UAV Swarm Mission Planning Development Using Evolutionary Algorithms - Part I
SCI-195

Gary B. Lamont
Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
WPAFB (Dayton), OH 45433-7765, U.S.A.
Gary.Lamont@afit.edu

Abstract

Embedding desired behaviors in autonomous vehicles is a difficult problem at best and in general probably impossible to completely resolve in complex dynamic environments. Future technology demands the deployment of small autonomous vehicles or agents with large-scale decentralized swarming capabilities and associated behaviors. Various techniques inspired by biological self-organized systems as found in forging insects and flocking birds, revolve around control approaches that have simple localized rule sets that generate some of the desired emergent behaviors. To computationally develop such a system,an underlying organizational structure or framework is required to control agent rule execution. Thus, autonomous self-organization features are identified and coalesced into an entangled-hierarchical framework. The use of this self-organizing multi-objective evolutionary algorithmic approach dynamically determines the proper weighting and control parameters providing highly dynamic swarming behavior. The system is extensively evaluated with swarms of heterogeneous vehicles in a distributed simulation system with animated graphics. Statistical measurements and observations indicate that bio-inspired techniques integrated with an entangled self-organizing framework can provide desired dynamic swarming behaviors in complex dynamic environments.

1 Introduction

The future deployment of autonomous vehicles or agents will be in large-scale decentralized swarming environments with associated behaviors. Examples include homogeneous and heterogeneous autonomous robotics and aerial vehicles for reconnaissance and possible action. Examples include commercial UAVS such as the Raven [1], DARPA's micro-UAV project with fixed wing, flapping wing, and rotary wing efforts as found in [2], and UAVs built from MEMS micro-devices [3]. Biological inspired self-organized 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 some of the desired emergent behaviors. In generalizing this approach, localized agent rules that evolve the requisite agent and vehicle swarming behaviors are desired. To computationally develop such a system, an underlying organizational structure or framework is required to control agent rule execution. Thus, autonomous self-organization (SO) features are identified and coalesced into a SO system entangled-hierarchical framework. The dynamic vehicle or agent swarm behavior is evolved using a multi-objective genetic algorithm to successfully perform dynamic reconnaissance and as appropriate attack targets. This provides the swarm with the evolved ability to dynamically react to hostile environments with low computational complexity and high effectiveness without requiring constant human interaction. The self-organizing multi-objective evolutionary algorithmic approach dynamically determines the proper weighting and control parameters without requiring parameter tuning, yet providing highly dynamic swarming behavior.

This paper initially discusses the UAV research background in Section 2. A generic vehicle or agent swarm multi-objective problem domain model is developed in Section 3. Because of the NP-Complete problem complexity, in Section 4, the proposed system uses a multi-objective evolutionary algorithm (MOEA) to explore the associated search space. The concept and design of a self-organized multi-objective evolutionary algorithm is developed in Section 5. In Section 6, addition of a new DE-inspired controller and more advanced swarm behaviors are shown. In Section 7 the system is extensively evaluated with swarms of heterogeneous vehicles in a distributed simulation system with animated graphics. Section 8 concludes with measures of success and possible future directions.

2 Generic Swarming Approaches

Approaches to control and swarming are as 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 swarming domain problem domain. Then several associated efforts with other techniques are discussed and briefly evaluated.

2.1 Algorithmic Structures/Frameworks

Our problem domain has a resemblance to Nikovski's [4] 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 [5] 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. And in A technique of utilizing a single objective genetic algorithm (GA)successfully accomplished dynamic environment reaction in [6].

There are several approaches to developing SO vehicle swarm controllers using genetic programming (GP) [7–11]. Woolley's [12] defines a control structure called the Unified Behavior Framework (UBF) which constrains the GP, with consistent success in a constrained problem. In [13] they develop a complex control architecture for agents utilizing the principles of artificial immune systems (AIS), with mild success.

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, finding simple behaviors sets and relations between them allow for lighter computational processing.

2.2 Rule Based Behavior Archetypes

When self-organization (SO) is applied to a system, rules sets must be developed in such a way that all solutions are feasible in sub-domains. In the case of our Swarmfare, the system gathers these rules together to form behavioral archetypes (BA). Through these groupings the rules are weighted and applied to establish each subsequent action. This is similar to the modes in Rosenblatt's architecture [14]. Although, this system uses a set of neural-network preceptrons to form the control agent and find the appropriate BA to use at a precise moment. This open scheme of BAs allows for flexible and quick response of different variations given the dynamic environment.

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

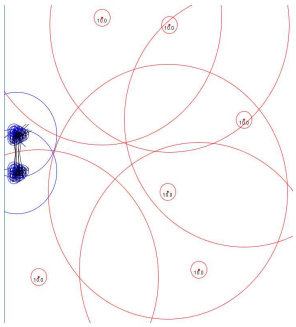

Figure 1: Release of a swarm of Heterogeneous vehicles (UAVS) in a target rich environment. The UAVs start clustered on the left engagement and sensor rings illustrated. The targets are show throughout the space with matching sensor and engagement rings. [16]

- 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 [15, 16]. The orbit stems from Lua's work [17]. 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 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. For example from [6], Figure 1 reflects the problem domain constrained environment or domain of the Swarmfare animated simulation package.

The objectives are two fold. First the system must establish a SO Swarm. The intent of this objective is to increase search effectiveness, safety, and attack force. The second objective requires the swarm to successfully engage targets. This means two things, maximized damage to the target and minimized causalities in the swarm. All of this must happen while moving through a populated terrain space.

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.

3.1 System Mapping to POMDP

Mapping this problem domain to a formal model requires explanation of several critical elements. First 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, Markovian assumptions are used. Without global knowledge and the inability 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 1, 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, the mapping of this specific swarming problem (target engagement) is extended to a POMDP model [18].

The agent actions are defined by the set of beliefs state transitions and the scope of the set defined by the following. Within each state $\mathcal{S}$ there exists a set of agents υ, set of targets τ and set of obstacles ζ in the domain space.

Agent

The agent (UAVs) is defined by the tuple in Equation 2

$$\upsilon(\lambda_{\upsilon t}, e_\upsilon, d_\upsilon) \tag{2}$$

Here the $\lambda_{\upsilon t}$ defines the location and velocity vector, e_υ is the engagement range of agents. d_υ is the detection range of agents. This state describes the agent from the global view.

Target

Equation 3 defines the target (SAM site) sets:

$$\tau(\lambda_\tau, e_\upsilon, d_\upsilon) \tag{3}$$

Again the λ_τ defines the location vector, e_υ is the engagement range of targets, and d_υ is the detection range of agents.

Obstacle

Equation 4 defines the obstacle set:

$$\zeta(\lambda_\zeta, s_\zeta) \tag{4}$$

Again the λ_τ defines the location vector and s_ζ defines repulsion field strength of the obstacle.

Action

The action set is defined by the agents actions Equation 5:

$$\{\mu_{A_1}, \ldots, \mu_{A_n}\} \epsilon \mathcal{A}(\mu) \tag{5}$$

The μ_{A_k} is the movement action of all agents taken in the domain. The null action does not exist in the set as the agent must always be moving while in flight. This also means velocity vector λ must not be 0. The set also includes other actions such as, detect, engage, turning and others depending on the capabilities of the agents.

Transition Set

The transition in the world domain is dependent on the independent agent action and interactions. Therefore Equation 6 show the transition probability between states.

$$T : P(s') = P(s', \Upsilon, s)P(s) \tag{6}$$

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 shown in Section 2.2.

The probability of entrance into a new state $P(s')$ is dependent on the previous state $P(s)$ and the set of agent actions, Υ. The agent state itself as a local view is extended with the I-POMDP model. The Interactive POMDP (Interactive Partially Observable Markov Decision Process) used by Doshi [19] specifically defines the state transitions of each agent based on the probability of the interactions with other agents. This approach focuses the agent actions based on its knowledge base and behavior set independent of the entirety of the domain.The agents local state and actions can be defined by the I-POMDP in Equation 7. The purposed I-POMDP model allows the agents to work separately and interactively update the global state.

$$I_i = \{IS_i, A, T_i, \Omega_i, O_i, R_i\} \tag{7}$$

The interactive state IS_i depends on the state S and the belief that other agents interact with that state, θ_j, using $IS_i = S \times \theta_j$.

Individual Agent State

The state of the agents in the I-POMDP motivates a further description of the agents. From the agents perspective on the domain, many local knowledge pieces need to articulated. Further development of the I-POMDP mathematical model for the UAV problem domain can provide predictability in the effectiveness through a detailed system verification (proof) of performance. At this point in the design process the system develops from the substates of the model but it does not employ full articulation of all the mathematic structures. Since the low-level sub-state design is done from an engineering perspective, further mathematical decomposition does not directly effect the associated computational validation. For this reason the design decomposition is continued from this level of the model, which is a reasonable representation of the real-world operations.

3.2 Simulation of Autonomous Vehicles

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

- Dynamics of the vehicles/agents (UAVs, robots, ...)
- Physics constraints of not only the craft but munitions
- Sensors range constraints and noise
- Communications bandwidth constraints and unreliability
- Geographic incursion on movement, sensors, and communications
- Fog and Friction of battle

All of these real-world constraints if hidden create a scenario where relying on details of a state 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 state and reality of the level of states that are used by the 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 Swarm Behavioral Control

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 weightings using a stochastic search technique. The swarming problem and algorithm domain is expanded into the multi-objective problem (MOP) realm using 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.

4.1 Chromosome Development

Formulation of the EA 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 relate to the control parameters or weighting of the rule sets.

Control Development Implementing the transactions defined by the behavior set in Section 2.2 requires synergistic integration of those behaviors. Here behaviors are vector fields that direct the agent's movement, similar to that seen in [20]. Arbitration in the benchmark work uses a multi-layer perceptron to chose between SO behaviors sets. The control weightings for this structure are also include with the behaviors sets weights in the chromosome shown in Figure 2. Here each control weighting 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 [6]. 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 (BAa). These BAs allow the system to dynamically develop several sets of weightings 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.

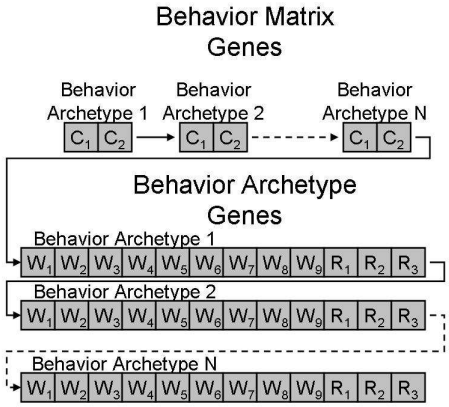

Figure 2: There is a connection weight for each sense for each behavior archetype. These are followed by 12 genes which describe the weights and radii for the behavior rules for each behavior archetype.

Figure 2 shows the representation used. The chromosome values are used to map the weightings 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) will only change 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. Equation 8 defines values based on successful engagement.

$$D_t = \tau_{destroyed} * 100 + \tau_{destruction} * 10 \tag{8}$$

The second fitness function is the casualty rate, defined in Equation 9:

$$C_t = \nu_{damage} * 10 \tag{9}$$

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

Crossover and Mutation Operators Normal crossover and mutation rates and controls apply. The difference in this implementation comes in what gets modified. In both instances an entire BA gets modified. The complex form of the chromosome is particular to this 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 parents and children to continue to the next generation.

4.2 Classical MOEA

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 also used because of its *fastnondominated* sort which finds possesses a balance between exploitation and exploration properties. [21] indicates increasingly effective solutions while maintaining a wider range, as compared to other MOEAs. Although such MOEAs such as NPGA, PAES, MOMGA, and SPEA2 could be employed and compared [21].

Because we desire that the NSGA-II minimizes the functions, there had to be a slight modification in the way the objective functions are shown in Section 4.1. 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 lower values which are better.

5 Self-Organized MOEAs

In the natural world many systems develop through Self Organization(SO); emergent properties evolve as a result of localized agent interaction sans global knowledge. [22–24] In DNA there is no global knowledge of the DNA structure from the allele's/nucleotides perspective, however the nucleotides do interact and structure does emerge. Although modern biochemistry does not state whether this organization is truly SO, the possibility that these agents, alleles/nucleotides, have properties similar to other SO systems exist. This is the foundational impetus for the SOGA.

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 these behaviors are easily coded. Lower computational cost stems all computations executing localized at the agent level and interaction and communications are also very localized. Finally dynamic response happens as emergent behaviors do not restrict the system to a script but instead are capable of quick response to unpredictable stimulus through those rules.

What we desire is that the terms of a SO GA reflect the response to problems 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. [25, 26] Here 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 look at modifying the highly varied world of recombination operators and two we add an operator to sense the current genotype space act upon that information. Recombination has been the studied extensively spawning many different approaches to satisfy different problem constraints.

SOGA Figure 3 shows the added operators and new algorithm flow. The calculate step determines the 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

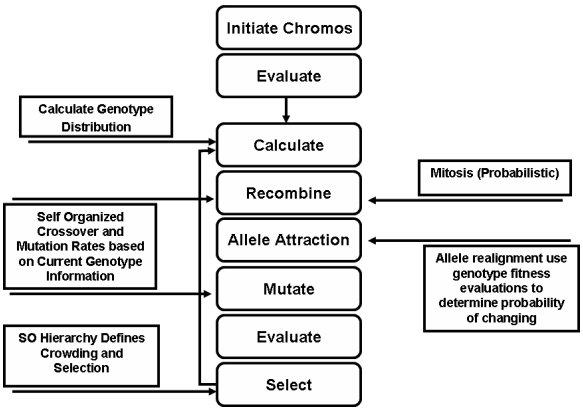

Figure 3: Self Organized Gene and Allele Genetic Algorithm SOGA

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 will continue 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 10 and 11 shows the updating function for the rates of mutation and crossover.

$$c = \frac{\sum_{i=0}^{chromo_{len}} (\Gamma_i(t-1) - \Gamma_i(t))}{\Gamma_{size}} \tag{10}$$

$$m = \frac{\sum_{i=0}^{chromo_{len}} (\Gamma_i(t-1) - \Gamma_i(t))}{\Gamma_{size}} * min((\Gamma_i(t-1) - \Gamma_i(t)! = 0) \tag{11}$$

Mitosis The second operator change comes in the recombination step. The process recognizes when good building blocks exist and attempts to perpetuate their existence. As shown in Figure 4 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 ϖ. If the normalized summation of the alleles in the crossover section is above the threshold it will choose 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.

$$P_{change}(x|\gamma) = \Gamma_{w_i} * (W_c - W_n) \tag{12}$$

Here both W_c and W_n are the normalized summation of correct and incorrect mappings, respectively, between Γ_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 *fastnondominatedsort* as NSGAII. 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

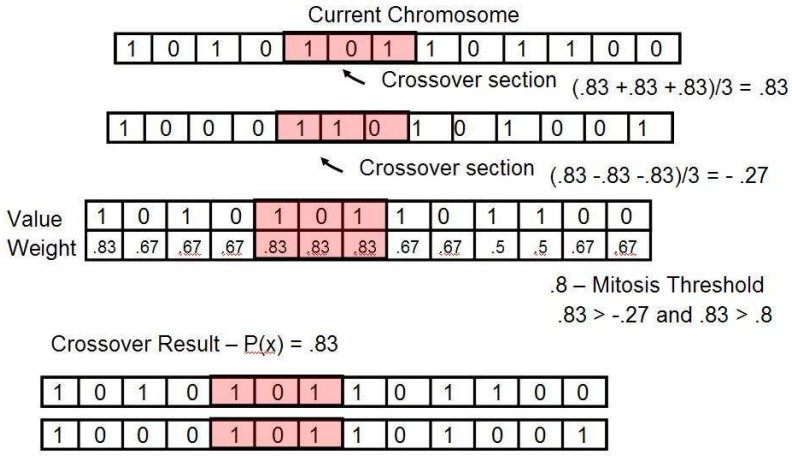

Figure 4: Crossover operator

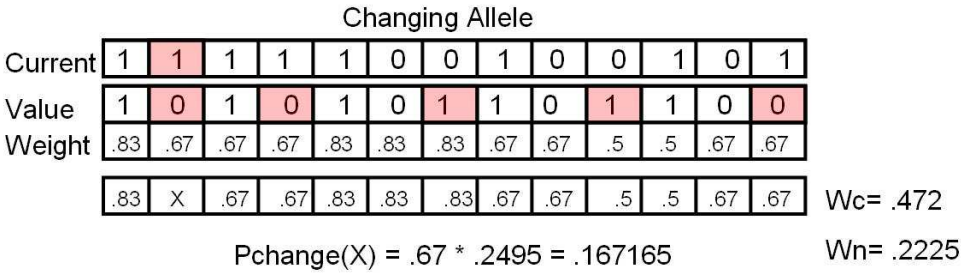

Figure 5: Changing the allele in red on the top chromosome through the weighting of the GDC shown below with the best known value and percent certainty below it. The result of the GDC for correct predictors (W_c) and non-correct predictors (W_n) is normalized and added. The probability of change is the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

of every objective. Then the individuals that qualify as neighbors use a SO ranking structure similar to that outlined in [27]. The remaining positions in the child population are filled based on the ranking structure. Early on 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 13 shows the formula for determining the probability of an individual winning an hierarchy engagement.

$$Prob_{i_{win}} = \frac{1}{1 + \exp\left(f_{sup} + (rank_i - rank_j)\right)} \tag{13}$$

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.

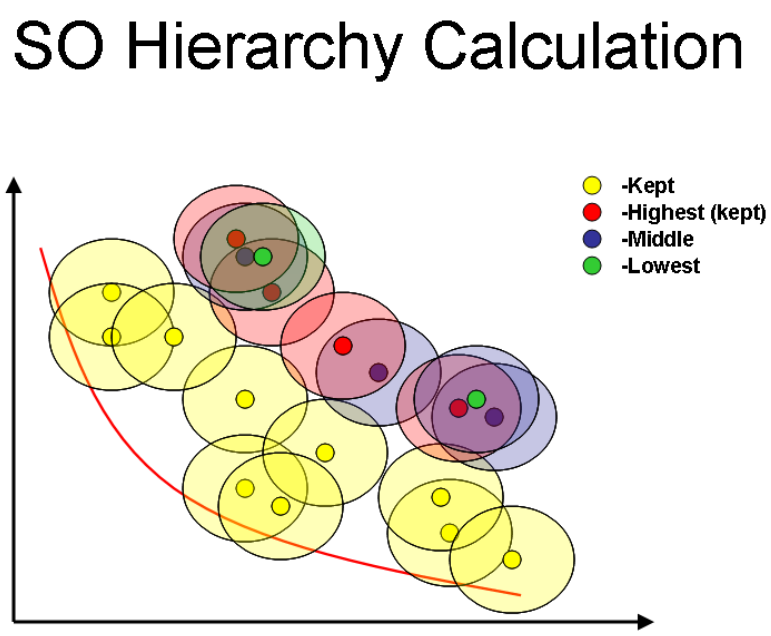

Figure 6: SO selection operator example. The inner circles are the individuals in the population. The outer circles represent the neighborhood. The yellow individuals are kept because of their rank, the red individuals have the highest levels in the SO hierarchy and are also kept.

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 through experience with the old chromosomes as well.

6 Advanced SO Swarm Control

This effort also focused on extending the combination of the basic SO swarm controls outlined by Reynolds [15]. 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 [6], 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, as shown in Figure 7.

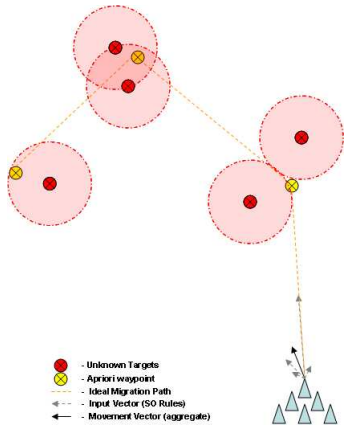

Figure 7: Migration patterns relative to target areas.

Advanced Attack Attacking "en mass" is only marginally effective in [6]. 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 [28]. Figure 8 shows the threshold decision process.

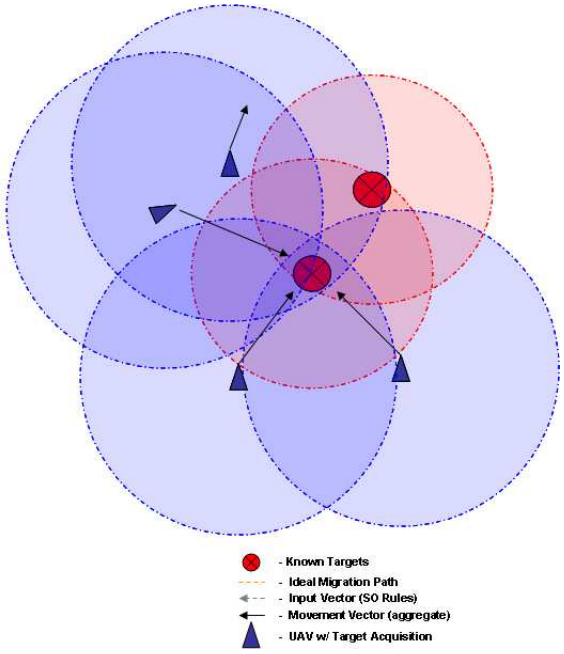

Figure 8: Attack is based on the number of agents in the swarm, the location of the targets and the number agents ready to engage.

2nd Generation Controller In order to arbitrate over the BAs in dynamic environments the DE-inspired controller [29,30] was developed in [18]. 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 14, where I defines the foci vector and BAw the weightings for that BA.

$$F = \sqrt{\sum \frac{I_k^2 - BA_k^2}{BAw_k}} \tag{14}$$

Given the controller has the same data structure as used by Differential Evolution, a DE mutation operator is added to the GA, as well. The this produces a situation where each BA has an area of influence on the abstracted state space, Figure 9 illustrates a two sensor input with three control foci.

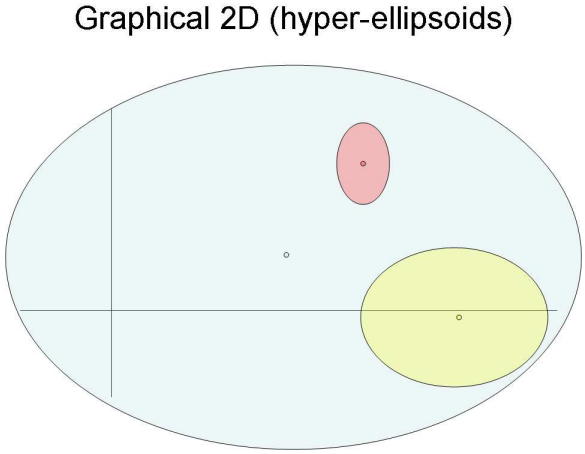

Figure 9: Three BA origin points and their hyper-ellipsoids

This new system architecture generates a control structure that mediates the set of Behavior Archetypes, (BAs). In addressing the complex entangled hierarchies introduced by the *emergent control structure* in SO and dynamic nature of the environment, the design of a control structure is of course particularly crucial. The required characteristics include the ability to handle dynamic domains, search the rugged solutions spaces created by the entangled structures, and develop multidimensional partitioning shapes.

To address these issues, the system allows an emergent structure to form with the basic form shown in Figure 10. Figure 10 illustrates that the information and algorithmic dependencies are not strictly linear but develop based on minimization of computational and informational complex between related states. The connection of abstracted UAV state to the environment state presents an entrance point for control of the UAV's BA . The abstracted state are formed in such a way that systems only take relevant information. Of course this is only a subset of the information in that state, that which is perceivable by the agent.

The previous work [6] used a controller that was based in Neural Networks (NN) . Choosing the BA in this setting, presents a problem because the space is extremely dynamic. In order to be effective NN need extensive and robust sample sets [31]. This is not always available in a probabilistic environment. In response to this phenomena, the crucial DE control approach chooses between BAs that can handle unknown states.

7 Design of Experiments

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 known results. Specific experimental objectives are discussed within each set of tests along with statistical evaluation measures.

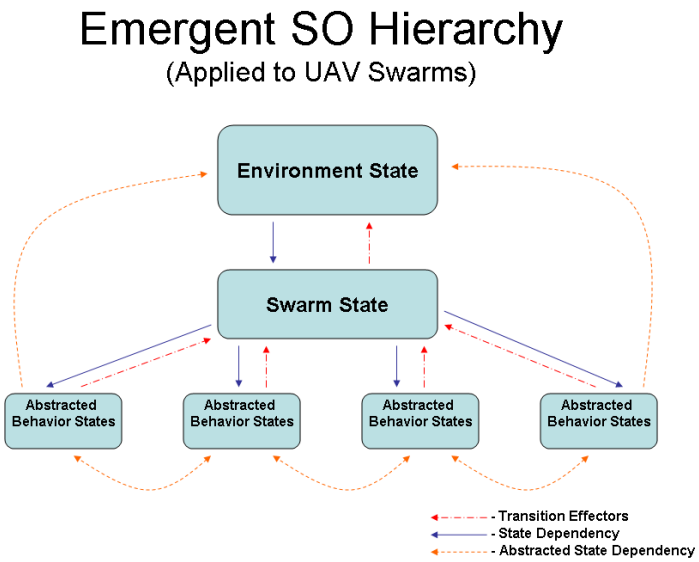

Figure 10: Relationships based on the direct, indirect or abstracted state translation for SO.

The system is tested against the benchmarks outlined in [6]. 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 flight dynamics are used to model the UAVs in the 2D environment. Engagement happens probabilistically with relative strengths of the targets and UAVs defined. Each 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. 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 GA'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 ϵ-indicators on the final Pareto Front because of the unknown $\mathcal{PF}_{true}$.

7.1 Results

The proposed dynamic UAV systems are computationally tested on the AFIT HPC clusters. These HPCs use the Linux operating system, with dual core processing at between 2.0 and 3.0 Ghz. The Infiniband crossbar backplane gives virtual appearance of a fully connected network. The Swarmfare system utilizes a farming model for distributed/parallel execution. All results from both data sets with the GAs are shown.

First NSGA-II was tested through the original configuration described in Section 7. 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 6. Final test include a run through the original 6 scenarios and then 8 more scenarios to specifically design to exploit the new control and attack configurations. Because of this the finally run of the original 6 scenarios was completed in both cases and used for comparison.

7.1.1 Statistical Analysis

Figure 7.1.1 compares the final statistics of the system with the original. The diamond line to the far left shows the growth. The achieved successes in attack are notable. The top damage number moved from 280 (or just shy of 3 targets) to 430 (or solidly over 4 targets of 6). [6] showed approximately 2/3 success in

target kill as well, but the system did not consider casuality rates. Upon visual inspection the difference is notable. Agents rarely colloid or fly out of the target area with the MOP implementation and improved controls.

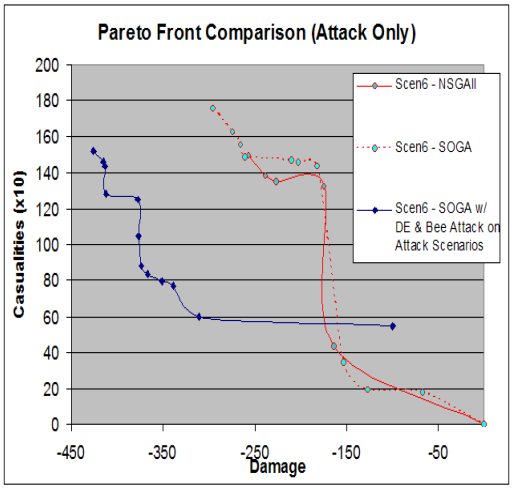

Figure 11: This graph compares the $\mathcal{PF}known$ of various scenarios.

Table 1 presents 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.

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

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 gives the system more effective target engagement.

8 Conclusion

This discussion and development is to establish an autonomous aerial vehicle, UAV, swarms or agent systems that through limited user expert knowledge, desired autonomous behavioral control structures could be developed. The initial aspect is the establishment of an entangled hierarchical swarm control structure through self-organization concepts, SO. The exploitation of modularized SO behaviors with emergent properties facilitated the reconnaissance movement through the problem domain space and rudimentary bio-inspired engagement with targets. The individual agent control structures are simple, but effective, without degrading the capabilities of the overall SO emergent behavior.

The specific aspect sought to find the "optimized" weight ratios for the different agent rules and BAs again without expert a priori knowledge. Through the SO Genetic Algorithm, SOGA, the system is able to *optimize*

in a dynamic unknown search space; the multi-objective fitness landscape. The dynamic innovative process via computational testing did so without requiring parameter tuning on any of the mutation or crossover rates, the selection operator, the crowding distance neighborhoods or the SO allele attraction operator. The system in all scenarios preformed as well if not better than the NSGA-II. The addition of advanced attack techniques and a more flexible controller created a system that moved the system ever closer to employment. Via simulation, an animation of the bio-inspired UAV movement and attack are demonstrated for various scenarios. Future research should focus on exploiting SO capabilities in a more realistic physical realm. Effective use of SO UAV swarms is a force multiplier which reduces the risk to combatants, and increases effectiveness of policy implementation in various governmental and industrial applications.

9 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). Also, Dustin Nowark for his research and MS thesis which is one of the foundations for this discussion.

References

[1] Aerovironment, "Raven rq-11b," Datasheet, June 2007. [Online]. Available: http://www.aerovironment.com/

[2] R. J. Wood, S. Avadhanula, E. Steltz, M. Seeman, J. Entwistle, A. Bachrach, G. Barrows, S. Sanders, and R. S. Fearing, "An autonomous palm-sized gliding micro air vehicle," *IEEE Robotics and Automation Magazine*.

[3] N. Glauvitz, "Towards a flying mems robot," Master's thesis, Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2007.

[4] D. Nikovski and I. Nourbakhsh, "Learning probabilistic models for decision-theoretic navigation of mobile robots," in *Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2000, pp. 671–678. [Online]. Available: citeseer.ist.psu.edu/nikovski00learning.html

[5] D. Khosla and T. Nichols, "Hybrid evolutionary algorithms for network-centric command and control," in *Defense Transformation and Network-Centric Systems. Edited by Suresh, Raja. Proceedings of the SPIE, Volume 6249, pp. 624902 (2006).*, ser. Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, vol. 6249, Jun. 2006.

[6] I. C. Price, "Evolving self organizing behavior for homogeneous and heterogeneous swarms of uavs and ucavs," Master's thesis, Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2006.

[7] K. M. Milam, "Evolution of control programs for a swarm of autonomous unmanned aerial vehicles," Master's thesis, Air Force Instituite of Technology, 2004.

[8] R. V. Michael A. Kovacina, Daniel W. Palmer, "Swarm rule-base development using genetic programming techniques," in *Workshop on Autonomous Swarm Programming (WASP),*, 2003.

[9] G. J. Barlow and C. K. Oh, "Robustness analysis of genetic programming controllers for unmanned aerial vehicles," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2006, pp. 135–142.

[10] C. Lazarus and H. Hu, "Using genetic programming to evolve robot behaviours," in *Proceedings of the 3 rd British Workshop on Towards Intelligent Mobile Robots (TIMR) '01, Manchester*, 2001.

[11] R. Poli, W. Langdon, and O. Holland, "Extending particle swarm optimization via genetic programming," 2005.

[12] B. G. Woolley and G. L. Peterson, "Genetic evolution of hierarchical behavior structures," in *2007 Genetic and Evolutionary Computation Conference*, D. T. et al., Ed., vol. II. Associaition for Computing Machinery, 2007, pp. 1731–1738.

[13] H. Y. K. Lau, V. W. K. Wong, and I. S. K. Lee, "Immunity-based autonomous guided vehicles control," *Appl. Soft Comput.*, vol. 7, no. 1, pp. 41–57, 2007.

[14] J. K. Rosenblatt, "DAMN: A distributed architecture for mobile navigation," in *Proc. of the AAAI Spring Symp. on Lessons Learned from Implemented Software Architectures for Physical Agents*, Stanford, CA, 1997. [Online]. Available: citeseer.ist.psu.edu/article/rosenblatt97damn.html

[15] C. W. Reynolds, "Steering behaviors for autonomous characters," in *Proceedings of the 2005 Winter Simulation Conference. San Jose, California*, 2005, pp. 763–782.

[16] ——, "Flocks, herds, and schools: A distributed behavioral model," *Computer Graphics*, vol. 21, pp. 265–280, 1987.

[17] K. A. Lua, Chin A. and K. E. Nygard, "Synchronized multi-point attack by autonomous reactive vehicles with local communication," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, 2003.

[18] D. Nowak, "Exploitation of self organization in uav swarms for optimization in combat environments," Master's thesis, Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2008.

[19] P. Gmytrasiewicz and P. Doshi, "Interactive pomdps: Properties and preliminary results," 2004. [Online]. Available: citeseer.ist.psu.edu/gmytrasiewicz04interactive.html

[20] A. T. Hayes and P. Dormiani-Tabatabaei, "Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots," in *In Proceedings of the 2002 IEEE International Conference on Robotics and Automation IROS-02*, 2002, pp. 3900–3905.

[21] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[22] F. Heylighen and C. Joslyn, "Cybernetics and second-order cybernetics," *Encyclopedia of Physical Science & Technology*, vol. 3rd ed., 2001, academic Press, New York.

[23] R. Cottam, W. Ranson, and R. Vounckx, "Autocreative hierarchy ii: dynamics self-organization, emergence and level-changing," in *Integration of Knowledge Intensive Multi-Agent Systems, 2003. International Conference on*, 30 Sept.-4 Oct. 2003, pp. 766–773.

[24] S. Camazine, *Self Organization in Biological Systems*. USA: Princeton University Press, 2003.

[25] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," IBM Almaden Research Center and Sante Fe Institute, 1996.

[26] C. A. Coello Coello and G. B. Lamont, Eds., *Application of Multi-Objective Evolutionary Algorithms*. World Scientific Publishing, 2004.

[27] M. Tsujiguchi and T. Odagaki, "Self-organizing social hierarchy and villages in a challenging society," *Physica A Statistical Mechanics and its Applications*, vol. 375, pp. 317–322, Feb. 2007.

[28] T. D. S. P. K. Visscher, "Quorum sensing during nest-site selection by honeybee swarms," *Behavioral Ecology and Sociobiology*, vol. 56, pp. 594–601, 2004.

[29] H. Abbass, "Self-adaptive pareto differential evolution," 2002. [Online]. Available: citeseer.ist.psu.edu/abbass02selfadaptive.html

[30] H. A. Abbass, R. Sarker, and C. Newton, "PDE: A pareto-frontier differential evolution approach for multi-objective optimization problems," in *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*. COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea: IEEE Press, 27-30 2001, pp. 971–978. [Online]. Available: citeseer.ist.psu.edu/abbass01pde.html

[31] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach 2nd Ed.*, M. J. Horton, Ed. Prentice Hall, 2003.