Control

This effort also focused on extending the combination of the basic SO swarm controls outlined by Reynolds [?]. In order to do this the problem domain must be decomposed into sections that are implementable as low level rules which spawn a desired emergent behavior. After simple swarm

formation and reconnaissance capabilities were established in [?], the next logical step is transitioning to a target area and optimized target engagement. Two behaviors are developed to accomplish those: Migration and Bee-Inspired Attack. These more advanced behaviors required a more dynamic controller, the DE-Inspired Controller.

*Migration* The goal with migration is to develop the ability of the swarm to move fluidly and non-deterministically through a set of waypoints. Migration happens at an individual level, feeding into the movement a vector headed towards the closet waypoint. As a result the swarm moves together without separating, through a rough environment to achieve given waypoints.

*Advanced Attack* Attacking"en masse" is only marginally effective in [?]. The major failure was the agents found themselves in situations where targets sets over powered them. For this reason target engagement is decomposed into three phases: target area reconnaissance, target deliberation, and threshold based attack. In the first step each individual agent approaches the target area collecting the local state of the environment. Then the agents do individualized selection of the most opportune target. Finally, the swarm announces their choices and if the agents choice has enough votes the attack ensues. The reconnaissance and threshold voting come from Bee Hive selection as investigated by Visscher [**?**]. Figure **??** shows the threshold decision process.

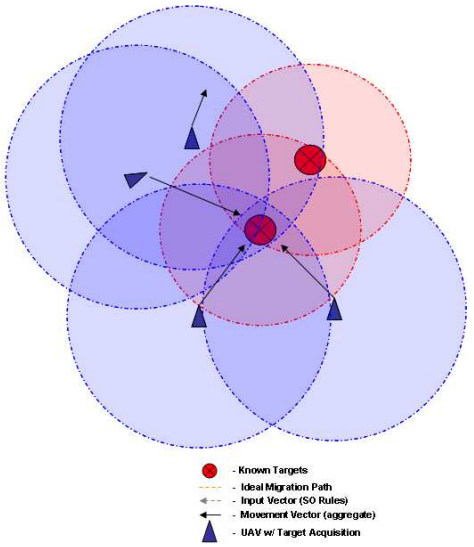


**Figure 7: Attack is based on the number of agents in the swarm, the location of the targets and the number agents ready to engage.**

*Next Generation Controller* In order to arbitrate over the BAs in dynamic environments the DE-inspired controller is developed [?]. This arbiter worked on the sensor defined abstract domain space in order to choose the best BA. The DE controller works with foci and governs a section of the domain space through Equation **??**, where $I$ defines the foci vector and $BAw$ the weighings for that BA.

$$F = \sqrt{\sum \frac{I_k^2 - BA_k^2}{BAw_k}} \tag{10}$$

Given that the controller has the same data structure as used by Differential Evolution, the DE mutation operator is added to the GA. This produces a situation where each BA has a local area of influence on the abstracted state space. Thus, the "standard" GA has been extended to the SOGA which evolves a self-organized agent system.

# 6. DESIGN OF EXPERIMENTS & RESULTS

A set of experiments is defined with several data collection techniques to show effectiveness and efficiency of the various approaches in a UAV problem domain. The objective of these tests is to compare the newly implemented aspects to previous results. Specific experimental objectives are discussed within each set of tests along with statistical evaluation measures.

The system tests against the bench marks outlined in [?]. For this population of 60 it runs 60 generations, 5 times with 30 simulations a piece. Every 10 generations it reaches an epoch. At epochs the system loads sequentially more difficult scenarios. Scenarios consist of a set of 20 UAVs, with a group of targets inside a discrete domain 800 by 800. The *predator* UAV flight dynamics are used to model the UAVs in the 2D environment. Engagement happens probabilistically with relative strengths of the targets and UAVs defined. In each continuing epoch the scenario has more targets with large engagement and sensor rings. This allows the swarm time to develop the simple flocking before creating extremely dangerous situations for one individual agent. Once the basic swarming behavior are evolved through this first series of training, another set of attack specific scenarios are run. This second set of tests gives the new controller and advanced attack behavior a chance to "flex" their muscle.

During the tests two sets of data are gathered. First the system gives the mean and best scores for each generation. An analysis of this shows the SOGA's ability to generate increasingly better solutions over time. Second, the system outputs the populations for analysis of the Pareto fronts. Analysis of the first data set indicates any difference through a Kruskal-Wallis test. Analysis on the second data set uses the hypervolume and $\epsilon$-indicators on the final Pareto Front because of the unknown $\mathcal{PF}_{true}$.

NSGA-II is tested through the original configuration described in Section **??**. Several iterative steps developed the each aspect of the system, and at every stage improvement is shown. Finally the system is run in the configuration described in Section **??**. Final testing includes a run of the original 6 scenarios and then 8 more scenarios to specifically design to exploit and compare the new control and attack configurations.

Table **??** shows the statistical analysis of the fronts and populations. The p-values leave no room for doubt in the independence of populations! The hypervolume increase is approximately 70% as well. The final front also dominates the previous two fronts with an error of over 100 points. The 54.6 error indicator in the final front represents the previous tests ability to find a solution that accomplishes nothing without causalities.

There is a clear statistical difference in the system after the addition of the new behaviors and controllers. The controller provides more flexibility and more accurate transition based on the state space. The added behaviors of migration and bee-inspired attack give the system more effective target engagement. Thus, the statistical improvement in the desired emergent agent behavior is achieved.

| Type | NSGA-II | SOGA | Advanced Control & Behavior |
|---|---|---|---|
| P-value (obj 1) | 0.0002 | 0.0002 | - |
| P-value (obj 2) | 0.0002 | 0.0003 | - |
| Hypervolume | 31301.71 | 34132.44 | 55061.76 |
| Epsilon | 130.6 | 170 | 54.6 (both) |

**Table 1: Statistical Comparison of Original and Advanced Setups**

# 7. CONCLUSION

The goal of the investigation is to establish an autonomous self-organized multi-vehicle or multi-agent swarm system with emergent behavioral control structures that could perform missions in complex dynamic environments. The initial aspect was to utilized a formal structure to model the agent problem domain so as to develop a non-ambiguous lower level operational model. The resulting lower level structures are integrated with multi-objective evolutionary algorithms to provide individual agent rule sets that are simple but effective and generate the desired SO emergent behavior. This design process establishes an entangled hierarchical swarm control architecture with self-organization. Statistical testing reflected very satisfactory results. Future research is focusing on using the formal model to prove convergence to desired behavior characteristics as well as exploiting SO capabilities in a more generic bio-inspired realm. The intent is to eventually integrate such a autonomous system model into "small" unmanned aerial vehicles in continuing developmental efforts.


# Acknowledgment

This effort is in support of the AFIT Advanced Navigation Technology (Ant) Center. The sponsors of this research are the Air Force Research Laboratory (AFRL) Information Directorate (Dr. Robert Ewing) and the Sensors Directorate - Virtual Computing Laboratory (Mike Foster).